INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI
NOTE TO USERS

Page(s) not included in the original manuscript are unavailable from the author or university. The manuscript was microfilmed as received.

66

This reproduction is the best copy available.
ORGANIZING, AND DOCUMENTING
COMPONENT–ORIENTED TOOLKITS

By
MOHAMMAD AHMAD RADAIDEH, M.Sc.

A Thesis
Submitted to the School of Graduate Studies of McMaster University
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy
2000

McMaster University
© Copyright by Mohammad Ahmad Radaideh, 2000
ORGANIZING, AND DOCUMENTING COMPONENT-ORIENTED TOOLKITS
Doctor of Philosophy (2000)  
(Electrical and Computer Engineering)  
McMASTER UNIVERSITY  
Hamilton, Ontario, Canada

TITLE: Organizing, and Documenting Component-Oriented Toolkits

AUTHOR: Mohammad A Radaideh, M.Sc. (Jordan University of Science and Technology)

SUPERVISORY COMMITTEE:

Professor David Lorge Parnas (Supervisor)  
Professor Ryszard Janicki  
Professor Martin von Mohrenschmidt  
Professor Ridha Khedri

EXAMINATION COMMITTEE:

Professor D. Shaw  
Professor D. Schmidt  
Professor H. Stover  
Professor A. Hitchcock  
Professor D. Parnas  
Professor M. Mohrenschmidt  
Professor R. Khedri  
(Chair)  
(External Examiner)  
(Examiner – McMaster)  
(Examiner – McMaster)  
(Supervisor)  
(Supervisory Committee Member)  
(Supervisory Committee Member)

NUMBER OF PAGES: xiii + 407 pages
ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor Professor David Parnas for his continuous support and guidance since September 1995. I would like to thank as well, Professor Ryszard Janiscki, Dr. Martin von Mohrenschilddt, and Dr. Ridha Khedri for serving on my supervisory committee. Also, I would like to Professors Shaw, Stover, and Hitchcock from McMaster university for serving on my examining committee. A special thank you goes to Dr. Douglas Schmidt for the detailed comments he provided in his report as the external examiner for my thesis.

I gratefully acknowledge the financial support provided by the Telecommunication Research Institute of Ontario (TRIO), and the Department of Electrical and Computer Engineering at McMaster University during the first two years of my study at McMaster. Special thank you goes to Michael White from IBM Canada for his continuous encouragement and support to complete this thesis.

My parent, brothers, and sisters deserve much more than “thank you” for their sincere encouragement.

Finally, I would like to express my grateful appreciation and special thanks to my wife Rawiah, and our beloved sons Sa’eb, Ahmad, Saleh, and Khaled for their continuous patience, love, encouragement, and moral support throughout these long years.
ABSTRACT

Component–Oriented Software Technology started to emerge during the last few years. The purpose is to make it easier to build new applications from existing components that can be obtained from Component–Oriented Toolkits (CO–Toolkits for short). Experience shows that CO–Toolkits are hard to understand and use. Therefore, developers would write their own code from scratch using programming languages that they know instead of using components from such CO–Toolkits.

In this research, we have used ADAPTIVE Communication Environment (ACE for short) – as an example of CO–Toolkits – to establish a set of initial assumptions for good design practice from the perspective of their file structures, design and implementation, and documentation of CO–Toolkits. Then (i) we show how to formalize rules for good file structures, design and implementation, and documentation that are based on these initial assumptions, (ii) we show that checking tools can be built based on these formalized rules, and then (iii) we demonstrate that such tools can be used to check existing CO–Toolkits code and report any rule violations back to developers.
TABLE OF CONTENTS

ACKNOWLEDGEMENTS ........................................ iv

ABSTRACT ............................................... v

TABLE OF CONTENTS ..................................... vi

LIST OF FIGURES ........................................ xiii

CHAPTER 1 INTRODUCTION ............................... 1
1.1 Overview ............................................ 1
1.2 Research Motivation ................................. 6
1.3 Related Work ......................................... 7
  1.3.1 The “Automatic Detection of Design Problems” Work .............................. 7
    1.3.1.1 What Is The “Automatic Detection of Design Problems”? ...................... 7
    1.3.1.2 Our Work vs. The “Automatic Detection of Design Problems” ............... 8
1.3.2 Rigi .................................................. 9
  1.3.2.1 What is Rigi? .................................. 9
  1.3.2.2 Our Prototypes vs. Rigi .................................. 9
1.3.3 Together/C++ ................................... 10
  1.3.3.1 What is Together/C++? ......................... 10
  1.3.3.2 Our Prototypes vs. Together/C++ ......................... 10
1.3.4 DATRIX ........................................... 11
  1.3.4.1 What is DATRIX? ............................... 11
  1.3.4.2 Our Prototypes vs. DATRIX ......................... 11
TABLE OF CONTENTS

1.3.5 Other Parsing Tools ............................................. 11
    1.3.5.1 Our Prototypes vs. Other Parsing—Tools ............ 12
1.4 Our Audience ..................................................... 12
1.5 The Primary Common Characteristics of CO—Toolkits ......... 13
1.6 Key Contributions Made By This Research .................... 14
1.7 Thesis Organization .............................................. 14

Chapter 2 CASE STUDY: STRUCTURE AND ANALYSIS ............... 17

2.1 Overview ......................................................... 17
2.2 Case Study: ADAPTIVE Communication Environment .......... 18
    2.2.1 What Is The “ADAPTIVE Communication Environment”? 18
2.2.2 Why Did We Choose ACE As A Case Study? ................. 20
2.2.3 Are Our Views ACE—Dependent? ........................... 20
2.3 File Structures .................................................. 20
    2.3.1 Why Do We Need To Study File Structures? ............ 20
2.3.2 Visualizing ACE File Structure ............................ 21
    2.3.2.1 ACE File Structures Graphs .......................... 21
    2.3.2.2 What Issues Do These Figures Raise? ............... 23
        2.3.2.2.1 Inclusion Loops ............................... 26
        2.3.2.2.2 Inclusion Multiple Paths .................... 26
2.4 Implementation .................................................... 29
    2.4.1 Inheritance ............................................... 29
        2.4.1.1 What Is Inheritance ............................. 29
        2.4.1.2 Does Multiple Inheritance Causes Any Additional
                Difficulties? .................................... 29
        2.4.1.3 Why Do We Need To Study Inheritance? .......... 30
        2.4.1.4 ACE Inheritance ................................ 30
    2.4.2 Association .................................................. 31
        2.4.2.1 What Is Association? ............................ 31
        2.4.2.2 Why Do We Need To Study Association? .......... 32
        2.4.2.3 ACE Association ................................. 32
2.5 Documentation ................................................... 34
    2.5.1 What Documents Did We Find With ACE? ................. 34
    2.5.2 Are Such Documents Enough From The User Point Of View? 34
2.6 Conclusions ...................................................... 34

vii
TABLE OF CONTENTS

Chapter 3 ASSUMPTIONS FOR GOOD DESIGN .................................. 37
  3.1 Overview ................................................................. 37
  3.2 Software Engineering Principles ...................................... 38
    3.2.1 Separation of Concerns ........................................... 38
      3.2.1.1 Overview ............................................... 38
      3.2.1.2 Information Hiding ....................................... 39
      3.2.1.3 Abstraction .............................................. 39
    3.2.2 Modularity ...................................................... 39
    3.2.3 Formality ....................................................... 40
    3.2.4 Mastering Complexity And Design for Change .................. 40
  3.3 Assumptions For Good Software Design Practice ..................... 41
    3.3.1 General Assumptions ........................................... 41
    3.3.2 Assumptions For Good File Structures .......................... 42
    3.3.3 Assumptions For Good Implementation Structures .............. 45
    3.3.4 Assumptions For Good Documentation ........................... 47
  3.4 Conclusions ....................................................... 48

Chapter 4 FIRST VIEW: FORMALIZING RULES FOR ORGANIZING
FILE STRUCTURES OF CO-TOOLKITS ................................. 49
  4.1 Overview ............................................................. 49
  4.2 Towards A Mathematical View For CO-Toolkits’ File Structures .. 49
    4.2.1 Purpose ......................................................... 49
    4.2.2 The View Structure ............................................. 50
    4.2.3 The View Characteristics ..................................... 53
      4.2.3.1 Explanation of (4–5): GoodFileStructure() ............. 54
      4.2.3.2 Explanation of (4–7): GoodDistinctionOfFiles() ....... 55
      4.2.3.3 Explanation of (4–8): GoodFileInclusionRelation() ... 56
      4.2.3.4 Explanation of (4–11): GoodUseOfFiles() ............... 56
      4.2.3.5 Explanation of (4–13): GoodUseOf_top_Files() ......... 58
      4.2.3.6 Explanation of (4–14): GoodUseOf_intermediate_Files() .. 60
      4.2.3.7 Explanation of (4–15): GoodUseOf_Internal_Files() .... 60
      4.2.3.8 Explanation of (4–16): GoodUseOf_Shared_Files() ...... 61
    4.2.4 The View Lexicon ............................................... 61
      4.2.4.1 Variables And Acronyms ................................... 62
      4.2.4.2 Symbols And Constants ................................... 62
CHAPTER 5 SECOND VIEW: FORMALIZING RULES FOR DESIGNING AND IMPLEMENTING CO-TOOLKITS

5.1 Overview .................................................................................. 65
5.2 Towards A Mathematical View For CO-Toolkits’ Design and Implementation Structures .................................................. 65
  5.2.1 Purpose .................................................................................. 66
  5.2.2 The View Structure ................................................................. 66
    5.2.2.1 Component: Component_{ij} As In (5–2) ...................... 69
        5.2.2.1.1 Component’s Interface: Interface_{ij} As In (5–4) ... 69
        5.2.2.1.2 Component’s Implementation:
                      Implementation_{ij} As In (5–4)...................... 72
  5.2.3 The View Characteristics ..................................................... 74
    5.2.3.1 Explanation of (5–18):
                      GoodDesignAndImplementation() .......................... 74
    5.2.3.2 Explanation of (5–19):
                      GoodUseOfRelationsAmongComponents() ............... 75
    5.2.3.3 Explanation of (5–20): SafeInheritance() ..................... 77
    5.2.3.4 Explanation of (5–21): GoodAssociation() .................. 79
    5.2.3.5 Explanation of (5–24):
                      SafeInheritance_AmongComponents_WithinTheSameFramework() 81
    5.2.3.6 Explanation of (5–25):
                      SafeInheritance_AmongClasses_WithinTheSameComponent() 82
    5.2.3.7 Explanation of (5–26):
                      SafeInheritance_AmongClasses_WithinTheSharedPools() ... 83
    5.2.3.8 Explanation of (5–27):
                      GoodAssociationOnTheComponentLevel() .................. 84
    5.2.3.9 Explanation of (5–28):
                      GoodAssociationOnTheInternalClassesLevel() ............. 85
  5.2.4 The View Lexicon ................................................................. 85
    5.2.4.1 Variables And Acronyms ............................................... 85
    5.2.4.2 Symbols And Constants ............................................... 87
    5.2.4.3 Auxiliary Functions ..................................................... 88
4.3 Conclusions ............................................................................ 89
### TABLE OF CONTENTS

#### CHAPTER 6  THIRD VIEW: FORMALIZING HOW TO DOCUMENT CO-TOOLKITS  .................................... 91

6.1 Overview ............................................. 91

6.2 Component Guide Document ........................................... 92


6.2.2 Who Is A “Component Guide Document” intended For? ...... 93

- 6.2.2.1 Maintainers ....................................... 93
- 6.2.2.2 Users (Developers of CO-Applications) .............. 94
- 6.2.2.3 Sales Staff ....................................... 94
- 6.2.2.4 Technical Support .................................. 94

6.2.3 When Should A “Component Guide Document” Be Read? ... 94

6.3 Component Interface Specifications Document .................. 95

6.3.1 What Information Should Be Contained In A “Component Interface Specifications Document”? And How Should It Be Organized? ................................................. 95

6.3.2 Who Is A Component Interface Specifications Document” intended For? ........................................ 96

6.3.3 When Should A Component Interface Specifications Document” Be Read? ........................................ 96

6.4 Component Internal Design Specifications Document ........... 96

6.4.1 What Information Should Be Contained In A “Component Internal Design Specifications Document”? And How Should It Be Organized? ............................................. 97

6.4.2 Who Is A “Component Internal Design Specifications Document” intended For? ........................................ 97

6.4.3 When Should A “Component Internal Design Specifications Document” Be Read? ........................................ 97

6.5 Towards A Mathematical View For CO-Toolkits’ Documentation 98

6.5.1 Purpose ............................................. 98

6.5.2 The View Structure ....................................... 98

- 6.5.2.1 Explanation of The CGD Document In (6-1)
  
  COToolkit .............................................. 99

- 6.5.2.2 Explanation of The CISD Document In (6-1)
  
  COToolkit .............................................. 99

- 6.5.2.2.1 Explanation of (6-3): CISD ................. 100

- 6.5.2.2.2 Explanation of (6-4):
  
  CISD_Chapter_Framework

  100

x
### TABLE OF CONTENTS

6.5.2.2.3 Explanation of (6–5):

- \( \text{CIDSD}_\text{Section}_\text{Component}_{ij} \) 101

6.5.2.2.4 Explanation of (6–8):

- \( \text{CIDSD}_\text{SubSection}_\text{CANONICAL_REPRESENTATION}_{ij} \) 104

6.5.2.3 Explanation of The CIDSD Document In (6–1)

- \( \text{COToolkit} \) ........................................ 104

6.5.2.3.1 Explanation of (6–12): \( \text{CIDSD} \) ............... 105

6.5.2.3.2 Explanation of (6–13):

- \( \text{CIDSD}_\text{Section}_\text{Component}_{ij} \) .................. 106

6.5.2.3.3 Explanation of (6–14):

- \( \text{CIDSD}_\text{Chapter}_\text{SharedPool}_{l} \) 107

6.5.2.3.4 Explanation of (6–15):

- \( \text{CIDSD}_\text{Section}_\text{Component}_{ij} \) 108

6.5.2.3.5 Explanation of (6–16):

- \( \text{CIDSD}_\text{Section}_\text{SharedClass}_{ij} \) 109

6.5.2.3.6 Explanation of (6–18):

- \( \text{CIDSD}_\text{SubSection}_\text{InternalClass}_{ij} \) 109

6.5.3 The View Characteristics 111

6.5.4 The View Lexicon 112

6.5.4.1 Variables And Acronyms 112

6.5.4.1.1 CGD's Variables And Acronyms 112

6.5.4.1.2 CISD's Variables And Acronyms 112

6.5.4.1.3 CIDSD's Variables And Acronyms 114

6.5.4.2 Symbols And Constants .................................. 115

6.6 Conclusion .................................................. 116

### CHAPTER 7 INITIAL CONDITIONS THAT MAKE THE THREE VIEWS CONSISTENT .......................................................... 117

7.1 Overview .......................................................... 117

7.2 Conditions That Relate All Three Views .......................... 118

7.3 Conditions That Relate The First Two Views .................... 121

7.4 Conditions That Relate The Second And Third Views .......... 123

7.4.1 Conditions That Relate The Second View And Third View's 
CIDSD Document .................................................. 123

7.4.2 Conditions That Relate The Second View And Third View's 
CIDSD Document .................................................. 124
LIST OF FIGURES

Figure 1–1: CO–Toolkits’ Checking Cycle ........................................... 6
Figure 2–1: Overall ACE File structure .............................................. 22
Figure 2–2: REACTOR Framework File Structure ............................... 24
Figure 2–3: Synch.h File .............................................................. 25
Figure 2–4: Log_Msg.h File ............................................................ 27
Figure 2–5: Time_Value.h File ......................................................... 27
Figure 2–6: OS.h File ................................................................. 28
Figure 2–7: Inheritance in the REACTOR framework ......................... 31
Figure 2–8: Inheritance in the MEMORY framework .......................... 31
Figure 2–9: Association in the REACTOR framework .......................... 33
Figure 2–10: Association in the MEMORY framework .......................... 33
Figure 4–1: Motivating example: Model for CO–toolkits’ file structures .... 51
Figure 5–1: Motivating example: Model for inheritance relation in CO–toolkits .... 67
Figure 5–2: Motivating example: Model for association relation in CO–toolkits .... 70
CHAPTER 1

INTRODUCTION

1.1 Overview

Component–Oriented toolkits (i.e. CO–Toolkits for short) started to emerge during the last few years. CO–Toolkits provide developers with components that can be used to build new applications. This can save developers a lot of time since they do not have to write their applications from scratch.

The main objectives of this research are: (i) to show how to formalize rules and guidelines for good design practice, (ii) to show that such formalized rules can be used to develop tools, and (iii) to show that such tools can be used to check CO–Toolkits’ code against these formalized rules. In order to demonstrate the practical value of these results, and for the reasons mentioned in the next chapter (c.f. section 2.2.2), we chose the “ADAPTIVE Communication Environment” (i.e. ACE for short) toolkit as a case–study for this research [41, 1, 43–52].

Many words (i.e. frameworks, component, etc.) are commonly used by software developers with inconsistent meanings. In the following paragraphs, we show how such words are used throughout this thesis.
CHAPTER I: INTRODUCTION

In [31], Meyer stated that "there is no generally accepted definition for components". However, he described a component as "a program element with the following properties: the element may be used by other program elements (client), and the clients and their authors do not need to be known to the element's author". In [2], Nelson referred to components as "classes or frameworks". In [19], Fayad and Schmidt defined components as "self-defined instances of abstract data types (ADTs) that can be plugged together to form complete applications". In [28], Lakos defined a component as follows: "a component is the smallest unit of physical design".

As there is no generally accepted definition for components, a component (i.e. throughout this thesis) is a pre-defined application that is composed of a set of internal classes, and a main part such that (i) an internal class is a class in the context of object-oriented languages, and (ii) a main part is a program that contains the component's execution entry point (i.e. main function). Also, a main part can contain other functions that do not belong to any of the component's internal classes.

For convenience, we used three files for each component. We refer to them as the top, intermediate, and internal files. The top file is intended to hold the component's main part, the intermediate one is intended to hold the component's interface, and the internal one is intended to hold the data structure of the component as well as the implementation of all of its internal functions and methods.

According to Schmidt [43], the "component collaboration" can be defined as follows: "a set of components is considered to be collaborating if they exchange services".

The "component invocation" is used throughout this thesis in the sense that a component is considered invoked by an application when that component is executed at
CHAPTER 1: INTRODUCTION

run time from within that application. In such a case, the user’s application needs only to have a proper path to the directory under which the invoked component is stored. Also, it is considered invoked when its classes are “inherited from” and/or “associated with” by the application. In this case, the application needs to include the files used to implement the invoked component.

According to Schmidt [43], a “framework” is defined as follows: “A framework is an integrated collection of components that collaborate to produce a reusable architecture for a family of related applications”. For example, a set of components that provide signal detection, signal sampling, signal data manipulation and processing, etc. can be considered as a single framework that provide a set of relevant signal processing services.

We define a “CO–Toolkit” as a toolkit that provides developers with components. Inside a CO–Toolkit, components are grouped into a set of frameworks. In order to keep frameworks within the same CO–Toolkit independent from each other, shared–pools are suggested such that each shared–pool contains the set of classes and other entities that are shared by a number of the frameworks.

The word “good” is used throughout this thesis in the sense that a “good” thing is a thing that can be “useful” and “beneficial”. For example, we have discussed in chapter 3 the assumptions for “good” design practice. We think that these assumptions can be useful if they are considered in the construction of CO–Toolkits.

The word “perspective” is used throughout this thesis in the sense that a software perspective can be described in a formal relation. For example, file structures, imple-
mentation, and documentation of CO-Toolkits are three separate perspectives of CO-Toolkits.

The word "view" is used throughout this thesis in the sense that a "view" is an abstraction for reducing complexity of CO-Toolkits from a particular perspective through a formal structure and a set of rules and guidelines. Others used the word "view" for meanings that are close to our meaning for the word "view". For example, Embley [17] defined the word "view" as an abstraction mechanism for reducing complexity in large object-oriented system analysis models. Others like Ayers [3], Goldberg [22], and Rumbaugh [42] defined the word "view" as the method of presenting the information contained in the underlying model of an application.

The word "concern" is used throughout this thesis in the sense that a software concern is a matter or a design decision that needs the attention of the software developer who should be able to modify that concern without having to worry about the detail of the software's other concerns. According to the Webster's dictionary [29], a concern is defined as "a matter that engages a person's attention, interest, or care".

We have looked at ACE—as an example—from the perspectives of its file structure, relations (i.e. inheritance and association) among its components, and its documents (c.f. chapter 2). Based on (i) what we learned from ACE, (ii) software design principles as explained in the first half of chapter 3, and (iii) object-oriented programming experience gained during the last several years, we made a set of assumptions that can be good for software design practice (c.f. chapter 3).

Based on the assumptions referenced in the previous paragraph, we derived and formalized a set of rules for: (i) how to organize CO-Toolkits' file structures (c.f. chapter
CHAPTER I: INTRODUCTION

4), (ii) how to implement CO–Toolkits' components (c.f. chapter 5), and (iii) how to document CO–Toolkits (c.f. chapter 6).

Based on the formalized rules (i.e. views) referenced in the previous paragraph, we developed prototype systems that can be used to check CO–Toolkits' file structures (c.f. Appendix A), and their implementation structures (c.f. Appendix B) against these formalized rules. These prototypes report any violation to any of these formalized rules back to developers (c.f. Appendixes A2–A4, B2–B4). Java has been used to write these prototypes because we are familiar with it.

The formal models presented in this thesis (c.f. chapters 4–6), and the prototypes (c.f. appendices A and B) are intended to be general, but they are incomplete. The "Future Work" section (c.f. chapter 8, section 8.4) describes further potential improvements on the rules and the prototypes. Our prototypes were intended just to show that formalized rules can be used to develop checking tools. Therefore, we think that it is not necessary to provide any front ends that can be used to directly access CO–Toolkits' source code.

As shown in Figure 1–1 next, we expect a separate front end to extract the necessary artifacts from the CO–Toolkit's source code and build the various relations (i.e. the file structure, implementation, documentation, and consistency relations) that are required by the prototypes in order to check each for its own rules. Given that the purpose of our prototypes is just to show that formalized rules can be used to build tools, we have chosen to develop prototypes only for the first two views. Building prototypes for the third view and the views' consistency is left for future work (c.f. chapter 8, section 8.4:(4)).
This chapter has seven sections. The second section describes the motivation for this research. The third section describes some of the related work. The fourth section describes the audience intended by this research. The fifth section describes the primary common characteristics of the CO-Toolkits. The sixth section describes the main contributions made by this research. The last section describes how this thesis is organized.

1.2 Research Motivation

The motivation for this research can be summarized as follows:

(1) Upon starting this research, the guiding principle was "before using components from a given CO-Toolkit, developers need to understand that toolkit first, and to understand it, they need proper documents".

(2) CO-Toolkits provide components that can be used to build new applications. This is intended to simplify, enhance, and speedup the process of software development. However, since proper documents are not available with such systems, users of CO-Toolkits find it hard to understand and use their components. As a result, the process of software development slows down.
CHAPTER 1: INTRODUCTION

(3) Precise documents can help to better understand CO-Toolkits. Such documents can save developers a lot of time. As a result, the process of software development becomes faster. For example, ACE documentation comes in a set of academic papers [43–52] that we think were intended for people who are already familiar with ACE. Users can not just rely on these papers to understand how to use components from ACE.

(4) The need for precise documents encouraged us to investigate how to produce them. To be able to produce them, CO-Toolkits should be well-organized and designed.

(5) The need for checking tools that can be used to check existing software systems against a certain set of rules and guidelines and then report problems back to developers. We think that formalizing rules can make it easier to build such checking tools.

1.3 Related Work

1.3.1 The “Automatic Detection of Design Problems” Work

1.3.1.1 What Is The “Automatic Detection of Design Problems”?  

This work is being performed at the FZI Forschungszentrum Informatik and it has been introduced to us around the time of completing this thesis. All details provided in this section are based on a paper [14] by Dr. Ciupke from the above mentioned organization.

This work has been focused on analyzing legacy code (i.e. non object-oriented code), simplifying frequent design problems as queries and locating occurrences of these
problems in a model derived from source code. A tool set has been implemented to support these tasks. The following are some of the criteria that have been considered in this work:

(1) Methods and tools work on the design level, and the quality of the overall structure of a system, determines the flexibility of that system.

(2) All of the required information is extracted from the source code of the input system.

(3) To cope with large amounts of code, problem detections should run automatically.

(4) A problem detection method should point us directly to the problems we are searching for.

(5) The approach should be language independent.

(6) An approach to problem detection in reengineering should cover a wide range of possible problems.

1.3.1.2 Our Work vs. The “Automatic Detection of Design Problems”

Work

"Automatic Detection of Design Problems" approach is relevant to our work in the sense that:

(1) Similar to their approach, our approach is based on identifying a well-defined set of problems that can be detected based on a set of well-defined rules.

(2) Similar to their tool set, our prototypes can be used to check source code against pre-defined set of rules and guidelines. Any violations to these rules will be reported back to developers.

(3) Similar to their methods, our methods (i.e. rules) are language-independent.
(4) The main difference between the "Automatic Detection of Design Problems" (i.e. ADDP for short) and our research, is that the ADDP is focused on legacy software systems, while our approach has been focused on Component–Oriented Toolkits.

1.3.2 Rigi

1.3.2.1 What is Rigi?

As described in [57], Rigi is a system used to understand large information spaces like software programs. Rigi can model the input system by extracting artifacts from the information space using a language–dependant parser (e.g. cparse for C programming language), organizing them into higher level abstractions, and then visualize them using a graph editor called rigiedit.

The rigiedit is the user interface that is used to browse, analyze, and modify a graph that models the input system. The rigiedit can be used to simplify the graph by clustering related artifacts into subsystems. This clustering is mainly based on the components functions.

The rigiedit has built–in operations to assist in program understanding. It can be used to select and group artifacts based on certain modularity principles such as data abstraction within subsystems. Various statistical reports can help with maintenance or re–engineering tasks. Also, rigiedit is programmable using a script language called Tcl [35].

1.3.2.2 Our Prototypes vs. Rigi

Rigi is mainly a visualizing tool that can be used to visualize artifacts that are extracted from code using other parsing tools.
Similar to Rigi, our prototypes are dependent on appropriate parsers that can be used to extract the required artifacts (i.e. #include statements, inheritance, association relations, etc.) from the source code. What makes our prototypes different from Rigi is that they are checking engines that check code against certain set of formalized rules and guidelines (c.f. chapters 4 and 5). These formalized rules are based on the initial assumptions for good design practice that we made in chapter 3.

1.3.3 Together/C++

1.3.3.1 What is Together/C++?

Object International [NO TAG, 34] introduced the Together/C++ visual development tool. Source code (i.e. C++ code) can be developed (or imported in) and visualized using the Together/C++ tool. Unlike Rigi, Together/C++ tool uses its own embedded parsing utility and keeps its own internal repositories to hold the artifacts.

1.3.3.2 Our Prototypes vs. Together/C++

Our prototypes are not development tools like Together/C++. They can be used to check the input source code against a certain set of rules and guidelines (c.f. chapters 3 and 4). These rules and guidelines can be described as an extension to programming languages in the sense that existing programming languages may permit many situations where our rules can be violated. While the Together/C++ tool permits any situation that is permitted by C++.
1.3.4 DATRIX

1.3.4.1 What is DATRIX?

DATRIX is a parsing tool developed at Bell Canada. It can be used to extract many kinds of artifacts from the source code such as the #include relation, data flow, etc. GRAPHPLACE CASE is then used to postscript these extracted artifacts and then tools like GHOSTVIEW can be used to visualize them [8, 9].

1.3.4.2 Our Prototypes vs. DATRIX

DATRIX can be related to our work in the sense that it can be modified such that it can then be used to extract the artifacts that are required by our prototypes to conduct their checking analysis.

1.3.5 Other Parsing Tools

There are many parsing tools available from both the academic institutions and the industrial organizations (i.e. software development houses). We can group parsing tools into two main groups:

(1) Programming language-specific parsers: This group contains any parser that is used by a language-specific compiler to parse in source code that needs to be compiled. Also, it contains any parser that is used by a language-specific linker to parse in object code that needs to be bound.

(2) Task-specific parsers: This group contains any parser that is used to extract particular set of artifacts from its input source.
1.3.5.1 Our Prototypes vs. Parsing–Tools

Given that our prototypes check for a particular set of rules and guidelines that we introduced in this research, only the language task–specific group of parsing tools can be complementary to our prototypes.

A task–specific parser can be integrated into our prototypes if that parser can extract the set of artifacts from the given language–specific source code such as the "#include" statements, the inheritance and association relations, objects (and their types) etc.

1.4 Our Audience

This research is intended for the following people:

(1) Developers of CO–Toolkits:

i— We think that following our guidelines in documenting, designing and organizing CO–Toolkits will make it easier to maintain such toolkits.

ii— Developers can use tools that are developed based on our views to check their CO–Toolkits’ code.

(2) Maintainers and technical support staff of CO–Toolkits:

i— If developers provide precise documents with their CO–Toolkits, maintainers will be able to find the parts of their systems’ code that need to be modified in order to fix bugs or change some features.

ii— However, it is the maintainers obligation to keep such documents always up–to–date, otherwise, such documents will loose their preciseness and correctness.

(3) CO–applications developers (i.e. “Users” of CO–Toolkits):
i– Users can use our documents to determine if the given CO–Toolkit has particular components that can provide their applications with certain services.

(4) Software testers/evaluators of CO–Toolkits and their applications can use our documents to understand the components they are dealing with.

(5) Given that a Component Guide Document (i.e. CGD for short) that we propose in chapter 6 can contain graphical pictures about its system, sales staff can use some of these pictures for their advertisements.

1.5 The Primary Common Characteristics of CO–Toolkits

The common characteristics of CO–Toolkits that are the focus of our research, could be summarized as follows:

(1) A CO–Toolkit is a software system that is composed of a set of frameworks such that each framework is composed of a set of components.

(2) A CO–Toolkit is developed using an object–oriented programming language such as C++, etc. It may be possible to develop CO–Toolkits using non Object Oriented Languages, however, the outcome of this research can not be used for them.

(3) Integration and Composability: A component must be able to integrate with one or more other components from the same system to build a new application.

(4) Basic concepts of software engineering such as information hiding, design for change, expandability, etc.[21] must be maintained in such CO–Toolkits:

i– Information hiding and encapsulation (i.e. separating the interface from the implementation): The interface is a contract that specifies a component’s
functionality. The interface should not expose any implementation or platform or environment dependencies.

ii— Expandability of CO–Toolkits means that developers and maintainers of such toolkits can add new components to any framework without having to change any of its existing components. Also, they should be able to add new frameworks without having to change any of the existing ones.

1.6 Key Contributions Made By This Research

The main contributions made by this research can be summarized as follows:

(1) We have shown how to formalize rules for good practice concerning file structures (c.f. chapter 4), design and implementation (c.f. chapter 5), and documentation (c.f. chapter 6) of CO–Toolkits. This is a contribution because formalization of rules is an important step towards building tools. As explained next, we found it easy to build our prototypes based on the formalized rules made in this thesis.

(2) We have shown that such formalized rules can be used to develop tools. In this thesis, we provided prototype systems (c.f. optional reading material is provided in appendixes A and B) that we developed based on these formalized rules (i.e. views).

(3) We have shown that such prototypes can be used to check software systems and report any rules violations back to developers (c.f. Optional reading material is provided in appendixes A2–A4, B2–B4).

1.7 Thesis Organization

This thesis has eight chapters, and three appendixes:
Chapter 1 is this introduction chapter.

Chapter 2 describes the case study of this research (i.e. ACE).

Chapter 3 describes the set of assumptions for good design practice that we made based on our experience with ACE as well as based on our own work experience. Based on these assumptions, we derived and formalized rules as described in the three views of this research (c.f. chapters 4 through 6).

Chapter 4 describes our view for CO–Toolkits’ file structures.

Chapter 5 describes our view for CO–Toolkits’ design and implementation structures.

Chapter 6 describes our view for CO–Toolkits’ documentation.

Chapter 7 shows how to check three views to be sure that they are consistent.

Chapter 8 describes the limitation of our views, conclusions, and future work.

Appendix A describes the prototype that we built based on the view for CO–Toolkits’ file structure of chapter 4. Appendix A1 describes how to use this prototype as well as describes and presents its source code. Appendix A2 presents the first artificial example that has been used to demonstrate the kind of file structures that can be passed by the prototype. Appendix A3 presents the second artificial example that has been used to demonstrate the kind of file structures that will be failed by the prototype, and to demonstrate the kind of violation messages that are reported back to developers. Appendix A4 presents a third example that is based on a sample ACE file structure.

Appendix B describes the prototype that we built based on the view for CO–Toolkits’ implementation structures of chapter 5. Appendix B1 describes how to use this prototype as well as describes and presents its source code. Appendix B2 pres-
 CHAPTER 1: INTRODUCTION

ts the first artificial example that has been used to demonstrate the kind of design and implementation structures that can be passed by the prototype. Appendix B3 presents the second artificial example that has been used to demonstrate the kind of design and implementation structures that will be failed by the prototype, and to demonstrate the kind of violation messages that are reported back to developers. Appendix B4 presents a third example that is based on a sample ACE inheritance structure.

- Appendix C provides a set of sample tables for the view for CO-Toolkits' documentation of chapter 6.
CHAPTER 2

CASE STUDY: STRUCTURE AND ANALYSIS

2.1 Overview

This chapter describes the “ADAPTIVE Communication Environment” (i.e. ACE for short) that has been used as a starter example for this research. The ACE version used for this research goes back to April 1996. At that time ACE was suggested as a good example to look at by people from the industry at a TRIO meeting. Improvements may have been made on ACE since that time, however, this research was not intended to keep up with such improvements. As explained in the next chapter, after looking at ACE, we looked back at the software engineering principles and then derived a set of assumptions for good design practice.

We think that a less coupled software design (i.e. designing software for change and mastering complexity) would reduce the learning curve in any software area. However, someone may argue that this can be true in some areas such as GUI frameworks, but not necessarily true in some other areas such as telecommunications, medical imaging, or real–time embedded systems where efficiency, predictability, and scalability are key forces. We think that “designing software for change and mastering complexity” needs to be always a key force in order to have systems that can be maintained at reason-
able costs. It may be hard to achieve a software system that is designed for a high degree of performance and a high degree of simplicity at the same time. However, with the “designing software for change and mastering complexity” as the key force, software systems can be constructed or reconstructed in away such that performance implications can be minimized. For example, providing precise and correct specifications for the design and the interface of a CO-Toolkit can help developers in choosing the right components for their applications.

This chapter has six sections. The next section describes ACE. In the third section, we discuss the need for studying CO-Toolkits’ file structures, and present samples of the file structures that we derived from ACE frameworks. In the fourth section, we discuss the need for studying the inheritance and association relations among CO-Toolkits’ components, and present samples of these two relations as used among ACE components. The fifth section describes the kinds of documents that we found with ACE and the need for more documents. The last section presents the conclusions of this chapter.

2.2 Case Study: ADAPTIVE Communication Environment

This section describes ACE according to Schmidt in his papers on ACE [43–52].

2.2.1 What Is The “ADAPTIVE Communication Environment”?

The ADAPTIVE Communication Environment is a freely available object-oriented toolkit that provides a rich set of reusable components that can be used to perform common communication software tasks across a range of platforms. The communication software tasks provided by ACE include event demultiplexing and event handler dis-
patching, signal handling, service initialization, inter-process communication, shared memory management, message routing, dynamic configuration of distributed services, concurrent execution and synchronization. Many benefits can be accomplished by using ACE. Examples are: (i) increased portability, (ii) increased software quality, and (iii) increased efficiency and predictability.

The "ACE OS Adapter Layer" resides directly above the Operating System (i.e. OS for short) APIs (i.e. Application Programming Interfaces) written in C. It provides platform-specific components that are associated with the following OS APIs: (i) concurrency and synchronization, (ii) inter-process communication (i.e. IPC for short) and shared memory, (iii) event demultiplexing mechanisms, (iv) explicit dynamic linking, and (v) file system mechanisms.

The "ACE C++ Wrapper for OS Interfaces" simplifies application development by providing C++ interfaces that encapsulate the OS concurrency, communication, memory management, event demultiplexing, and others. Applications can use from these wrappers (i.e. programs) by, for example, inheriting and/or instantiating the following components: (i) concurrency and synchronization components, (ii) IPC and file system components, and (iii) memory management components.

The "ACE Frameworks Layer" is a higher-level programming layer that integrates the lower-level C++ wrapper (i.e. programs). It contains the following: (i) Event Demultiplexing components, (ii) Service initialization components, (iii) service configuration components, and (iv) hierarchically-layered stream components.
2.2.2 Why Did We Choose ACE As A Case Study?

We have selected ACE as a case study for the following reasons:

(1) It was presented by industry to the Telecommunication Research Institute of Ontario (i.e. TRIO for short) as an example of something that they could use.

(2) ACE provides a rich set of components (over 300 classes categorized into several frameworks). Thus, it would serve as a good case study for this research.

(3) It has been developed to meet the high demand for robust and high-performance distributed communication software systems such as global communication systems, network management platforms, etc.

(4) It has been used to build real applications in the fields just mentioned before.

(5) Its code is available for free and the author (Dr. D. Schmidt) has agreed to help.

2.2.3 Are Our Views ACE–Dependent?

As we think that ACE is a good representative of CO–Toolkits, our mathematical views (c.f. chapters 4–6) are intended to be general such that they can be used for any CO–Toolkit that satisfies the characteristics stated in section 1.5 of chapter 1.

2.3 File Structures

2.3.1 Why Do We Need To Study File Structures?

Software developers reuse files among the components of their systems without paying enough attention to the growing dependency among these files. As a result, the "#include" relation among files (i.e. file dependencies) becomes complex and unorganized [9]. This has encouraged us to investigate how CO–Toolkits’ file structures should
be organized where the "designing software for change and for mastering complexity" is the key force.

2.3.2 Visualizing ACE File Structure

2.3.2.1 ACE File Structures Graphs

This section describes snapshots of the existing ACE file structure. We used DATRIX and GRAPHPLACE CASE tools [8] to extract and visualize the "#include" statements from ACE files as shown in Figures 2–1 through 2–6.

As ACE has been used as a research subject, the following examples from ACE are presented in this section for the purpose of demonstrating our ideas. For example, pointing to the bi-directional inclusion relation between the ACE.h and ACE.i files (i.e. in Figure 2–4) is intended to explain what we mean by bi-directional inclusion between the intermediate and the internal files of the same component.

Figure 2–1 shows the file structure of all ACE components is not easy to understand. This would be the result of using an add-hoc approach during the development process of software systems. When it comes to adding new classes to an existing system, developers try to reuse as much of the existing entities (variables, classes, etc.) as possible. They simply "#include" the files that contain these entities without watching the growth of their software systems' file structures.
FIGURE 2-1: Overall ACE File structure
Figure 2–2 shows the file structure of the REACTOR framework of ACE. This graph comes with four extension nodes that are illustrated in Figures 2–3 through 2–6. Each extension node is intended to represent a pattern sub-graph that is used in several places (i.e. OS.h). As shown in Figures 2–2 through 2–4, each extension node is denoted by "+++++included-file+++++".

Figure 2–2 shows the following multiple path case: Event_Handler_T.cpp "#include"s both Event_Handler_T.h and Event_Handler_T.i, and Event_Handler_T.h "#include"s Event_Handler_T.i. It is clear that Event_Handler_T.cpp has two possible paths to "#include" Event_Handler_T.i, either directly or through including Event_Handler_T.h. Many other multiple-path cases can be noticed in Figures 2–2 and 2–3.

Figure 2–4 shows a sequence of inclusions in which Log_Msg.h header file "#include"s Log_Record.h, Log_Record.h "#include"s ACE.h, then ACE.h "#include"s ACE.i, which in turn "#include"s Log_Msg.h forming a loop. In this graph we notice two things: having loops, and having a .i file "#include"ing a .h file.

2.3.2.2 What Issues Do These Figures Raise?

Figures 2–2 through 2–6 raise issues such as having inclusion loops, and inclusion multiple paths in file structures. We understand that a language like C++ [26] can handle these issues through special checking statements that precede the "#include" ones. Such checking statements ignore any "#include" statement if the file to be "#include"d has been "#include"d before in the current sequence of inclusions.
FIGURE 2-2: REACTOR Framework File Structure
FIGURE 2-3: Synch.h File
Having a software system to pass compilation and then run does not necessarily mean the system has been well-organized. A File of a well-organized system need to “#include” other files only when there is a need to do so.

2.3.2.2.1 Inclusion Loops

We define an inclusion loop as a sequence of files such that each file “#include”s its next in the sequence, and the last file “#include”s the first one in the same sequence. In other words, an inclusion loop could be viewed as a ring of file inclusion operations.

As a special case of inclusion loops, it is interesting to mention that each of the bi-directional inclusions in ACE exists between a pair of files that share the same filename with different extensions (i.e. one with .h and the other with .i extension.). Given that an internal file is intended to contain its component’s internal data structure and the top file is intended to host the user interface, we think that such bi-directional inclusions violate the software design concept of information hiding [36–40].

2.3.2.2.2 Inclusion Multiple Paths

We mean by multiple-paths that a file could “#include” another file through more than one possible sequence of inclusions. For example; let file A “#include”s files B and C, file B “#include”s files D and E, and file C “#include”s files E and F. In this example, file A “#include”s file E through two possible sequences; <A, B, E> or <A, C, E>.
CHAPTER 2: CASE STUDY: STRUCTURE AND ANALYSIS

FIGURE 2-4: Log_Msg.h File

FIGURE 2-5: Time_Value.h File
2.4 Implementation

2.4.1 Inheritance

2.4.1.1 What Is Inheritance?

Inheritance is a relationship among components (classes) in the same system. In object-oriented technology, a class is the basic construct that is used to build programs. A class is called a parent class (also called super class, base class, ancestor class) of one or more other children classes (also called sub-classes, derived, inheriting classes) if it is inherited by these children classes [19, 26]. A parent class has two possible types of clients: sub-classes and objects (i.e. instantiations) [55]. A class has three possible sections: public, protected, and private. General speaking, inheritance is intended to facilitate programming by extension rather than programming by re-invention [6, 13, 15, 17, 18]. In chapter 5, we will show how this relation should be used among components.

Some programming languages such as Java [11–11, 16], support only single inheritance type. While other languages such as C++, support both single and multiple inheritance types. In the case of single inheritance, a class inherits from only one class (i.e. the inheriting class has only a single parent). While in the case of multiple inheritance, a class inherits from two or more other classes (i.e. the inheriting class has multi-parents) [6, 13]. Inheritance can be safe if it reveals only the behavior of the inherited classes to the inheriting ones. In other words, inheritance can be safe if the basic software design concept of information hiding is kept maintained. For further detail on inheritance, the following references are recommended [5, 13–16, 18–19, 22, 30–31, 39–40, 44, 64–66].

2.4.1.2 Does Multiple Inheritance Cause Any Additional Difficulties?

With single inheritance, any name can be simply resolved to a class member by simply accessing the first name encountered for data members and by accessing the first
signature match encountered for methods. Since several parents can declare a member within a multiple inheritance hierarchy, it becomes hard to decide which one to choose from these declarations. As a result, multiple inheritance causes additional difficulties to object-oriented programming [24].

2.4.1.3 Why Do We Need To Study Inheritance In CO-Toolkits?

Inheritance is used in almost every Object-Oriented-program. It is intended to enable developers to reuse existing components instead of completely writing their programs from the scratch. However, inheritance is not always safe. Regardless of the selected programming language, every time inheritance is used, developers should be concerned about the degree of safety that their programs will maintain [55, 56]. In this research, we want to study inheritance in order to show how to formalize rules for safe usage of inheritance among CO-Toolkits' components.

2.4.1.4 ACE Inheritance

We have used Together/C++ [34] to extract and visualize inheritance relation that exist among some of ACE components. Given that a CO-Toolkit is composed of a set of frameworks that are intended to be independent from each other, we decided to look at inheritance relations that exist in two ACE frameworks: the REACTOR, and the MEMORY (c.f. Figures 2–7, and 2–8).

Figure 2–7 shows the inheritance relation that exists among the REACTOR framework's components. The component called ACE_Event_Handler is the parent of five other components (i.e. ACE_Proactor, ACE_Sig_Adapter, ACE_ReactorNotify, ACE_ReactorNotify, and ACE_Event_Handler_T). Consequently, the component called ACE_Sig_Handler is the parent of the ACE_Sig Handlers component. It is important to mention at this point that the REACTOR framework contains
many other components that are not shown in Figure 2–7 simply because they are not involved in inheritance.

![Inheritance in the REACTOR framework](image)

**FIGURE 2–7**: Inheritance in the REACTOR framework

![Inheritance in the MEMORY framework](image)

**FIGURE 2–8**: Inheritance in the MEMORY framework

Figure 2–8 shows the inheritance relation among the MEMORY framework’s components. In this Figure, the ACE_Event_Handler component (i.e. the shaded one) belongs to a different framework (i.e. the REACTOR framework).

### 2.4.2 Association

#### 2.4.2.1 What Is Association?

Association is a relation between two components in the same system. We consider a pair of components to be associated with each other if at least one of them contains (or has a reference to) an object that is instantiated from the other component [6, 29]. Association can be uni or bi-directional.
2.4.2.2 Why Do We Need To Study Association In CO-Toolkits?

Association is a relation that is common in almost every program, therefore it is important to show how to formalize rules for using association such that CO-Toolkits structures remain simple.

2.4.2.3 ACE Association

We have used Together/C++ [34] to extract and visualize the association relations that exist in the REACTOR and MEMORY frameworks of ACE. These relations are illustrated in Figures 2–9 and 2–10 respectively.

In Figure 2–9, we can notice the following: (a) Several classes associate with themselves. (b) A class could associate with another class or with a struct (i.e. a C++ construct). (c) Only uni-directional association appears. (d) In one case, a class associates with a stranger class that does not belong to the same framework. (e) A class may associate with several other classes (i.e. multiple association). Similar observations can be seen in Figure 2–10 as well.

Association can be a uni-directional or bi-directional relation. For example, when component A contains an object from component B, but B does not have any objects of A, then we can say that component A has a uni-directional association relation with component B. If both components contain object(s) from each other, then the two components have a bi-directional association relation with each other.

In order to keep the frameworks independent from each other, association (i.e. similar to inheritance) should not be permitted among components from different frameworks. Within the same framework, association among components can be either uni-directional or bi-directional. However, association between components from frame-
works and classes from shared pools should be uni-directional since that we do not want a shared class to depend on any component.

FIGURE 2-9: Association in the REACTOR framework

FIGURE 2-10: Association in the MEMORY framework
2.5 Documentation

2.5.1 What Documents Did We Find With ACE?

Besides the code of ACE, we found a set of papers. One of them [43] provides an overview of ACE, while each of the others describes a part ACE [44–52]. When we started this research, we found it much easier to look at ACE through its code than through its papers. We think that ACE papers were written for professionals who are already familiar with ACE. Also, we found a set of manuals that basically tell what file to "#include" for a given component.

2.5.2 Are Such Documents Enough From The User Point Of View?

We believe that the answer for such question is NO. Our experience and the experience of others [9] show that toolkits like ACE are not something easy to use. In most cases, this would encourage developers to avoid the difficulties of using CO–Toolkits by simply not using them.

ACE was presented by people from the industry as a good tool that they could use. However, it became clear (i.e. only by use) that it was not good enough for widespread use.

2.6 Conclusions

In this chapter, we have:

(1) Described ACE, the case study that is considered for this research.

(2) Discussed the need for:
i— Studying CO–Toolkits' file structures (c.f. section 2.3.1).

ii— Studying the inheritance relation among Components (c.f. section 2.4.1.3).

iii— Studying the association relation among Components (c.f. section 2.4.2.2).

(3) Presented the:

i— File structure that is used in ACE (c.f. section 2.3.2).

ii— Inheritance relations that are used in some of ACE frameworks (c.f. section 2.4.1.4).

iii— Association relations that are used in some of ACE frameworks (c.f. section 2.4.2.3).

iv— ACE documents (c.f. section 2.5)
CHAPTER 3

ASSUMPTIONS FOR GOOD DESIGN

3.1 Overview

Having looked at ACE as an example to start this research with, the next step has been to review the software engineering principles and derive a set of realistic assumptions for good design practice.

This chapter describes the main software engineering principles (i.e. Ghezzi et al text book [21], Parnas et al papers [36, 37, 38]) that we used to derive our assumptions. These assumptions are described in this chapter as well (c.f. section 3.3). Throughout this thesis, we used these assumptions to derive and formalize rules for: (i) organizing CO–Toolkits' file structures (c.f. chapter 4), (ii) implementing CO–Toolkits' components (c.f. chapter 5), and (iii) documenting CO–Toolkits (c.f. chapter 6).

This chapter has four sections. The next section describes the main software design principles that we used in this research to make the assumptions for good design practice. These assumptions are described in the third section. The fourth section concludes what has been done in this chapter.
3.2 Software Engineering Principles

This section describes the main software design principles that have been considered when making the initial assumptions for good design (c.f. section 3.3). Separation of concerns (c.f. section 3.2.1) can be seen as the primary software design principle. It can be achieved by employing principles such as modularity (c.f. section 3.2.2) and abstraction (c.f. section 3.2.1.3) with information hiding (c.f. section 3.2.1.2) being the guiding principle when identifying modules and designing interfaces. The primary goal behind having such design principles is to master the software complexity by separating concerns.

As a result of mastering software complexity, it becomes easier to make changes because each concern can be changed by changing only one part of the system (c.f. section 3.2.4). This gives another view of information hiding and abstraction. We hide what is likely to change, and the abstraction contains what we do not expect to change.

3.2.1 Separation of Concerns

3.2.1.1 Overview

This principle can be seen as the primary principle for software design. By separating concerns, we can concentrate on each concern without concerning about the details of other perspectives [21]. For example, a software system can be evaluated from separate perspectives such as: (i) how it has been documented, (ii) how its file structure has been organized, (iii) how it has been implemented, (iv) what functions it offers, (v) how reliable it is, (vi) how efficient it is, (vii) how correct it is, (viii) how friendly its user interface is, etc.. In this research, CO-Toolkits have been viewed from three separate per-
3.2.1.2 Information Hiding

Information hiding is another view of the “separation of concerns” principle. Concerns are separated by hiding design details of each software module from the designers of other modules. Another view of information hiding is that we hide what is likely to change. Also, “information hiding” can be seen as a means to separate the user interface of a given module from its implementation details. For more details on “information hiding”, readers can refer to Parnas et al works [36, 37]. In this research, we have enforced the data structure implementation of any component to be provided in the internal file of that component (i.e. as in chapter 4’s view for CO-Toolkits’ file structures). Also, we have enforced the public section (i.e. as per the C++ terminology) of any internal class (i.e. of any component) to contain only the interface descriptions (i.e. as in chapter 5’s view for CO-Toolkits’ design and implementation).

3.2.1.3 Abstraction

Abstraction is also, another view of “separation of concerns”. Abstraction is the way to communicate what can not and should not be hidden. Modules must communicate, therefore, we abstract from what is hidden and what remains is called “the abstraction”. Another view of abstraction is that the abstraction of a system contains what is not expected to change.

3.2.2 Modularity

The general purpose of this principle is to reduce what we have to think about at any one time. Modularity can help us to compose systems from smaller modules. Also,
it can help us to decompose systems into smaller modules (parts) such that it becomes easier to understand each module by itself [36]. The “main benefit of this principle is that it allows the principle of separation of concerns to be applied in two phases: when dealing with the details of each module in isolation (and ignoring details of other modules) and when dealing with the overall characteristics of all modules and their relationships in order to integrate them into a coherent system” [21]. In this research, we have viewed a CO-Toolkit as a set of independent frameworks such that each of them is composed of a set of collaborating components.

3.2.3 Formality

This principle requires the software process to be driven and evaluated by mathematical laws [21]. Providing formalized and correct specification for software design and interface (i.e. as in Parnas et al [39, 40]) can help developers to produce systems that can function as expected. Formality can be seen as a means to achieve other ends such as precision, compatibility, ease of use, etc.. In this research, we have formalized the rules and guidelines for constructing CO-Toolkits' file structures (c.f. chapter 4), implementing CO-Toolkits (c.f. chapter 5), and documenting Co-Toolkits (c.f. chapter 6).

3.2.4 Mastering Complexity And Design For Change

Mastering complexity in software systems is the primary goal of applying the "separation of concerns" principle. By separating concerns, it becomes easier to handle each concern by itself without concerning about the details of other concerns. As a result, it becomes easier to change any concern by only changing one part of the system without affecting the system's other parts [21, 38]. In this research, we have provided formalized rules (c.f. chapters 4, 5, and 6) based on which, we have built our prototypes (c.f. appen-
3.3 Assumptions For Good Software Design Practice

This section describes the set of assumptions that we made based on the software engineering principles described in the previous section as well as based on our experience with ACE as explained in the previous chapter.

We think that these assumptions are good for design practice. However, we do not claim that they are firm ones that should be used "as is" by developers.

In [28], Lakos provides principles and rules for how to construct components. These principles discuss issues that are mainly arbitrary and C++ language-dependent. As our assumptions and rules are intended to be general, and as this research is intended to show how to derive and formalize rules, not to derive language-dependent ones, we think that discussing Lakos rules is out of the scope of this research. However, showing how to formalize Lakos rules (i.e. as indicated in the "Future Work" section of chapter 8) can be a good research opportunity in the future.

3.3.1 General Assumptions

This section describes the set of initial assumptions that were considered in the three views (c.f. chapters 4, 5, and 6).

As it is important to consider the software engineering principles in the development process, we think that the following concerns need to be considered when organizing, designing, and documenting CO-Toolkits: (i) the file structures used for the implementation of CO-Toolkits, (ii) the clustering of CO-Toolkits' components into
frameworks, (iii) the dependencies among frameworks inside the same CO-Toolkit, and
(iv) the internal design of components and the dependencies among them.

As we are studying software systems that are complicated and hard to understand
because of the many existing relations among their components, we need to use existing
tools in order to abstract and visualize these relations.

As per the Formality principle (c.f. section 3.2.3), having formalized and precise
rules in-place before developing checking tools can help in speeding up the development
process. Checking tools can be used then to check systems and report problems back to
developers.

A CO-Toolkit can be composed of a set of frameworks and a set of shared pools
such that: (i) each framework can be composed of a set of components, and (ii) each
shared pool can be composed of a set of classes and other entities that are shared by a
number of the frameworks. A framework can share (i.e. have access to) more than one
shared pool. This has been based on: (i) the modularity principle (c.f. section 3.2.2) in
the sense that a CO-Toolkit has a modular structure, (ii) the separation of concerns prin-
ciple (c.f. section 3.2.1) in the sense that the shared elements are separated from the
frameworks, and (iii) our experience with ACE in the sense that ACE is composed of
a set of frameworks.

3.3.2 Assumptions For Good File Structures

This section describes the assumptions that are considered for the CO-Toolkits'
file structures view (c.f. chapter 4).

A CO-Toolkit's file structure can be simplified by grouping the files that are
shared among the CO-Toolkit's frameworks into a set of shared pools (c.f. section 3.2.1:
separation of concerns). This is to minimize duplication of files and to eliminate dependencies among frameworks such that it becomes possible to study, analyze, and use each of these frameworks by itself (c.f. section 3.2.4: mastering complexity and design for change).

A number of files can be used to implement a component. As indicated in the overview section of chapter 1, we use three files for each component; top, intermediate, and internal. The top file is to hold the component’s main part, the intermediate file is to hold the component’s interface, and the internal file is to hold the implementation of that component. This has been based on our experience with ACE where up to three files are used to implement each component.

In order to keep frameworks independent from each other, files can not exist at the same time in more than one framework. Shared–pools are proposed in this thesis to hold any files that need to be shared by frameworks. This has been based on: (i) the separation of concerns and modularity principles (c.f. sections 3.2.1, and 3.2.2 respectively) in the sense that we need to be able to decompose a CO–Toolkit into smaller modules (i.e. frameworks in this case) that can be studied separately, (ii) the objective that we need to master the complexity of our systems (c.f. section 3.2.4) such that it becomes possible to make changes in one framework without affecting the others.

As the “designing software for change and mastering its complexity” is the key force in this research, we think that it is not a good practice to permit either conditional (i.e. a header file is to be ignored if it has been already included in the same inclusion sequence) or unconditional (i.e. as opposed to conditional) loops in file inclusion. Such inclusion loops can make it hard to track down the code parts that need to be changed.
Concerning the use of top files, we assumed that it is a good practice to satisfy the following:

1. A top file should not be included by any other file. This is to keep file structures simple and easy to understand,

2. A top file can include only its intermediate file. This is to help maintain design principles such as information hiding (c.f. section 3.2.1.2), and

3. Only top files can "include" shared files from shared-pools. This is to minimize the external effect on components.

Concerning the use of intermediate files, we assumed that it is a good practice to enable an intermediate file to include intermediate files from other components within the same framework. This is to enable components collaboration as learned from ACE.

Concerning the use of internal files, we assumed that it is a good practice to satisfy the following:

1. An internal file can not include any other file. This is to keep the component’s data structure in one file (c.f. section 3.2.1.2: information hiding), and

2. An internal file can not be included by any other file except its intermediate file. This is to ensure that a component’s data structure can be accessed by its functions only.

Concerning the use of files from the shared pools, we assumed that it is a good practice to satisfy the following:

1. enable inclusion among file within the same shared pool. This is to keep shared pools independent from each other (c.f. section 3.2.1: separation of concerns, and section 3.2.2: modularity),
(2) have each shared-pool be accessible by a sub-set of the frameworks. This is to maintain organized and simple file structures, and

(3) have any library file that is provided by the platform system to set in the main shared-pool that is shared by all frameworks. This is to enable the usability of system libraries by all frameworks.

3.3.3 Assumptions For Good Implementation Structures

This section describes the initial assumptions that are considered for the view on CO-Toolkits’ design and implementation (c.f. chapter 5).

As defined in the overview section of chapter 1, a component can be implemented using a main part and a set of internal classes. This is to make it possible to either use each component as a stand-alone application, or invoke its internal classes by other components.

We assumed that the implementation of CO-Toolkits needs to address the following: (i) their framewoarking scheme, (ii) the internal design of their components, and (iii) the dependencies (i.e. relations) among their components. This has been based mainly on the: (i) abstraction principle (c.f. section 3.2.1.3) in the sense that a CO-Toolkit can be seen (i.e. abstracted) as a set of frameworks and also a framework can be seen as a set of components, (ii) modularity principle (c.f. section 3.2.2) in the sense that a CO-Toolkit has a modular framework structure, and (iii) objective of mastering complexity and design for change (c.f. section 3.2.4) in the sense that we want to make it easy to understand and use CO-Toolkits.

Concerning the use of the inheritance relation among components, we assumed that it is a good practice to satisfy the following:
(1) Components from different frameworks can not inherit from each other. This is to keep frameworks independent from each other (c.f. section: 3.2.2 modularity, and section: 3.2.4 mastering complexity and design for change).

(2) A component can inherit from only one other component from the same framework. This is to maintain simple inheritance structures (c.f. section: 3.2.4 mastering complexity and design for change).

(3) Only public inheritance can exist among components. This is a limitation that we propose on the use of inheritance among components. This limitation helps in maintaining a safe use of inheritance. Protected and private inheritance can reveal internal structures of components and that violates the information hiding principle (c.f. section 3.2.1.2).

(4) A component's internal class can inherit from only one other internal class from the same component. This is another limitation that we propose on the use of inheritance in order to maintain simple inheritance structures (c.f. section: 3.2.4 mastering complexity and design for change).

(5) Inheritance can not exist among shared classes from different shared pools. This is to keep shared-pools independent from each other. This is mainly based on the separation of concerns, modularity, and design for change principles.

(6) A shared class can inherit from only one other shared class from the same shared pool. This is to maintain only single inheritance relations among shared classes (c.f. section: 3.2.4 mastering complexity and design for change).
Concerning the use of the association relation among components, we assumed that it is a good practice to satisfy the following:

(1) Association can not exist among components from different frameworks. This is to keep frameworks independent from each other. As with inheritance, this assumption is mainly based on the modularity, separation of concerns, and design for change principles.

(2) Association and inheritance relations can not exist at the same time among a component’s two internal classes. This is for convenience in order to maintain simple inheritance and association structures.

3.3.4 Assumptions For Good Documentation

This section describes the assumptions that are considered for the view on CO–Toolkits’ Documentation (c.f. chapter 6). Since developers of CO–Toolkits often do not provide proper documentation for their systems, it becomes hard to understand and use these systems. Reusability of components is intended to simplify and enhance the process of software development. However, it often takes developers a long time to understand the CO–Toolkits they want to use. As a result of not having proper and formal documentation, we think that using existing components from CO–Toolkits often slows down the process of development rather than speeding it up.

We assumed that the “table based Documentation methods” [7, 36–40] can be used to document CO–Toolkits. This is based on the fact that these methods have been in use during the last 23 years and different people from different countries have used them. Examples are the A7 project [7], and the work by Parnas et al [36–40]. Such documentation methods require a lot of time and effort to be invested up–front. However, if automation tools such as the Table Tool System (i.e. TTS for short) [1] are used to pro-
duce such formal document, the up-front investment can be substantially reduced. Also, using such documentation methods can reduce the cost of developing and maintaining systems code. Our experience with building the prototypes for this research has shown that having formal specifications up-front can make it easy to develop the system’s code. Investigating the effectiveness of these documentation methods is out of the scope of this research.

3.4 Conclusions

In this chapter, we have:

(1) Discussed the main software design principles (c.f. section 3.2) that we considered when we made the initial assumptions for this research.

(2) Made the following assumptions for CO–Toolkits’ design practice:

i— General assumptions (c.f. section 3.3.1) that were considered in the three views discussed in chapters 4 through 6.

ii— Assumptions for good file structures (c.f. section 3.3.2) that were considered in the first view for CO–Toolkits’ file structures (c.f. in chapter 4).

iii— Assumptions for good design and implementation structures (c.f. section 3.3.3) that were considered in the second view for CO–Toolkits’ design and implementation (c.f. chapter 5).

iv— Assumptions for good CO–Toolkits’ documentation practice (c.f. section 3.3.4) that were considered in the third view for CO–Toolkits’ documentation (c.f. chapter 6).
CHAPTER 4

FIRST VIEW: FORMALIZING RULES FOR ORGANIZING FILE STRUCTURES OF CO–TOOLKITS

4.1 Overview

The next section describes in detail our view for CO–Toolkits’ file structures. This view is based on the assumptions for good design practice that we made in chapter 3 (c.f. sections 3.3 and 3.3.2). Section 2.3.1 of chapter 2 explains why we need to study CO–Toolkits’ file structures. Appendix A describes a prototype system that we built based on this view. Two artificial examples and a third one on a sample ACE file structure were used to demonstrate this prototype. Section 8.2 of chapter 8 discusses the limitations of our views (c.f. chapters 4 through 6) including this file structures one.

4.2 Towards A Mathematical View For CO–Toolkits’ File Structures

4.2.1 Purpose

This view is intended to formalize a set of rules and guidelines for how to maintain well–organized file structures in CO–Toolkits. These rules and guidelines are based
on the initial assumptions for good design practice that we made in chapter 3 (c.f. sections 3.3 and 3.3.2). This view can be expanded by adding new rules that we are not aware of at this time.

This view can be used to develop a tool that can be used to check CO-Toolkits' file structures and report problems back to developers. To support this claim, we have built a simple prototype system (c.f. Appendix A) based on this view and demonstrated it by using two artificial examples and a third example on a sample ACE file structure.

4.2.2 The View Structure

The CO-Toolkit's file structure that we considered for this research (c.f. Figure 4-1) is simple. A CO-Toolkit is represented by two disjoint groups of files, one group is used to implement the frameworks, and the second is used to implement the shared-pools. The first group is also divided into several subsets of files such that each of them is used to implement a particular framework. Also, the second group is divided into several subsets of files such that each of them is used to implement a particular shared-pool. Consequently, each framework's set of files is divided into a number of subsets such that each of them is used to implement a particular component. Also, each component's set of files contains three files that share the same filename, but with different extensions (.top refers to the component's top file, .inm refers to the component's intermediate file, and .int refers to the component's internal file). Readers can refer to the lexicon of this view (c.f. section 4.2.4) for the meanings of the variables and symbols that appear in the various formulae throughout this section.
FIGURE 4–1: Motivating example: Model for CO–Toolkits’ file structures
In formula (4–1), the variable $CO\_Files$ is used to represent the set of all files used to implement the given CO–Toolkit. This $CO\_Files$ is represented by the union of the following two subsets of files:

1. $\bigcup_{i=1}^{F} Fr\_Files_i$: this element represents the union of the sets of files that are used to implement the frameworks. $Fr\_Files_i$ is used to represent the set of files that are used to implement the $i^{th}$ framework given that $1 \leq i \leq F$, where $F$ represents the number of frameworks.

2. $\bigcup_{k=0}^{S} Sh\_Files_k$: this element represents the union of the sets of files that are used to implement the shared–pools. $Sh\_Files_k$ is used to represent the set of files that are used to implement the $k^{th}$ shared–pool given that $0 \leq k \leq S$, where $S$ represents the number of shared–pools. We should notice that a given CO–Toolkit may have no shared–pools at all.

$$CO\_Files = \bigcup_{i=1}^{F} Fr\_Files_i \cup \bigcup_{k=0}^{S} Sh\_Files_k \tag{4–1}$$

Given that each of the $F$ frameworks is composed of a set of components, $C\_Files_i$, in (4–2) is used to represent the subset of files that are used to implement the $j^{th}$ component of the $i^{th}$ framework such that $1 \leq j \leq C_i$, where $C_i$ represents the number of components in the $i^{th}$ framework—throughout this thesis, we will call it for short, the $ij^{th}$ component. Readers should notice that a framework should contain at least one component.

$$Fr\_Files_i = \bigcup_{j=1}^{C_i} C\_Files_{ij} \tag{4–2}$$
Given that there are \( S \) shared–pools, \( Sh\_File_{ik} \) in (4–3) is used to represent the \( l^{th} \) file in the \( k^{th} \) shared–pool such that \( 0 \leq l \leq s_k \), where \( s_k \) represents the number of files in the \( k^{th} \) shared–pool.

\[
Sh\_Files_k = \bigcup_{l=1}^{s_k} Sh\_File_{lk}
\]  

(4–3)

In formula (4–4), each component is represented by its three files referred to by \( topFile_{ij} \), \( intermediateFile_{ij} \) and \( internalFile_{ij} \).

1. \( topFile_{ij} \): this element represents the top (i.e. .cpp) file of the \( ij^{th} \) component.
2. \( intermediateFile_{ij} \): this element represents the intermediate (i.e. .h) file of the \( ij^{th} \) component.
3. \( internalFile_{ij} \): this element represents the internal (i.e. .i) file of the \( ij^{th} \) component.

\[
C\_Files_i = \{topFile_{ij}, intermediateFile_{ij}, internalFile_{ij}\}
\]  

(4–4)

### 4.2.3 The View Characteristics

This sub–section describes the set of characteristics (i.e. rules) that we think should be maintained by the above view. These characteristics are based on the initial assumptions for good design that we made in chapter 3 (c.f. sections 3.3 and 3.3.2). Readers can refer to the lexicon of this view (c.f. 4.2.4) for the meanings of the variables and symbols that appear in the various formulas throughout this section.

In formula (4–5), the predicate \( GoodFileStructure() \) reflects what we mean by having well–organized file structure. This predicate is formed by taking the conjunction of the three sub–predicates that are listed next and defined in more detail in section 4.2.3.1.
(1) OnlyFourFileCategories(): this means that any file should be either a top, an intermediate, an internal, or a shared file.

(2) GoodDistinctionOffiles(): this means that frameworks and their components should have distinctive files.

(3) GoodFileInclusionRelation(): this means that the overall file "#include" relation should be well-organized.

\[
\text{GoodFileStructure()} \equiv \left\{ \begin{array}{c}
\text{OnlyFourFileCategories()} \land \\
\text{GoodDistinctionOffiles()} \land \\
\text{GoodFileInclusionRelation()}
\end{array} \right. 
\]  

(4-5)

4.2.3.1 Explanation of (4-5): GoodFileStructure()

In formula (4-6), the definition for OnlyFourKindsOfFiles(), the first sub-predicate of (4-5) above is final. Category(A) represents an auxiliary function that returns the category of file A, and that is defined in the lexicon of this view (c.f. section 4.2.4).

\[
\text{OnlyFourFileCategories()} \equiv \left\{ \forall x, (x \in \text{CO_Files}) \Rightarrow \{ \text{Category}(x) \in \{ \text{top}', \text{\'intermediate}', \text{\'internal}', \text{\'shared'} \} \} \right. 
\]  

(4-6)

In formula (4-7), the definition for GoodDistinctionOffiles(), the second sub-predicate of (4-5) above is not final. It is formed by taking the conjunction of the two sub-predicates that are listed next and defined in more detail in section 4.2.3.2.

(1) Fr_SetsOfFilesShouldBeDistinct(): this means that frameworks should have distinctive files.

(2) ComponentsMainFilesShouldBeDistinct(): this means that each component within a given framework should have its own top, intermediate, and internal files.

These two sub-predicates are explained in this chapter.

54
\textbf{CHAPTER 4: FIRST VIEW: FORMALIZING RULES FOR ORGANIZING FILE STRUCTURES OF CO-TOOLKITS}

\[ \text{GoodDistinctionOfFiles}() \equiv \left( \text{Fr\_SetsOfFilesShouldBeDistinct()} \land \right. \]
\[ \left. \text{ComponentsMainFilesShouldBeDistinct()} \right) \]  \hfill (4-7)

In formula (4-8), the definition for \text{GoodFileInclusionRelation()}, the third sub-predicate of (4-5) above is not final too. It is formed by taking the conjunction of the two sub-predicates that are listed next and defined in more detail in section 4.2.3.3.

1. \text{GoodUseOfFiles}(): this states how to make good use of each of the four possible kinds of files (top, intermediate, internal, and shared files).

2. \text{InclusionLoopsMayNotExist}(): this states that inclusion loops are not permitted in CO-Toolkits file structures.

\[ \text{GoodFileInclusionRelation}() \equiv \left( \text{GoodUseOfFiles}() \land \right. \]
\[ \left. \text{InclusionLoopsMayNotExist()} \right) \]  \hfill (4-8)

\textbf{4.2.3.2 Explanation of (4-7): GoodDistinctionOfFiles()}

In formula (4-9), the definition for \text{Fr\_SetsOfFilesShouldBeDistinct()}, the first sub-predicate of (4-7) above is final. It means that the intersection of frameworks sets of files is an empty set (i.e. frameworks have distinctive sets of files).

\[ \text{Fr\_SetsOfFilesShouldBeDistinct}() \equiv \]
\[ \left( \forall i,j, \left( \left( 1 \leq i \leq F \right) \land \right. \right. \]
\[ \left. \left. \left( 1 \leq j \leq F \right) \land \right) \right) \Rightarrow \left( \text{Fr\_Files}_i \cap \text{Fr\_Files}_j = \phi \right) \]  \hfill (4-9)

In formula (4-10), the definition for \text{ComponentsMainFilesShouldBeDistinct()}, the second sub-predicate of (4-7) above is final too. It means that the intersection of components subsets of files is an empty set too.

\[ \text{ComponentsMainFilesShouldBeDistinct}() \equiv \]
\[ \left( \forall i, \left( 1 \leq i \leq F \right) \Rightarrow \left( \forall j, \left( 1 \leq j \leq C_i \right) \Rightarrow \left( \bigcap C\_Files_j = \phi \right) \right) \right) \]  \hfill (4-10)
4.2.3.3 Explanation of (4–8): GoodUseOfFiles()

In formula (4–11), the definition for GoodUseOfFiles(), the first sub-predicate of (4–8) above is not final. It is formed by taking the conjunction of the four sub-predicates that are listed next and defined in more detail in section 4.2.3.4.

1. **GoodUseOf_top_Files():** this states how top (i.e., .cpp) files should be used.
2. **GoodUseOf_intermediate_Files():** this states how intermediate (i.e., .h) files should be used.
3. **GoodUseOf_internal_Files():** this states how internal (i.e., .i) files should be used.
4. **GoodUseOf_shared_Files():** this states how shared files should be used.

\[
\text{GoodUseOfFiles()} = \left[ \begin{array}{c}
\text{GoodUseOf_top_Files()} \\
\land \\
\text{GoodUseOf_intermediate_Files()} \\
\land \\
\text{GoodUseOf_internal_Files()} \\
\land \\
\text{GoodUseOf_shared_Files()}
\end{array} \right]
\]  

(4–11)

In formula (4–12), the definition for InclusionLoopsMayNotExist(), the third sub-predicate of (4–8) above is final. It states that inclusion loops are not permitted in file structures. This covers the special inclusion loops case of bi-directional inclusions.

Let \( \text{Loop} = \exists \text{SPA} \left[ \left( \forall j, (1 \leq j < n) = \left( \text{includes}(\text{SPA}[j], \text{SPA}[j+1]) \right) \right) \land \left( \text{includes}(\text{SPA}[n], \text{SPA}[1]) \right) \right] \), where

\( \text{SPA} = \text{SEQUENCE}[a_1, a_2, \ldots, a_n] \) of FileNames, then

\[ \text{InclusionLoopsMayNotExist()} \equiv (\forall n, (1 \leq n \leq \text{NumberOfFilesInCO_Files}) \Rightarrow \neg \text{Loop}) \]  

(4–12)

4.2.3.4 Explanation of (4–11): GoodUseOfTop_Files()

In formula (4–13), the definition for **GoodUseOf_top_Files(),** the first sub-predicate of (4–11) above is not final. It is formed by taking the conjunction of the three sub-predicates that are listed next and defined in more detail in section 4.2.3.5.
(1) \textit{NoFilesMayIncludeA\_top\_File():} this states that a top file cannot be "#include"d by any other file.

(2) \textit{FromSame\_Fr\_A\_top\_FileMayIncludeOnlyIts\_intermediate\_File():} this states that a top file can "#include" only the intermediate file that shares the same name (i.e. belong to the same component).

(3) \textit{A\_top\_FileMayInclude\_shared\_Files\_From\_Permitted\_Shared\_Pools():} this states that a top file can "#include" shared files from accessible shared-pools.

\[
\text{GoodUseOf\_top\_Files() } \equiv \\
\left\{ \begin{array}{l}
\text{NoFileMayIncludeA\_top\_File() } \land \\
\text{A\_top\_FileMayIncludeOnlyIts\_intermediate\_File\_From\_The\_Same\_Framework() } \land \\
\text{A\_top\_FileMayInclude\_shared\_Files\_Only\_From\_Permitted\_Shared\_Pools() } \\
\end{array} \right. \\
\] (4–13)

In formula (4–14), the definition for \textit{GoodUseOf\_intermediate\_Files()}, the second sub-predicate of (4–11) above is represented by the single predicate described in (1) next. The final mathematical representation for this predicate is provided in formula (4–20). Given that this predicate represents a rule for \textit{GoodUseOf\_intermediate\_Files()} and that more practice may uncover more rules for using intermediate files, we prefer not to replace this predicate by its final representation in (4–20). In the case of uncovering more rules for using intermediate files, then we can simply hook them through formula (4–14).

(1) \textit{An\_intermediate\_FileMayIncludeOnlyIts\_internal\_File\_And\_Other\_intermediate\_Files\_From\_Same\_Fr():}

this states that an intermediate file can "#include" only the internal file of the same component, as well as other intermediate files from the same framework.

\[
\text{GoodUseOf\_intermediate\_Files() } \equiv \\
\left( \text{An\_intermediate\_FileMayIncludeOnlyIts\_internal\_File\_And\_Other\_intermediate\_Files\_From\_Same\_Fr() } \right) \\
\] (4–14)
In formula (4–15), the definition for \textit{GoodUseOf\_internal\_Files()}, the third sub-predicate of (4–11) above is not final too. It is formed by taking the conjunction of the two sub-predicates that are listed next and defined in more detail in section 4.2.3.7.

(1) \textit{An\_internal\_FileMayNotIncludeAnyFile()}: this states that an internal (i.e. .i) file can not “\#include” any other file.

(2) \textit{NoFileMayIncludeAn\_internal\_FileExceptIts\_intermediate\_File()}: this states that an internal file can not be “\#include”d by any other file except the intermediate file that shares the same name (i.e. belongs to the same component).

\[
\text{GoodUseOf\_internal\_Files}() = \\
\left\{ \begin{array}{l}
\text{An\_internal\_FileMayNotIncludeAnyFile()} \\
\text{An\_internal\_FileCanBeIncludedOnlyByIts\_intermediate\_File()}
\end{array} \right\}
\] (4–15)

In formula (4–16), the definition for \textit{GoodUseOf\_shared\_Files()}, the fourth sub-predicate of (4–11) above is not final too. It is represented by a single predicate. The justification for not replacing this predicate by its final mathematical representation in formula (4–21) follows from the justification for (4–14) above.

(1) \textit{A\_shared\_FileMayIncludeOnlyOther\_shared\_FilesFromSame\_SP()}: this states that a shared file can “\#include” other shared files from the same shared-pool.

\[
\text{GoodUseOf\_shared\_Files}() = \\
(A\_shared\_FileMayIncludeOnlyOther\_shared\_FilesFromSame\_SP())
\] (4–16)

4.2.3.5 \textbf{Explanation of (4–13): GoodUseOf\_top\_Files()}

In formula (4–17), the definition for \textit{NoFileMayIncludeA\_top\_File()}, the first sub-predicate of (4–13) above is final. This states that a top file can not be “\#include”d by any other file. \textit{Includes(A, B)} is a boolean auxiliary function that returns TRUE if file A “\#include”s file B. This function is defined in the lexicon of this view.
\[ \text{NoFileMayIncludeA\_top\_File()} = \left\{ \begin{array}{l} \forall x, y, \left( x \neq y \land \right. \\
\left. (x \in \text{CO\_Files}) \land \right. \\
\left. (y \in \text{CO\_Files}) \land \right. \\
\left. (\text{Category}(x) = \text{top}') \right) \Rightarrow (\neg \text{Includes}(x, y)) \end{array} \right\} \]  

(4-17)

In formula (4-18), the definition for \( \text{FromSame\_Fr\_A\_top\_FileMayIncludeOnlyIts\_intermediate\_File()} \), the second sub-predicate of (4-13) above is final too. This states that a top file may "\#include" from the same framework only the intermediate file that shares the same filename (i.e. belongs to the same component). \( \text{FileName}(A) \) is an auxiliary function that returns the filename of file \( A \). This function is defined in the lexicon of this view.

Let \( P1 = \left\{ \begin{array}{l} (x \in \text{Fr\_Files}) \land \\
(\text{Category}(x) = \text{top}') \land \\
(\text{Includes}(x, y)) \end{array} \right\} \), and \( P2 = \left\{ \begin{array}{l} y \in \text{Fr\_Files} \land \\
(\text{Category}(y) = \text{\text{intermediate}'}) \land \\
(\text{FileName}(y) = \text{FileName}(x)) \end{array} \right\} \), then

\[ A\_top\_FileMayIncludeOnlyIts\_intermediate\_FileFromTheSameFramework() \equiv (\forall i, (1 \leq i \leq F) \Rightarrow (\forall x, y, P1 \Rightarrow P2)) \]  

(4-18)

In formula (4-19), the definition for \( A\_top\_FileMayInclude\_shared\_FilesFromPermitted\_Shared\_Pools() \), the third sub-predicate of (4-13) above is final too. This states that a top file may "\#include" a shared file from any permitted shared-pool. \( \text{Fr\_Name}(A) \) is an auxiliary function that returns the name of the framework to which file \( A \) belongs. This function is defined in the lexicon of this view.

59
CHAPTER 4: FIRST VIEW: FORMALIZING RULES FOR ORGANIZING FILE STRUCTURES OF Co-TOOLKITS

Let $P_1 = (\forall x \in Fr_Files_i \land y \in Sh_Files_i \land (\text{Category}(x) = 'top') \land (\text{Category}(y) = 'shared') \land (\text{includes}(x, y)))$, and

$P_2 = (\text{Fr_Name}(\text{boldFr_Files}_i) \in Frs\text{PermittedToAccessSP}(k))$ then

$A_{top\ file\ may\ include\ shared\ files\ only\ from\ permitted\ shared\ pools}() =

\left( \forall i, k \left( \left( 1 \leq i \leq F \right) \land \left( 0 \leq k \leq S \right) \Rightarrow (\forall x, y, P_1 \Rightarrow P_2) \right) \right) \quad (4-19)$

4.2.3.6 Explanation of (4-14): GoodUseOf_intermediate_Files()

In formula (4-20), the definition for $An\_intermediate\_file\ may\ include\ only\ its\ internal\ file\ and\ other\ intermediate\ files\ from\ same\ Fr()$, the second sub-predicate of (4-14) above is final too. This states that an intermediate file can "#include" only the internal file that shares the same filename (i.e. of the same component) as well as other intermediate files from the same framework.

Let $P_1 = (\forall x \in Fr_Files_i \land (\text{Category}(x) = 'intermediate') \land (\text{includes}(x, y)))$, $P_2 = (\left( (\text{FileName}(y) = \text{FileName}(x)) \land (\text{Category}(y) = 'internal') \right) \lor \left( (\text{FileName}(y) \neq \text{FileName}(x)) \land (\text{Category}(y) = 'intermediate') \right))$, and

$P_3 = (y \in Fr_Files_i \land P_2)$, then

$An\_intermediate\_file\ may\ include\ only\ its\ internal\ file\ and\ may\ include\ other\ intermediate\ files\ from\ same\ Fr() =

\left( \forall i, k \left( 1 \leq i \leq F \Rightarrow (\forall x, y, P_1 \Rightarrow P_3) \right) \right) \quad (4-20)$

4.2.3.7 Explanation of (4-15): GoodUseOf_internal_Files()

In formula (4-21), the definition for $An\_internal\_file\ may\ not\ include\ any\ file()$, the first sub-predicate of (4-15) above is final too. This states that an internal file can not "#include" any file.
An\_internal\_FileMayNotIncludeAnyFile() ≡
\[
\forall x, y, \left(\{x \in CO\_Files\} \land \left(\{y \in CO\_Files\} \land (Category(x) = 'internal')\right)\right) \Rightarrow (\neg Includes(x, y))
\]

(4-21)

In formula (4-22), the definition for An\_internal\_FileCanBeIncludedByOnlyIts_intermediate\_File(), the second sub-predicate of (4-15) above is final too. This states that an internal file can be "#include"d only by the intermediate file that shares the same file name (i.e. of the same component).

Let P1 = \left(\{y \in Fr\_Files\} \land (Category(y) = 'internal') \land (Includes(x, y))\right), and P2 = \left(\{Category(x) = 'intermediate'\} \land \{FileName(x) = FileName(y)\} \land \{x \in Fr\_Files\}\right), then

An\_internal\_FileCanBeIncludedByOnlyIts_intermediate\_File() ≡
\[
(\forall i, (1 \leq i \leq F) \Rightarrow (\forall x, y, P1 \Rightarrow P2))
\]

(4-22)

4.2.3.8 Explanation of (4-16): GoodUseOf\_shared\_Files()

In formula (4-23), the definition for A\_shared\_FileMayIncludeOnlyOther\_shared\_FilesFromSame\_SP(), the second sub-predicate of (4-16) above is final too. This states that a shared file can "#include" other shared files from the same shared-pool.

A\_shared\_FileMayIncludeOnlyOther\_shared\_FilesFromSame\_SP() ≡
\[
\forall k, (0 \leq k \leq S) \Rightarrow \left(\forall x, y, \left(\{x \in Sh\_Files_k\} \land Includes(x, y)\right) \Rightarrow (y \in Sh\_Files_k)\right)
\]

(4-23)

4.2.4 The View Lexicon

This sub-section describes the variables, symbols, acronyms, constants, and auxiliary functions that are used in formulas of the above view structure and its characteristics.

61
4.2.4.1 Variables And Acronyms

**COToolkit**: Represents any Component-Oriented Toolkit that can provide reusable components to be used to build new applications.

**CO_Files**: Represents the overall set of files that are used to implement the CO_toolkit that is under consideration.

**Fr_Files**: Represents the set of files that are used to implement \(i^{th}\) framework of the CO_toolkit that is under consideration.

**C_Files**: Represents the set of files that are used to implement \(j^{th}\) component in the \(i^{th}\) framework of the CO_toolkit that is under consideration.

**Sh_Files**: Represents the set of files that are used to implement \(k^{th}\) shared-pool of the CO_toolkit that is under consideration.

**Sh_File**: Represents the \(l^{th}\) file in the \(k^{th}\) shared-pool.

**topFile**: Represents the .cpp file of the \(j^{th}\) component in the \(i^{th}\) framework of the CO_toolkit that is under consideration.

**intermediateFile**: Represents the .h file of the \(j^{th}\) component in the \(i^{th}\) framework of the CO_toolkit that is under consideration.

**internalFile**: Represents the .i file of the \(j^{th}\) component in the \(i^{th}\) framework of the CO_toolkit that is under consideration.

4.2.4.2 Symbols And Constants

**F**: Number of Frameworks in the given CO_toolkit.

**C**: Number of Components in the \(i^{th}\) Framework.

**S**: Number of Shared-Pools.

**s**: Number of Files in the \(k^{th}\) Shared-Pool.

**CO**: Component Oriented.

**Fr**: Framework.

**SP**: Shared-Pool.
CHAPTER 4: FIRST VIEW: FORMALIZING RULES FOR ORGANIZING FILE STRUCTURES OF CO-TOOLKITS

PA: Path A.

PB: Path B.

NumberOfFilesInCO_Files: Represents the total number of files used to implement the given CO-Toolkit.

4.2.4.3 Auxiliary Functions

Category(X): An auxiliary function that returns the category of the file X. It could be 'top', 'intermediate', 'internal', or 'shared'.

FileName(X): An auxiliary function that returns the filename of file X.

Fr_Name(Fr_Files): An auxiliary function that returns the name of the framework that is implemented using the Fr_Files file subset.

Frs_PermittedToAccessSP(k): An auxiliary function that returns the set of frameworks that are permitted to access the kth shared-pool.

Includes(X, Y): An auxiliary Boolean function that returns TRUE or FALSE. It returns TRUE when file X is "#include"ing file Y, otherwise it returns FALSE.

Links(X, Y): An auxiliary Boolean function that returns TRUE or FALSE. It returns TRUE when file X contains parts of file Y, otherwise, it returns FALSE.

SEQUENCE[f_1, f_2, ..., f_n]: is used to denote a sequence of file names as per Hoffman’s notation [25].

4.3 Conclusions

In this chapter, we have shown the following:

(1) Rules and guidelines for CO–Toolkits’ file structures can be formalized using a convenient mathematical terminology and notation (c.f. section 4.2).

(2) Such formalized rules for file structures can be used to build a checking tool. A prototype tool that we built based on these rules is described in Appendix A.

(3) Such checking tools can be used to check CO–Toolkits’ file structures and report any rule violations back to the developers of these CO–Toolkits.
Examples are provided in Appendixes A2–A4. In particular, an example on a sample ACE file structure is presented in Appendix A4.
CHAPTER 5

SECOND VIEW: FORMALIZING RULES FOR DESIGNING AND IMPLEMENTING CO–TOOLKITS

5.1 Overview

The next section describes in details our view for CO–Toolkits' design and implementation structures. This view is based on the initial assumptions for good design practice that we made in chapter 3 (c.f. sections 3.3 and 3.3.3). Sections 2.4.1.3 and 2.4.2.2 in chapter 2 explain why we need to study the inheritance and association relations among components. Appendix B describes a prototype system that we built based on this view for CO–Toolkits' design and implementation structures. Two artificial examples and a third one on a sample ACE inheritance structure were used to demonstrate this prototype. Section 8.2 of chapter 8 discusses the limitations of the our views (c.f. chapters 4 through 6) including this view for design and implementation structures.

5.2 Towards A Mathematical View For CO–Toolkits' Design and Implementation Structures

For convenience, we have used throughout this view a simplified indexing scheme. For example, \( \text{Class}_{i} \) is used instead of \( \text{Class}_{i,j} \) to represent the \( i^{th} \) class in the

65
NOTE TO USERS

Page(s) not included in the original manuscript are unavailable from the author or university. The manuscript was microfilmed as received.

66

This reproduction is the best copy available.

UMI
FIGURE 5-1: Motivating example: Model for inheritance relation in CO-Toolkits
(1) \( \bigcup_{i=1}^{F} \text{Framework}_i \): this is used to represent the set of frameworks that are provided by the given CO-Toolkit. The \( \text{Framework}_i \) is used to represent the \( i^{th} \) framework such that \( 1 \leq i \leq F \), where \( F \) represents the number of frameworks in the given CO-Toolkit.

(2) \( \bigcup_{k=0}^{S} \text{SharedPool}_k \): this is used to represent the set of shared-pools that are provided in the given CO-Toolkit. The \( \text{SharedPool}_k \) is used to represent the \( k^{th} \) shared-pool such that \( 1 \leq k \leq S \), where \( S \) represents the number of shared-pools in the given CO-Toolkit. Readers should notice that a given CO-Toolkit should contain at least one framework.

\[
\text{COToolkit} = \bigcup_{i=1}^{F} \text{Framework}_i \bigcup_{k=0}^{S} \text{SharedPool}_k \tag{5-1}
\]

In formula (5-1), the \( \text{Component}_i \) is used to represent the \( j^{th} \) component of the \( i^{th} \) framework such that \( 1 \leq j \leq C_i \), where \( C_i \) represents the number of components in the \( i^{th} \) framework. Readers should notice that each framework should contain at least one component.

\[
\text{Framework}_i = \bigcup_{j=1}^{C_i} \text{Component}_i \tag{5-2}
\]

In formula (5-3), the \( \text{SharedClass}_i \) is used to represent the \( j^{th} \) class in the \( k^{th} \) shared-pool such that \( 0 \leq j \leq c_i \), and the \( \text{OtherSharedEntity}_i \) is used to represent the other entities that are provided in the \( k^{th} \) shared pool such that \( 0 \leq f \leq c_i \). The symbols \( c_k^e \) and \( e_k^f \) respectively, represent the number of classes and the number of other shared entities that are provided in the \( k^{th} \) shared-pool. Reader should notice that a shared-pool should contain at least a one shared entity.
\[ \text{SharedPool}_k = \left[ \text{SharedClassesSet}_k, \right. \quad \text{OtherSharedEntitiesSet}_k, \left. \right], \quad \text{where} \]

\[ \text{SharedClassesSet}_k = \bigcup_{j=0}^{d} \text{SharedClass}_{ji}, \quad \text{and} \quad \text{OtherSharedEntitiesSet}_k = \bigcup_{j=1}^{d} \text{SharedEntity}_{ij} \]  \hspace{1cm} (5-3)

### 5.2.2.1 Component: \( \text{Component}_{ij} \) As In (5-2)

In formula (5-4), a component is specified by a tuple of its interface and implementation. The \( \text{Interface}_{ij} \) and \( \text{Implementation}_{ij} \) are used to respectively represent the interface and the implementation of the \( ij^{th} \) component.

\[ \text{Component}_{ij} = \left[ \text{Interface}_{ij}, \quad \text{Implementation}_{ij} \right] \] \hspace{1cm} (5-4)

### 5.2.2.1.1 Component's Interface: \( \text{Interface}_{ij} \) As In (5-4)

In formula (5-5), the interface of the \( ij^{th} \) component is represented by the union of its local and inherited subsets of interfaces. \( \text{LocalInterface}_{ij} \) and \( \text{InheritedInterface}_{ij} \) are used to represent these two subsets.

\[ \text{Interface}_{ij} = \text{LocalInterface}_{ij} \bigcup \text{InheritedInterface}_{ij} \] \hspace{1cm} (5-5)

In formula (5-6), the local interface of the \( ij^{th} \) component is represented by the union of all of its local public methods' signatures. The variable \( \text{LocalPublicMethodSignature}_{ijk} \) is used to represent the signature of the \( k^{th} \) public method that belongs to the \( l^{th} \) class of those used to implement the \( ij^{th} \) component. We should notice that \( c_{ij} \) represents the number of classes that are used, along with the main section, to form the the \( ij^{th} \) component, while \( M_{il} \) represents the number of public methods that belong to the \( l^{th} \) class of that component.

69
CHAPTER 5: SECOND VIEW: FORMALIZING RULES FOR DESIGNING AND IMPLEMENTING CO-TOOLKITS

**FIGURE 5-2**: Motivating example: Model for association relation in CO-Toolkits
LocalInterface_{ij} = \bigcup_{l=1}^{s_l} \bigcup_{f=0}^{M_{ij}^f} \text{LocalPublicMethodSignature}_{ijf}

(5-6)

In formula (5-7), the inherited interface of the \(ij\)th component is represented by the union of all inherited public methods' signatures. The variable \(\text{PubliclyInheritedMethodSignature}_{ijf}\) is used to represent the signature of the \(f\)th publicly inherited method. This excludes those methods that are inherited among internal classes that form the same component. We should notice that \(M_{ij}^f\) represents the number of methods that are publicly inherited by the \(i\)th class of the \(ij\)th component.

InheritedInterface_{ij} = \bigcup_{l=1}^{s_l} \bigcup_{f=0}^{M_{ij}^f} \text{PubliclyInheritedMethodSignature}_{ijf}

(5-7)

In formula (5-8), the \(\text{LocalPublicMethodSignature}_{ij}\) of (5-6) is represented by the union of the data type of the value returned by the method of this signature, subset of the types of the input arguments and the subset of the types of the output arguments that are passed to and returned by that method. The auxiliary \(\text{Type}(\cdot)\) function is used next to return the type of its argument.

\[
\text{LocalPublicMethodSignature}_{ij} = \left[ \text{Type}(P\_ReturnedValue_{ij}) \right] \bigcup_{a=0}^{p_{\text{IN}}^a_{ij}} \text{Type}(P\_IN^a_{ij}) \bigcup_{b=0}^{p_{\text{OUT}}^b_{ij}} \text{Type}(P\_OUT^b_{ij})
\]

(5-8)

As in (5-8) above, in formula (5-9), the signature of the \(f\)th method of those inherited by the \(i\)th class of the \(ij\)th component is also represented by the union of the data type of the returned value, the subset of the inputs' types, and the subset of the outputs' types of the \(ijkl\)th inherited method.

\[
\text{PubliclyInheritedMethodSignature}_{ijf} = \left[ \text{Type}(P_{ijkl\text{ReturnedValue}}_{ijf}) \right] \bigcup_{a=0}^{p_{\text{IN}}^a_{ijkl}^{ijf}} \text{Type}(P_{ijkl\text{IN}}^a_{ijf}) \bigcup_{\zeta=0}^{p_{\text{OUT}}^\zeta_{ijkl}^{ijf}} \text{Type}(P_{ijkl\text{OUT}}^\zeta_{ijf})
\]

(5-9)
5.2.2.1.2 Component’s Implementation: Implementation\_i\_y As In (5–4)

In formula (5–10), the implementation of the \( ij^{th} \) component is represented by the implementation of its main section, and the union of the implementations of its internal classes. The \( \text{MainPart}\_i\_y \) is used to represent the main part of the \( ij^{th} \) component. The \( \text{InternalClass} \_i\_y \) is used to represent the \( l^{th} \) class of those used to form the \( ij^{th} \) component.

\[
\text{Implementation}\_i\_y = \left[ \text{MainPart}\_i\_y , \text{InternalClassesSet}\_i\_y \right] \quad \text{where} \quad \text{InternalClassesSet}\_i\_y = \bigcup_{j=1}^{e_y} \text{InternalClass}\_i\_y
\]  

(5–10)

In formula (5–11), the \( \text{InternalClass}\_i\_y \) of (5–10) above is represented by the implementation of its public, protected, and private sections.

\[
\text{InternalClass}\_i\_y = \left[ \text{PublicSection}\_i\_y , \text{ProtectedSection}\_i\_y , \text{PrivateSection}\_i\_y \right]
\]  

(5–11)

In formula (5–12), the \( \text{PublicSection}\_i\_y \) of (5–11) above is represented by the set of its public methods. We should notice that there are no objects/variables present in this definition. This is to exclude any data structure entities (i.e. to maintain the concept of information hiding) from the public section of any class.

\[
\text{PublicSection}\_i\_y = \bigcup_{i=1}^{m_y} \text{PublicMethod}\_i\_y
\]  

(5–12)

In formula (5–13), the \( \text{PublicMethod}\_i\_y \) of (5–12) above is defined. Readers can refer to the lexicon of this view for more details about the variables and symbols used in this definition.

\[
\text{PublicMethod}\_i\_y = \text{SOF}_{i\_y} \left( \text{P\_INs}_{ijlk} , \text{P\_OUTs}_{ijlk} \right) \quad \text{where}
\]

\[
P\_INs_{ijlk} = \bigcup_{a=0}^{p_{\text{IN}}^{\text{OUT}}} P\_INs^a_{i\_y} \quad \text{and} \quad P\_OUTs_{i\_y} = \bigcup_{b=0}^{p_{\text{OUT}}^{\text{OUT}}} P\_OUTs^b_{i\_y}
\]  

(5–13)
In formula (5–14), the ProtectedSection$_{\alpha}$ of (5–11) above is represented by a tuple of the set of the protected objects and the set of the protected methods of the $l^{th}$ class of the $ij^{th}$ component. We should notice that protected sections may contain objects/variables as well as protected methods.

\[
\text{ProtectedSection}_{\alpha} = \left[ \bigcup_{i=0}^{o^i_\alpha} \text{ProtectedObjects}_{\alpha} \bigcup_{m^i_\alpha} \text{ProtectedMethod}_{\alpha} \right]
\]  

(5–14)

In formula (5–15), the ProtectedMethod$_{\alpha}$ of (5–14) above is defined. Readers can refer to the lexicon of this view for more details about the variables and symbols used in this definition.

\[
\text{ProtectedMethod}_{\alpha} = \text{SOF}_{\alpha}(R_{\text{INS}_{\alpha}}, R_{\text{OUTS}_{\alpha}}) \quad \text{where}
\]

\[
R_{\text{INS}_{\alpha}} = \bigcup_{\alpha=0}^{\alpha} R_{\text{INS}_{\alpha}} \quad \text{and} \quad R_{\text{OUTS}_{\alpha}} = [R_{\text{ReturnedValue}_{\alpha}}] \bigcup_{\beta=0}^{\beta} R_{\text{OUT}_{\beta}}
\]  

(5–15)

In formula (5–16), the PrivateObject$_{\alpha}$ and PrivateMethod$_{\alpha}$ are used to respectively represent the private objects and methods of the $l^{th}$ class of the $ij^{th}$ component. We should notice that private sections may contain private methods besides the the private objects/variables.

\[
\text{PrivateSection}_{\alpha} = \left[ \bigcup_{i=0}^{o^i_\alpha} \text{PrivateObjects}_{\alpha} \bigcup_{m^i_\alpha} \text{PrivateMethod}_{\alpha} \right]
\]  

(5–16)

In formula (5–17), the PrivateMethod$_{\alpha}$ of (5–16) above is defined. Readers can refer to the lexicon of this view for more details about the variables and symbols used in this definition.
PrivateMethod_{\psi} = SOF_{\psi}(V_{INs_{\psi}}, V_{OUTs_{\psi}}), \text{ where}

V_{INs_{\psi}} = \bigcup_{a=0}^{n_{\psi}} V_{IN_{a}} \quad \text{and} \quad V_{OUTs_{\psi}} = \{V_{ReturnedValue_{\psi}}\} \bigcup \bigcup_{\beta=\psi}^{\rho} V_{OUT_{\beta}} \quad (5-17)

5.2.3 The View Characteristics

This sub-section describes the set of characteristics (i.e. rules) that we think should be maintained by the above view. Readers can refer to the lexicon section of this view (cf. section 5.2.4) for the meanings of the variables and symbols that appear in the various definitions throughout this section. As long as the predicate \text{GoodDesignAndImplementation()} of (5-18) is true, we say that the above view is satisfied. This predicate is represented by the next listed sub-predicate that is defined in more details in section 5.2.3.1. In order to keep this view open to any new rules that can be found in further research, we prefer not to replace this \text{GoodUseOfRelationsAmongComponents()} by its representation that appears in section 5.2.3.1.

(1) \text{GoodUseOfRelationsAmongComponents()}: as explained in (5-19), this sub-predicate is used to ensure that relations among these components are acceptable. Throughout this section, each of these sub-predicates is explained in more and more details.

\text{GoodDesignAndImplementation()} \equiv \left( \text{GoodUseOfRelationsAmongComponenets()} \right) \quad (5-18)

5.2.3.1 Explanation of (5-18): \text{GoodDesignAndImplementation()}

In formula (5-19), the \text{GoodUseOfRelationsAmongComponents()} of (5-18) above, is formed by taking the conjunction of the three sub-predicates that are listed next and defined in more details in section 5.2.3.2.
(1) SafeInheritance(): as explained in (5–20), this sub-predicate is used to ensure that inheritance is used safely from our point of view.

(2) GoodAssociation(): as explained in (5–21), this sub-predicate is used to ensure that association is used in a good way from our point of view.

(3) GoodUseOfOtherRelations(): this sub-predicate is for any potential extension to this second view of CO-Toolkits. Any relation other than inheritance and association relations that are covered by this view can be glued to this view through this sub-predicate in the future.

\[
\text{GoodUseOfRelationsAmongComponents()} = \begin{cases} 
\text{SafeInheritance()} \land \\
\text{GoodAssociation()} \land \\
\text{GoodUseOfOtherRelations()} 
\end{cases} 
\]  
(5–19)

5.2.3.2 Explanation of (5–19): GoodUseOfRelationsAmongComponents()

In formula (5–20), the SafeInheritance() sub-predicate of (5–19) above, is formed by taking the conjunction of the four sub-predicates that are listed next and defined in section 5.2.3.3.

(1) InheritanceMayNotExistAcrossFrameworks(): as explained in (5–23), this sub-predicate is used to ensure that there is no inheritance among components that belong to different frameworks. We have recommended this since inheritance is not safe by default and in this case, inheritance would break the encapsulation at the level of frameworks.

(2) SafeInheritance_AmongComponents_WithinTheSameFramework(): as explained in (5–24), this sub-predicate is used to ensure that inheritance is safe whenever used among components within the same framework.
(3) \textit{SafeInheritance\_AmongClasses\_WithinTheSameComponent()}: as explained in (5–25), this sub-predicate is used to ensure that inheritance is safe whenever used among components within the same framework.

(4) \textit{SafeInheritance\_AmongClasses\_WithinTheSameSharedPool()}: as explained in (5–26), this sub-predicate is used to ensure that inheritance is safe whenever used among shared classes within the same shared pool.

\[
\text{SafeInheritance}() \equiv \left\{ \begin{array}{l}
\text{InheritanceMayNotExistAcrossFrameworks()} \land \\
\text{SafeInheritance\_AmongComponents\_WithinTheSameFramework()} \land \\
\text{SafeInheritance\_AmongClasses\_WithinTheSameComponent()} \land \\
\text{SafeInheritance\_AmongClasses\_WithinTheSameSharedPool()}
\end{array} \right\} \tag{5–20}
\]

In formula (5–21), the \textit{GoodAssociation()} sub-predicate is formed by taking the conjunction of the three sub-predicates that are listed next and defined in more details in section 5.2.3.4.

(1) \textit{GoodAssociation\_OnTheComponentLevel()}: as explained in (5–27), this sub-predicate is used to ensure that association among components is acceptable.

(2) \textit{GoodAssociation\_OnTheInternalClassesLevel()}: as explained in (5–28), this sub-predicate is used to ensure that association among internal classes within the same component is acceptable.

(3) \textit{Association\_May\_Not\_Exist\_Across\_Shared\_Pools()}: this sub-predicate is used to ensure that association does not exist among classes from different shared pools.

This design decision can keep frameworks independent from each other.

\[
\text{GoodAssociation}() \equiv \left\{ \begin{array}{l}
\text{GoodAssociation\_OnTheComponentsLevel()} \land \\
\text{GoodAssociation\_OnTheInternalClassesLevel()} \land \\
\text{Association\_May\_Not\_Exist\_Among\_Shared\_Pools}
\end{array} \right\} \tag{5–21}
\]
As mentioned earlier in this section, the \texttt{GoodUseOfOtherRelations()} sub-predicate is left for future expansion.

\texttt{GoodUseOfOtherRelations()} \iffalse This will be left for potential expansion in the future \fi \hfill (5-22)

5.2.3.3 Explanation of (5–20): \texttt{SafeInheritance()}

In formula (5–23), predicate logic is used to specify our design decision for prohibiting components to inherit from each other across frameworks. This decision has been already justified in section 5.2.3.2.

Let \( P1 = \left( (\text{Component}_i \in \text{Framework}_j) \land (\text{Inherits(\text{Component}_i, \text{Component}_j)}) \right) \), and

\( P2 = \left( (k = i) \land (\text{Component}_k \in \text{Framework}_i) \right) \), then

\( \text{InheritanceMayNotExistAcrossFrameworks()} \iffalse \fi \hfill (5-23) \fi

\[
\left( \forall i, k, \left(1 \leq i \leq F \land 1 \leq k \leq F \right) \Rightarrow \left( \forall j, l, \left(1 \leq j \leq C_i \land 1 \leq l \leq C_i \right) \Rightarrow (P1 \Rightarrow P2) \right) \right)
\]

In formula (5–24), the \texttt{SafeInheritance_AmongComponents_WithinTheSameFramework()} sub-predicate of (5–20) above is formed by taking the conjunction of the two sub-predicates that are listed next and defined in more details in section 5.2.3.5.

(1) \texttt{A_ComponentMayInheritFromOnlyOneOtherComponentsFromTheSameFramework()}:
as explained in (5–30), this sub-predicate is used to ensure that inheritance among components within the same framework can be only single (i.e. multiple inheritance is prohibited). It has been already justified in section 2.4.1.2 that multiple inheritance causes more complications to CO–Toolkits as well as to the applications that depend on components from these CO–Toolkits.

(2) \texttt{OnlyPublicInheritanceMayExistAcrossComponentsWithinTheSameFramework()}: as explained in (5–31), this sub-predicate states that only public inheritance may

77
exist among components within the same framework. By this, the protected and private inheritance (i.e. allowed in C++) are not allowed to exist among classes that belong to different components, even within the same framework. Protected and private inheritance add external stuff to the protected and private sections of the inhering classes (i.e. affect the implementation of the inheriting classes). However, we do permit protected and private inheritance among classes given that both the inhering and the inherited from classes belong to the same component.

$$\text{SafeInheritance\_Among\_Components\_Within\_The\_Same\_Framework}() \equiv$$

\[
\big( \text{A\_Component\_May\_Inherit\_From\_Only\_One\_Other\_Components\_From\_The\_Same\_Framework}() \land \\
\text{Only\_Public\_Inheritance\_May\_Exist\_Across\_Components\_Within\_The\_Same\_Framework}() \big)
\] (5-24)

In formula (5-25), the $\text{SafeInheritance\_Among\_Components\_Within\_The\_Same\_Component}()$ sub-predicate of (5-20) above is formed by taking the conjunction of the two sub-predicates that are listed next and defined in more details in section 5.2.3.6.

1. $\text{No\_Multiple\_Inheritance\_May\_Exist\_Among\_Classes\_Within\_The\_Same\_Component}()$: as explained in (5-32), this sub-predicate is used to ensure that inheritance among internal classes within the same component can be only single (i.e. multiple inheritance is prohibited). The justification for this decision is similar to the justification for permitting only single inheritance at the components level mentioned above.

2. $\text{Inheritance\_May\_Be\_Public}/\text{Protected}/\text{or}/\text{Private\_Only\_Within\_The\_Same\_Component}()$: as explained in (5-31), this sub-predicate states that inheritance among internal classes within the same component could be public, protected, or private. The justification for this is that there is no danger from permitting this as long as these internal classes belong to the same component. That means, the way in-
ternal classes collaborate with each other is an internal design issue and that will not be exposed to the externals.

\[
\text{SafeInheritance\_AmongClasses\_WithinTheSameComponent} = \left( \text{NoMultipleInheritanceMayExistAmongClassesWithinTheSameComponent} \land \text{InheritanceMayBe\_Public/Protected/\ or/\ Private\_OnlyWithinTheSameComponent} \right)
\] (5-25)

In formula (5-26), the \text{SafeInheritance\_AmongClasses\_WithinTheSameSharedPool} sub-predicate of (5-20) above is formed by taking the conjunction of the two sub-predicates that are listed next and defined in more details in section 5.2.3.7.

1. \text{InheritanceMayNotExistAcrossSharedPools}: as explained in (5-34), this sub-predicate states that inheritance may not exist among classes from different shared pools. This is similar to the previous decision for prohibiting inheritance across frameworks.

2. \text{NoMultipleInheritanceMayExistAmongClassesWithinTheSameSharedPool}: as explained in (5-35), this sub-predicate states that multiple inheritance is prohibited among classes within the same shared pool. Again this similar to the previous decision for prohibiting multiple inheritance among components within the same framework as well as among internal classes within the same component.

\[
\text{SafeInheritance\_AmongClasses\_WithinTheSharedPool} = \left( \text{InheritanceMayNotExistAcrossSharedPools} \land \text{NoMultipleInheritanceMayExistAmongClassesWithinTheSameSharedPool} \right)
\] (5-26)

5.2.3.4 Explanation of (5-21): \text{GoodAssociation()}

In formula (5-27), the \text{GoodAssociation\_OnTheComponentsLevel} sub-predicate of (5-21) above is formed by taking the conjunction of the two sub-predicates that are listed next and defined in more details in section 5.2.3.8.
(1) \textit{ComponentsMayHaveOnlyUniDirectionalAssociationWithClassesFromSharedPools()}:

as explained in (5–36), this sub-predicate states that only uni-directional association from components to shared-pools' classes. This means that a component may contain an object of a shared class. However, a shared class may not contain an object of any component's internal class since it makes no sense to permit this.

(2) \textit{AssociationMayNotExistAcrossFrameworks()}: as explained in (5–37), this sub-predicate is used to ensure that association does not exist among components from different frameworks. This design decision can keep frameworks independent from each other.

\[ \text{GoodAssociationOnTheComponentsLevel()} = (\text{ComponentsMayHaveOnlyUniDirectionalAssociationWithClassesFromTheSharedPools()} \land \text{AssociationMayNotExistAcrossFrameworks()}) \quad (5–27) \]

In formula (5–28), the \textit{GoodAssociationOnTheInternalClassesLevel()} sub-predicate of (5–21) above is represented by a single sub-predicate. The justification for not replacing this sub-predicate by its final mathematical representation in (5–38) is that we want to keep this view open for any new rule that may be found in further research about using association among internal classes.

(1) \textit{AssociationAndInheritanceMayNotExistBetweenTwoInternalClassesAtTheSameTime()}:

as explained in (5–38), this sub-predicate states that both association and inheritance relations may not exist at the same time between the same two internal classes. This has been already justified in section 5.2.3.4.

\[ \text{GoodAssociationOnTheInternalClassesLevel()} = \text{AssociationAndInheritanceMayNotExistBetweenTwoClassesAtTheSameTime()} \quad (5–28) \]
In formula (5–29), predicate logic is used to specify our design decision for prohibiting components to associate with each other across frameworks.

Let \( P_1 = \left( \left( \text{Component}_q \in \text{Framework}_k \right) \land \left( \text{Associates}(\text{Component}_q, \text{Component}_u) \right) \right) \), and
\[ P_2 = \left( \left( (k = i) \land (\text{Component}_u \in \text{Framework}_k) \right) \right) \]

\[ \text{InheritanceMayNotExistAcrossFrameworks}() \equiv \left( \forall i, k, \left( (1 \leq i \leq F) \land (1 \leq k \leq F) \right) \Rightarrow \left( \forall j, l, \left( (1 \leq j \leq C_i) \land (1 \leq l \leq C_k) \right) \Rightarrow (P_1 \Rightarrow P_2) \right) \right) \] (5–29)

5.2.3.5 Explanation of (5–24): \text{SafeInheritance_AmongComponents_WithinTheSameFramework}()

In formula (5–30), the definition for the first predicate of (5–24) above is final. We used the macros \( P_1, P_2, P_3, \) and \( P_4 \) just to simplify this predicate and improve its readability.

Let \( P_1 = \left( \left( (1 \leq j \leq C_i) \land \left( \text{Component}_q \in \text{Framework}_k \right) \right) \right) \), \( P_2 = \left( \exists k, l, \left( (1 \leq k \leq C_i) \land (k \neq j) \land \left( \text{Component}_u \in \text{Framework}_k \right) \land \left( \text{Inherits}(\text{Component}_u, \text{Component}_q) \right) \right) \right) \), and \( P_3 = \left( \left( (1 \leq l \leq C_k) \land (l \neq j) \land (l \neq k) \land \left( \text{Component}_u \in \text{Framework}_k \right) \right) \), and \( P_4 = \left( \text{Inherits}(\text{Component}_u, \text{Component}_q) \right) \), then

\[ \text{A_ComponentMayInheritFromOnlyOneOtherComponentFromTheSameFramework}() \equiv \left( \forall i, (1 \leq i \leq F) \Rightarrow (\forall j, (P_1 \land P_2) \Rightarrow (\forall l, P_3 \Rightarrow \neg P_4)) \right) \] (5–30)

In formula (5–31), the definition for the second predicate of (5–24) above is final too. Similar to (5–30), we used the macros \( P_1, \) and \( P_2 \) just to simplify this predicate and improve its readability.
Now let $P_1 = \left( \begin{array}{c} (1 \leq j \leq C_i) \land \\ (1 \leq k \leq C_i) \land (k \neq j) \land \\ (\text{Component}_k \in \text{Framework}) \land \\ (\text{Component}_j \in \text{Framework}) \land \\ (\text{Inherits(\text{Component}_j, \text{Component}_k)}) \end{array} \right)$, and $P_2 = (\text{InheritanceCategory(}\text{Component}_j, \text{Component}_k)).$

then

\[
\text{OnlyPublicInheritanceMayExistAcrossComponentsWithinTheSameFramework()} \equiv \\
(\forall i, (1 \leq i \leq F) \Rightarrow (\forall j, k, (P_1 \Rightarrow P_2)))
\]

(5–31)

5.2.3.6 Explanation of (5–25): \text{SafeInheritance_AmongClasses_WithinTheSameComponent()}

In formula (5–32), the definition for the first predicate of (5–25) above is final. We used the macros $P_1, P_2, P_3, P_4,$ and $P_5$ just to simplify this predicate and improve its readability.

Let $P_1 = \left( \begin{array}{c} (1 \leq j \leq C_i) \land \\ (\text{Component}_j \in \text{Framework}) \end{array} \right),$ $P_2 = \left( \begin{array}{c} (1 \leq l \leq C_d) \land \\ (\text{InternalClass}_d \in \text{InternalClassesSet}_d) \end{array} \right),$ $P_3 \equiv \left( \begin{array}{c} (1 \leq k \leq C_d) \land \\ (k \neq l) \land \\ (\text{InternalClass}_k \in \text{InternalClassesSet}_d) \land \\ (\text{Inherits(InternalClass}_k, \text{InternalClass}_k)) \end{array} \right),$ $P_4 \equiv \left( \begin{array}{c} (1 \leq f \leq C_d) \land \\ (f \neq k) \land (f \neq l) \land \\ (\text{InternalClass}_d \in \text{InternalClassesSet}_d) \end{array} \right),$ and $P_5 = (\text{Inherits(Class}_d, \text{Class}_d))$ then

\[
\text{NoMultipleInheritanceMayExistAmongClassesWithinTheSameComponent()} \equiv \\
(\forall i(1 \leq i \leq F) \Rightarrow (\forall j, P_1 \Rightarrow (\forall l, (P_2 \land P_3) \Rightarrow (P_4 \Rightarrow \neg P_5))))
\]

(5–32)

In formula (5–33), the definition for the second predicate of (5–25) above is final too. We used the macros $P_1,$ and $P_2$ just to simplify this predicate and improve its readability.
CHAPTER 5: SECOND VIEW: FORMALIZING RULES FOR DESIGNING AND IMPLEMENTING CO-TOOLKITS

Now let $P_1 = \left\{ \begin{array}{l}
(1 \leq j \leq C_i) \\
(1 \leq k \leq C_i) \\
(k \neq j)
\end{array} \right\}$, $P_2 = \left\{ \begin{array}{l}
(\text{InternalClass}_i \in \text{InternalClassesSet}_i) \\
(\text{InternalClass}_k \in \text{InternalClassesSet}_i) \\
(\text{Inherits(InternalClass}_i, \text{InternalClass}_k))
\end{array} \right\}$, and

$P_3 = \left\{ \begin{array}{l}
\text{InheritanceCategory(InternalClass}_i, \text{InternalClass}_k) \in \{\text{public'} , \\
\text{protected'}, \text{private'} \}
\end{array} \right\}$, then

$\text{InheritanceMayBe_Public/Protected/or/Private_OnlyWithinTheSameComponent()} \equiv
(\forall i, (1 \leq i \leq F) \Rightarrow (\forall j, k, P_1 \Rightarrow (\forall l, (1 \leq l \leq c_i) \Rightarrow (P_2 \Rightarrow P_3)))) \quad (5\text{-}33)$

5.2.3.7 Explanation of (5\text{-}26): SafeInheritance_AmongClasses_WithinTheSharedPools()

In formula (5\text{-}34), the definition for the first predicate of (5\text{-}26) above is final.

We used the macros $P_1$, $P_2$, and $P_3$ just to simplify this predicate and improve its readability.

Let $P_1 = \left\{ \begin{array}{l}
(\text{SharedClass}_i \in \text{SharedClassesSet}_i) \\
(\text{SharedClass}_k \in \text{SharedClassesSet}_i) \\
(\text{Inherits(SharedClass}_i, \text{SharedClass}_k))
\end{array} \right\}$, and $P_2 = \left\{ \begin{array}{l}
(i = k) \land (j \neq l) \\
(\text{SharedPool}_i = \text{SharedPool}_k)
\end{array} \right\}$, then

$\text{InheritanceMayNotExistAcrossSharedPools()} \equiv
(\forall i, k, (1 \leq i \leq S) \land (1 \leq k \leq S) \Rightarrow (\forall j, l, (1 \leq j \leq c_i) \land (1 \leq l \leq c_i) \Rightarrow (P_1 \Rightarrow P_2))) \quad (5\text{-}34)$

In formula (5\text{-}35), the definition for the second predicate of (5\text{-}26) above is final too. We used the macros $P_1$, $P_2$, and $P_3$ just to simplify this predicate and improve its readability.

Now let $P_1 = \left\{ \begin{array}{l}
(1 \leq k \leq c_i) \land (k \neq j) \\
(\text{SharedClass}_k \in \text{SharedClassesSet}_i) \\
(\text{Inherits(SharedClass}_i, \text{SharedClass}_k))
\end{array} \right\}$, and $P_2 = (\text{SharedClass}_i \in \text{SharedClassesSet}_i)$, and $P_3 = (\text{Inherits(SharedClass}_i, \text{SharedClass}_a))$, then

83
NoMultipleInheritanceMayExistAmongClassesWithinTheSameSharedPool() \equiv
\left[\forall i, (1 \leq i \leq S) \implies \forall j, (1 \leq j \leq e_i^t) \implies \left( P_1 \implies \left( \forall i, \left( (1 \leq f \leq e_i^t) \land (f \neq j) \land (f \neq k) \right) \implies (P_2 \implies \neg P_3) \right) \right) \right] \quad (5-35)

5.2.3.8 Explanation of (5-27): GoodAssociationOnTheComponentLevel()

In formula (5-36), the definition for the first predicate of (5-27) above is final. We used the macros $P_1$, and $P_2$ just to simplify this predicate and improve its readability.

Let $P_1 \equiv \left( \left( \text{Component}_{ij} \in \text{Framework}_i \right) \land \left( \text{InternalClass}_{ij} \in \text{InternalClassesSet}_i \right) \land \left( \text{SharedClass}_{ij} \in \text{SharedClassesSet}_i \right) \land \left( \text{ Associates(InternalClass}_{ij}, \text{SharedClass}_{ij}) \right) \right)$, and

\begin{align*}
P_2 &\equiv \left( \left( \text{Fr_Name(\text{Framework}_i) \in Frs_PermittedToAccess_SP(k)} \right) \land \left( \text{One_to_One UniDirectionalAssociation(Class}_{ij}, \text{SharedClass}_{ij}) \right) \land \neg \left( \text{BiDirectionalAssociation(Class}_{ij}, \text{SharedClass}_{ij}) \right) \right) \end{align*}

ComponentsMayHaveOnlyUnidirectionalAssociationWithClassesFromTheSharedPools() \equiv
\left[ \forall i, k, \left( (1 \leq i \leq F) \land (1 \leq k \leq S) \right) \implies \forall j, l, F, \left( (1 \leq j \leq C_i) \land (1 \leq l \leq C_i) \land (1 \leq f \leq e_i^t) \right) \implies (P_1 \implies P_2) \right] \quad (5-36)

In formula (5-37), the definition for the first predicate of (5-27) above is final too. We used the macros $P_1$, and $P_2$ just to simplify this predicate and improve its readability.

Now let $P_1 \equiv \left( \left( \text{Component}_{ij} \in \text{Framework}_i \right) \land \left( \text{Component}_{ij} \in \text{Framework}_k \right) \right)$, and $P_2 \equiv \left( \text{ Associates(\text{Component}_{ij}, \text{Component}_{ij})} \right)$, then

AssociationMayNotExistAcrossFrameworks() \equiv
\left[ \forall i, k, \left( (1 \leq i \leq F) \land (1 \leq k \leq F) \land (i \neq k) \right) \implies (\forall j, l, F, \left( (1 \leq j \leq C_i) \land (1 \leq l \leq C_i) \right) \implies (P_1 \implies \neg P_2) \right] \quad (5-37)
5.2.3.9 Explanation of (5–28): GoodAssociationOnTheInternalClassesLevel()

In formula (5–38), the definition for the first predicate of (5–28) above is final too. We used the macros $P_1$, and $P_2$ just to simplify this predicate and improve its readability.

Let $P_1 = \left( (\text{Component}_i \in \text{Framework}_i) \land (\text{Component}_a \in \text{Framework}_a) \land (\text{InternalClass}_a \in \text{InternalClassesSet}_a) \land (\text{InternalClass}_a \in \text{InternalClassesSet}_a) \right)$, and

$P_2 = \left( (\text{Inherits}(\text{InternalClass}_a, \text{InternalClass}_a)) \land (\text{Associates}(\text{InternalClass}_a, \text{InternalClass}_a)) \right)$, then

$\text{AssociationAndInheritanceMayNotExistBetweenTwoClassesAtTheSameTime()} = \left( \forall i, (1 \leq i \leq F) \Rightarrow \left( \forall j,k, (1 \leq j \leq C_i) \land (1 \leq k \leq C_i) \Rightarrow \left( \forall f,l, (1 \leq f \leq C_a) \land ((j = k) \Rightarrow (l = f)) \Rightarrow \left( P_1 \Rightarrow \neg P_2 \right) \right) \right) \right)$ (5–38)

5.2.4 The View Lexicon

In this subsection, we clarify all variables, symbols, acronyms, constants, and auxiliary functions that appear in the formalism of the above view and its characteristics.

5.2.4.1 Variables And Acronyms

$CO$: Component Oriented.

$COToolkit$: Represents any Component–Oriented Toolkit that can provide reusable components to be used to build new applications.

$\text{Component}_i$: Represents the $j^{th}$ component in the $i^{th}$ framework.

$\text{Framework}_i$: Represents the $i^{th}$ framework.

$\text{Implementation}_i$: Represents the implementation of the $ij^{th}$ component.

$\text{InheritedInterface}_i$: Represents the inherited interface of the $ij^{th}$ component. This refers to the interface of the methods that are inherited by that component.
Interface: Represents the interface of the $i^j$th component.

InternalClassesSet: Represents the set of the internal classes in the $i^j$th component.

InternalClass: Represents the implementation of the $l^i$th internal class in the $i^j$th component.

LocalInterface: Represents the local interface of the $i^j$th component. This refers to the interface of the local non-inherited methods that belong to that component.

PublicMethod: Represents the implementation of the $k^i$th public method of the $l^i$th internal class in the $i^j$th component.

LocalPublicMethodSignature: Represents the signature (interface description) of the $k^i$th local method in the $l^i$th class that belongs to the $i^j$th component.

MainPart: Represents the implementation of the main part of the $i^j$th component.

OtherSharedEntitiesSet: Represents the set of other shared entities in the $k^i$th shared-pool.

OtherSharedEntity: Represents the $j^i$th other shared entity in the $k^i$th shared-pool.

PubliclyInheritedMethodSignature: Represents the signature (interface description) of the $f^i$th method that is inherited by the $l^i$th class of the $i^j$th component.

PrivateMethod: Represents the implementation of the $f^i$th private method of the $l^i$th internal class in the $i^j$th component.

PrivateObject: Represents the $k^i$th private object of the $l^i$th internal class in the $i^j$th component.

PrivateSection: Represents the implementation of the private section of the $l^i$th internal class of the $i^j$th component.

ProtectedObject: Represents the $k^i$th protected object of the $l^i$th internal class of the $i^j$th component.

ProtectedMethod: Represents the implementation of the $f^i$th protected method of the $l^i$th internal class in the $i^j$th component.

ProtectedSection: Represents the implementation of the protected section of the $l^i$th internal class in the $i^j$th component.

PublicSection: Represents the implementation of the public section of the $l^i$th internal class in the $i^j$th component.

P_In: Represents the sequence of inputs to the $k^i$th public local method of the $l^i$th internal class in the $i^j$th component.

P_Out: Represents the sequence of outputs from the $k^i$th public local method in the $l^i$th internal class of the $i^j$th component.
\( P_{ReturnedValue}^{ijkl} \): Represents the returned value by the \( k^{th} \) public local method of the \( l^{th} \) internal class in the \( ij^{th} \) component.

\( P_{\text{INs}}^{ijkl} \): Represents the sequence of inputs to the \( f^{th} \) publicly inherited method of the \( l^{th} \) internal class in the \( ij^{th} \) component.

\( P_{\text{OUTs}}^{ijkl} \): Represents the sequence of outputs from the \( f^{th} \) publicly inherited method of the \( l^{th} \) internal class in the \( ij^{th} \) component.

\( P_{\text{ReturnedValue}}^{ijkl} \): Represents the returned value by the \( f^{th} \) publicly inherited method of the \( l^{th} \) internal class in the \( ij^{th} \) component.

\( R_{\text{INs}}^{ijkl} \): Represents the sequence of inputs to the \( f^{th} \) protected method of the \( l^{th} \) internal class in the \( ij^{th} \) component.

\( R_{\text{OUTs}}^{ijkl} \): Represents the sequence of outputs from the \( f^{th} \) protected method of the \( l^{th} \) internal class in the \( ij^{th} \) component.

\( R_{\text{ReturnedValue}}^{ijkl} \): Represents the returned value by the \( f^{th} \) protected method of the \( l^{th} \) internal class in the \( ij^{th} \) component.

\( SharedPool_i \): Represents the \( k^{th} \) shared-pool.

\( SharedClass_{ij} \): Represents the \( j^{th} \) class in the \( k^{th} \) shared-pool.

\( SOF_{x}(A, B) \): Represents the implementation of method \( X \).

\( V_{\text{INs}}^{ijkl} \): Represents the sequence of inputs to the \( f^{th} \) private method of the \( l^{th} \) internal class in the \( ij^{th} \) component.

\( V_{\text{OUTs}}^{ijkl} \): Represents the sequence of outputs from the \( f^{th} \) private method of the \( l^{th} \) internal class in the \( ij^{th} \) component.

\( V_{\text{ReturnedValue}}^{ijkl} \): Represents the returned value by the \( f^{th} \) private method of the \( l^{th} \) internal class in the \( ij^{th} \) component.

5.2.4.2 Symbols And Constants

\( F \): Represents the number of frameworks.

\( S \): Represents the number of shared-pools.

\( C_i \): Represents the number of components in the \( i^{th} \) framework.

\( c_{ij} \): Represents the number of local classes used to construct the \( ij^{th} \) component.

\( c_i \): Represents the number of shared classes in the \( k^{th} \) shared-pool.

\( c_i \): Represents the number of other shared entities in the \( k^{th} \) shared-pool.
$M^*: \text{ Represents the number of local public methods in } l^{th} \text{ class of the } i^{jth} \text{ component.}$

$pm^\text{In}_{ij}: \text{ Represents the number of inputs of the } k^{th} \text{ local public method in the } l^{th} \text{ class of the } i^{jth} \text{ component.}$

$pm^\text{Out}_{ij}: \text{ Represents the number of outputs of the } k^{th} \text{ local public method in the } l^{th} \text{ class of the } i^{jth} \text{ component.}$

$M^\text{In}_{ij}: \text{ Represents the number of public methods inherited by the } l^{th} \text{ class of the } i^{jth} \text{ component.}$

$pm^\text{In}_l^\text{In}_{ij}: \text{ Represents the number of inputs of the } f^{th} \text{ public method that is inherited by the } l^{th} \text{ class of the } i^{jth} \text{ component.}$

$pm^\text{Out}_l^\text{In}_{ij}: \text{ Represents the number of outputs of the } f^{th} \text{ public method that is inherited by the } l^{th} \text{ class of the } i^{jth} \text{ component.}$

$O^\text{In}_{ij}: \text{ Represents the number of objects in the protected section of the } l^{th} \text{ class of the } i^{jth} \text{ component.}$

$M^\text{In}_{ij}: \text{ Represents the number of local protected methods in the } l^{th} \text{ class of the } i^{jth} \text{ component.}$

$rm^\text{In}_{ij}: \text{ Represents the number of inputs of the } f^{th} \text{ local protected method in the } l^{th} \text{ class of the } i^{jth} \text{ component.}$

$rm^\text{Out}_{ij}: \text{ Represents the number of outputs of the } f^{th} \text{ local protected method in the } l^{th} \text{ class of the } i^{jth} \text{ component.}$

$O^\text{Out}_{ij}: \text{ Represents the number of objects in the private section of the } l^{th} \text{ class of the } i^{jth} \text{ component.}$

$M^\text{Out}_{ij}: \text{ Represents the number of local private methods in the } l^{th} \text{ class of the } i^{jth} \text{ component.}$

$vm^\text{In}_l^\text{Out}_{ij}: \text{ Represents the number of inputs of the } f^{th} \text{ local private method in the } l^{th} \text{ class of the } i^{jth} \text{ component.}$

$vm^\text{Out}_l^\text{Out}_{ij}: \text{ Represents the number of outputs of the } f^{th} \text{ local private method in the } l^{th} \text{ class of the } i^{jth} \text{ component.}$

5.2.4.3 Auxiliary Functions

$FrS\_PermittedToAccessSP(k)$: An auxiliary function that returns the set of frameworks that are permitted to access the $k^{th}$ shared-pool.

$One\_to\_One\_UniDirectionalAssociation(X, Y) = \text{ An auxiliary function that returns TRUE if Class X contains an object of Class Y while Y does not contain any object of X.}$

$One\_to\_One\_BiDirectionalAssociation(X, Y) = \text{ An auxiliary function that returns TRUE if Classes X and Y each contains an object of the other.}$

$One\_to\_Many\_UniDirectionalAssociation(X, Y) = \text{ An auxiliary function that returns TRUE if Class X contains a set of objects of Class Y while Y does not contain any object of X.}$
One_to_Many_BiDirectionalAssociation(X, Y) ≡ An auxiliary function that returns TRUE if Class X contains a set of objects of Class Y while Y contains only one object of class X.

Many_to_Many_BiDirectionalAssociation(X, Y) ≡ An auxiliary function that returns TRUE if Classes X and Y each contains multiple objects of the other.

UniDirectionalAssociation(X, Y) ≡ \( \left( \text{One-to-One UniDirectionalAssociation}(X, Y) \lor \text{One-to-Many UniDirectionalAssociation}(X, Y) \right) \): An auxiliary function that returns TRUE if X and Y have a form of uni-directional association with each other.

BiDirectionalAssociation(X, Y) ≡ \( \left( \text{One-to-One BiDirectionalAssociation}(X, Y) \lor \text{One-to-Many BiDirectionalAssociation}(X, Y) \lor \text{Many-to-Many BiDirectionalAssociation}(X, Y) \right) \): An auxiliary function that returns TRUE if X and Y have a form of bi-directional association with each other.

Associates(X, Y) ≡ \( \left( \text{UniDirectionalAssociation}(X, Y) \lor \text{BiDirectionalAssociation}(X, Y) \right) \): An auxiliary function that returns TRUE if classes X and Y associate with each other.

InheritanceCategory(Inherits(X, Y)): An auxiliary function that returns the inheritance category when component X inherits from component Y. The returned value is either “Public”, “Protected”, or “Private”.

Inherits(X, Y): An auxiliary boolean function that returns TRUE if component X does inherit from component Y. Otherwise, it returns FALSE.

Type(X): An auxiliary function that returns the data type of the argument X.

### 5.3 Conclusions

In this chapter, we have shown the following:

1. Rules and guidelines for CO–Toolkits’ design and implementation structures can be formalized using a convenient mathematical terminology and notation (c.f. section 5.2).

2. Such formalized rules for design and implementation structures can be used to build a checking tool. An example prototype tool that we built based on these rules is described in Appendix B.

3. Such checking tools can be used to check CO–Toolkits’ design and implementation structures and report any rule violations back to the developers of
these CO-Toolkits. Examples are provided in Appendixes B2–B4. In particular, an example of ACE inheritance structure is presented in Appendix B4.
CHAPTER 6

THIRD VIEW: FORMALIZING HOW TO DOCUMENT CO–TOOLKITS

6.1 Overview

This chapter describes our documentation view for CO–Toolkits. Documents that are provided with CO–Toolkits are hard to understand. For example, the documents of ACE [43–52] are a set of papers that can help those who are already familiar with ACE. Besides the papers, a lot of comment lines are inserted in the source code of ACE. In such cases, developers may have to start reading through the code itself in order to understand the CO–Toolkit that is under investigation.

We believe that proper documents can help developers understand the CO–Toolkits they are looking at without having to navigate throughout their code. In this chapter, we want to show what kinds of proper documents we think should be provided for CO–Toolkits, what contents should be included in these documents, and how they should be organized. The main documents would be a Component Guide Document (CGD), a Component Interface Specifications Document (CISD), and a Component Internal Design Specifications Document (CIDSD). We expect such documents to maintain the basic properties of the formal documents produced by the Software Engineering Research
Group at McMaster University [4, 7, 36–40, 53]. Appendix C provides sample tables for this third view.

6.2 Component Guide Document

A Component Guide Document (CGD) of a given CO–Toolkit is intended to provide an overview of how files are structured, how components are grouped into frameworks, and how these components are dependent on each other (i.e. the relations among these components such as inheritance and association). In summary, a CGD is an informal document that can be read first. This can help to understand the other provided documents (i.e. the interface specifications and internal design specifications ones).

6.2.1 What Information Should Be Contained In A “Component Guide Document”? And How Should It Be Organized?

As mentioned above, CGDs are informal documents that contain pictorial and textual contexts. Briefly, a CGD of a given CO–Toolkit is expected to contain the following chapters:

(1) A brief description for the given CO–Toolkit and its frameworks and their components.

(2) A brief description for each document that is provided with the given CO–Toolkit.

(3) A brief description associated with a picture that illustrates the structuring of frameworks in the given CO–Toolkit.

(4) A brief description associated with a set of pictures that illustrate the file structures implemented in the given CO–Toolkit. Each picture is expected to illustrate the file structure that is implemented in one framework.
(5) A brief description associated with a set of pictures that illustrate the relations among components of that CO–Toolkit. Each picture is expected to illustrate the inheritance and association relations among the components of the same framework as well as the relations among these components and other classes from the shared pools.

(6) A table that can serve as quick reference for what components are provided by each framework, and what files are used to implement each of these components. Readers can refer to appendix C for a guiding example.

(7) A table that can serve as a quick reference for what services each component can provide. CGDs can be helpful to those who may be interested in CO–Toolkits (i.e. maintainers, users, etc.). By looking at the service–indexed table, a user can easily find out if the services he needs are in that table. If those services are found, then he can easily locate the needed components.

6.2.2 Who Is A “Component Guide Document” intended For?

A CGD of a given CO–Toolkit is intended for maintainers, users, sales, and any other people who are interested in having a quick overview of that CO–Toolkit.

6.2.2.1 Maintainers

A CGD of a given CO–Toolkit can help maintainers to better understand that toolkit. For example, the pictorial views of the file structure, and the relations among components provide a quick and clear overview of the given CO–Toolkit. This reminds us with the motto “a picture is worth a thousand words”. Indexed tables tell maintainers which component provides which service. Also, such tables tell which file of code to look at when it comes to modify a certain component.
6.2.2.2 Users (Developers of CO–Applications)

As already mentioned in this thesis, the word users refers to developers who are interested in integrating components from CO–Toolkits to build their CO–applications. To be able to use a CO–Toolkit, users need to have a clear understanding the conjunction of that toolkit. We believe that the details provided in the CGD (in particular, the indexed table of components) can guide users to choose the components that can use to build their applications.

6.2.2.3 Sales Staff

Sales staff can use some of the pictorial views that are provided in CGDs in their promotional material and advertisement. This requires such pictorial views to be well–organized and clearly presented in the CGD.

6.2.2.4 Technical Support

Technical support staff can understand a given CO–Toolkit by first reading its CGD. Reading CGDs can help technical support staff to better deal with problems raised by customers (i.e. users of CO–Toolkits). In our text, technical support staff are somehow different from maintainers (but still one person can play both roles) since technical support staff are in the middle between maintainers and customers. They understand their customers’ problems, duplicate these problems in house, and then pass them to maintainers. Maintainers in turn, fix the problems and pass their solutions back to customers through technical support staff.

6.2.3 When Should A “Component Guide Document” Be Read?

We think that a CO–Toolkit’s CGD should be read first. This is because a CGD provides various informal information on the different aspects of the given CO–Toolkit.
For example, a CGD illustrates the file structure and the relations among components (i.e. inheritance, association, etc.) in both pictorial and textual forms of context. This can make it easier to understand the general features of the given CO-Toolkit. Also, this can help understand other provided documents with less pain.

### 6.3 Component Interface Specifications Document

A Component Interface Specifications Document (CISD) of a given CO-Toolkit is intended to provide a precise description (specification) for the interface of each component that is provided in that CO-Toolkit.

#### 6.3.1 What Information Should Be Contained In A “Component Interface Specifications Document”? And How Should It Be Organized?

A Component Interface Specifications Document (CISD) can be organized into chapters such that each chapter provides the complete interface specifications for a particular framework from the given CO-Toolkit. Consequently, each of these chapters can be divided into sections such that each section provides the interface specifications for a particular component from that framework. An example for how to make interface specifications for components is provided in Appendix C12. We should acknowledge the fact that this kind of specifications is being investigated at SERG [53–54]. The interface specification for each component is divided into six parts (1) CHARACTERISTICS, (2) SYNTAX (3) CANONICAL REPRESENTATION, (4) EQUIVALENCE, (5) VALUES, and (6) LEXICON. For more details about these parts, readers can refer to Parnas and Mady’s work presented in the “Functional Documentation for Computer Systems Engineering” paper [39].
6.3.2 Who Is A “Component Interface Specifications Document” intended For?

CISDs are mainly intended for users who are interested in building their applications from existing components. The interface specifications describes how to invoke these components.

Maintainers are obligated to keep CISDs updated. Whenever they make any modifications or changes to any piece of code of a given CO-Toolkit, it is essential that they reflect such modifications to all documents of that toolkit including the CISD. Otherwise, documents will loose their correctness and become useless.

Technical support staff need to read through CISDs. This can help them to understand the CO-Toolkits they are dealing with. This also can help them to reproduce problems reported by their customers.

6.3.3 When Should A “Component Interface Specifications Document” Be Read?

We suggest that a CO-Toolkit’s CISD to be read after the CGD.

6.4 Component Internal Design Specifications Document

Similar to CISDs, a CO-Toolkit’s CIDSD is intended to provide a precise description (specification) for the internal design of each component that is provided in that CO-Toolkit.
6.4.1 What Information Should Be Contained In A "Component Internal Design Specifications Document"? And How Should It Be Organized?

Similar to CISDs, CIDSDs can be organized into chapters such that each chapter provides the complete internal design specifications for a particular framework. Consequently, each chapter can be divided into sections such that each section provides the internal design specifications for a particular component. SERG kinds of specifications for internal designs [53–54] can still be used to provide the internal design specifications for components.

6.4.2 Who Is A "Component Internal Design Specifications Document" intended For?

CIDSDs are mainly intended for maintainers. They can use the CIDSD of the given CO-Toolkit to look at the internal design specifications for the component to be fixed. Such internal specifications can help them to locate the code portion that causes the problem. Again, we should mention that it is the obligation of maintainers to keep all documents up to date.

6.4.3 When Should A "Component Internal Design Specifications Document" Be Read?

Since CIDSDs are dealing with a low level of specifications that could be hard to understand, we suggest that a CO-Toolkit’s CIDSD be read after the CGD and CISD of that CO-Toolkit.
6.5 Towards A Mathematical View For CO-Toolkits’ Documentation

6.5.1 Purpose

This view is intended to formalize how to document CO-Toolkits. Also, it is intended to show how NRL documentation methods such those used by Parnas Group at McMaster University [36–40] can be used to document CO-Toolkits.

6.5.2 The View Structure

A CO-Toolkit documentation structure is represented by a tuple of that CO-Toolkit’s documents. Readers can refer to the lexicon section of this view (cf. section 6.5.4) for the meanings of the variables and symbols used this section.

In formula (6–1), a CO-Toolkit is represented by a tuple of the following four elements:

(1) \textit{CGD}: this is used to represent the Component Guide Document as described in section 6.5.2.1.

(2) \textit{CISD}: this is used to represent the Component Interface Specifications Document as described in section 6.5.2.2.

(3) \textit{CIDSD}: this is used to represent the Component Internal Design Specifications Document as described in section 6.5.2.3.

(4) \textit{OtherDocuments}: this is used to accommodate any future expansion to this view whenever there is a need to provide more documents.

\[
\text{COToolkit} = \left[ \text{CGD}, \text{CISD}, \text{CIDSD}, \text{OtherDocuments} \right] \tag{6–1}
\]
6.5.2.1 Explanation of The CGD Document In (6–1) COToolkit

In formula (6–2), the CGD document of (6–1) above is represented by a tuple of the following three elements:

(1) `CGD_DescriptiveOverview`: this element represents the first chapter of the proposed CGD document. This chapter should describe the various aspects of the CO–Toolkit under consideration. For example, it should describe the structure of the CO–Toolkit’s frameworks as well as their components. Also, brief descriptions for the other provided document can be included in this chapter.

(2) `CGD_PictorialOverview`: this element represents the second chapter of the proposed CGD document. This chapter should provide a set of pictorial views that describe several things such as the file structure used for the considered CO–Toolkit, the various relations across the frameworks and among their components (i.e. such as inheritance and association relations), and etc.

(3) `CGD_IndexedQuickReferenceTables`: this element represents the third chapter of the proposed CGD document. This chapter should provide a set of index table that can help users as well as maintainers to find out what services are provided by each component.

\[
CGD = \begin{bmatrix}
CGD\_DescriptiveOverview, \\
CGD\_PictorialOverview, \\
CGD\_IndexedQuickReferenceTables
\end{bmatrix}
\] (6–2)

6.5.2.2 Explanation of The CISD Document In (6–1) COToolkit

In formula (6–3), the CISD document of (6–1) above is represented by a sequence of chapters such that each chapter provides the interface specifications for a certain framework. As explained later on, each of these chapters will be divided into sections such that each section provides the interface specifications for a certain component.

99
Readers can refer to the lexicon of this view (i.e. section 6.5.4) for the meanings of the different variables and symbols used in this view. For example, \( F \) is used to represent the number of frameworks provided in the considered CO-Toolkit.

\[
C_{ISD} = \left( C_{ISD\_Chapter\_Framework} \right)_{i=1}^{\gamma} \tag{6-3}
\]

### 6.5.2.2.1 Explanation of (6-3): \( C_{ISD} \)

In formula (6-4), the interface specifications of the \( i^{th} \) framework (i.e. \( C_{ISD\_Chapter\_Framework} \), of (6-3) above) is represented by a set of sections such that each section provides the interface specifications for a certain component (i.e. \( C_{ISD\_Section\_Component} \)). As explained in the lexicon of this view (i.e. section 6.5.4), \( C_i \) is used to represent the number of Components provided in the \( i^{th} \) framework.

\[
C_{ISD\_Chapter\_Framework} = \left( C_{ISD\_Section\_Component} \right)_{j=1}^{c_i} \tag{6-4}
\]

### 6.5.2.2.2 Explanation of (6-4): \( C_{ISD\_Chapter\_Framework} \)

In formula (6-5), the interface specifications of the \( ij^{th} \) component (i.e. \( C_{ISD\_Section\_Component} \), of (6-4) above) is represented by a tuple of the following six elements. We should notice that these elements are used for the same meanings as in Parnas’s works [36–40].

1. \( C_{ISD\_SubSection\_CHARACTERISTICS} \); as explained in (6-6), this element represents the sub-section that provides the general characteristics of the \( ij^{th} \) component.

2. \( C_{ISD\_SubSection\_SYNTAX} \); as explained in (6-7), this element represents the sub-section that provides the signatures of the access programs of the \( ij^{th} \) component.
(3) \( \text{CISD\_SubSection\_CANONICAL\_REPRESENTATION}_i \): as explained in (6–8), this element represents the sub-section that provides the canonical representation for the \( i^j \text{th} \) component.

(4) \( \text{CISD\_SubSection\_EQUIVALENCES}_i \): as explained in (6–9), this element represents the sub-section that provides the equivalences for the \( i^j \text{th} \) component.

(5) \( \text{CISD\_SubSection\_VALUES}_i \): as explained in (6–10), this element represents the sub-section that provides the values returned by the access programs of the \( i^j \text{th} \) component.

(6) \( \text{CISD\_SubSection\_LEXICON}_i \): this element represents the sub-section that provides the lexicon (i.e. dictionary) that provides the needed clarifications for the symbols, variables, etc. that are used in the interface specifications of the \( i^j \text{th} \) component.

\[
\text{CISD\_Section\_Component}_i = 
\begin{bmatrix}
\text{CISD\_SubSection\_CHARACTERISTICS}_i , \\
\text{CISD\_SubSection\_SYNTAX}_i , \\
\text{CISD\_SubSection\_CANONICAL\_REPRESENTATION}_i , \\
\text{CISD\_SubSection\_EQUIVALENCES}_i , \\
\text{CISD\_SubSection\_VALUES}_i , \\
\text{CISD\_SubSection\_LEXICON}_i 
\end{bmatrix}
\] (6–5)

6.5.2.2.3 Explanation of (6–5): \( \text{CISD\_Section\_Component}_i \)

In formula (6–6), the \( \text{CISD\_SubSection\_CHARACTERISTICS}_i \) of (6–5) above, is represented by a tuple of the following five elements:

(1) \( \text{TypeSpecified}_i \): this element represents the specification type of the \( i^j \text{th} \) component.

(2) \( \text{PublicPrimitiveSystemDataTypes}_i \): this element represents the set of primitive data types (i.e. int, boolean, etc.) that appear in the interface specifications of the \( i^j \text{th} \) component.
(3) **PublicConstantParameters**$_{ij}$: this element represents the set of public constant that appear in the interface specifications of the $ij^{th}$ component.

(4) **PublicSpecificationTypes**$_{ij}$: this element represents the set of types that appear in the interface specifications of the $ij^{th}$ component:

(5) **PublicSpecificationVariables**$_{ij}$: this element represents the set of variables that appear in the interface specifications of the $ij^{th}$ component.

$$
CISD\_SubSection\_CHARACTERISTICS_{ij} \equiv \begin{bmatrix}
\text{TypeSpecified}_{ij}, \\
\text{PublicPrimitiveSystemDataTypes}_{ij}, \\
\text{PublicConstantParameters}_{ij}, \\
\text{PublicSpecificationTypes}_{ij}, \\
\text{PublicSpecificationVariables}_{ij}
\end{bmatrix}
$$

(6-6)

In formula (6-7), the $CISD\_SubSection\_SYNTAX_{ij}$ of (6-5) above, is represented by a tuple of the following two elements:

(1) **Public_ACCESS_ProgramsInComponent**$_{ij}$: this element represents the set of public access programs (methods) that are contained in the $ij^{th}$ component.

(2) **InheritedPublic_ACCESS_ProgramsByComponent**$_{ij}$: this element represents the set of access programs (methods) that are publicly inherited by the $ij^{th}$ component from other components.

$$
CISD\_SubSection\_SYNTAX_{ij} \equiv \begin{bmatrix}
\text{Public_ACCESS_ProgramsInComponent}_{ij}, \\
\text{InheritedPublic_ACCESS_ProgramsByComponent}_{ij}
\end{bmatrix}
$$

(6-7)

In formula (6-8), the $CISD\_SubSection\_CANONICAL\_REPRESENTATION_{ij}$ of (6-5) above, is represented by a tuple of the following two elements:

(1) **CANONICAL\_REPRESENTATION\_OfComponent**$_{ij}$: this element represents the canonical representation for the $ij^{th}$ component. As explained in (6-11), a canonical rep-
representation of a component is composed of the canonical representations of its internal classes.

(2) ReferencesTo_CANONICAL_REPRESENTATIONS_OFInheritedComponents\_ij: this element represents a set of references to the canonical representations of the other components that are inherited from by the \(ij\)th component.

\[
CISD\text{\_SubSection\_CANONICAL\_REPRESENTATION}_i = \\
\begin{bmatrix}
CANONICAL\_REPRESENTATION\_OfComponent\_i, \\
ReferencesTo\_CANONICAL\_REPRESENTATIONS\_OfInheritedComponents\_i
\end{bmatrix}
\] (6–8)

In formula (6–9), the \(CISD\_SubSection\_EQUIVALENCES\_i\) of (6–5) above, is represented by a tuple of the following two elements:

(1) \(EQUIVALENCES\_OfComponent\_i\): this element represents the set of equivalent states to the canonical one. An object of the \(ij\)th component goes into certain one of these equivalent states whenever a certain access program is invoked on that object.

(2) ReferencesTo\_EQUIVALENCES\_OFInheritedComponents\_i: this element represents a set of references to the equivalences of the other components that are inherited from by the \(ij\)th component.

\[
CISD\text{\_SubSection\_EQUIVALENCES}_i = \\
\begin{bmatrix}
EQUIVALENCES\_OfComponent\_i, \\
ReferencesTo\_EQUIVALENCES\_OFInheritedComponents\_i
\end{bmatrix}
\] (6–9)

In formula (6–10), the \(CISD\_SubSection\_VALUES\_i\) of (6–5) above, is represented by a tuple of the following two elements:

(1) \(VALUES\_OfComponent\_i\): this element represents the set of values that are returned by the public access programs (methods) of the \(ij\)th component.
(2) ReferencesTo_VALUES_OFInheritedComponents_{ij}: this element represents a set of references to the values returned by the public access programs (methods) of the other components that are inherited from by the \( i_{j}^{th} \) component.

\[
CISD\_SubSection\_VALUES_{ij} = \begin{bmatrix}
VALUES\_OfComponent_{ij}, \\
ReferencesTo\_VALUES\_OfInheritedACCESS\_Programs_{ij}
\end{bmatrix}
\]  \hspace{1cm} (6-10)

6.5.2.2.4 Explanation of (6-8): \( CISD\_SubSection\_CANONICAL\_REPRESENTATION_{ij} \)

Each component's CANONICAL representation is specified by the sequence of the CANONICAL representations of its internal classes.

\[
CANonical\_REPRESENTATION\_OfComponent_{ij} = \left< \left< CANONICAL\_REPRESENTATION\_OfInternalClass_{ij} \right> \right> \]  \hspace{1cm} (6-11)

6.5.2.3 Explanation of The CIDSD Document In (6-1) COTOolkit

In formula (6-12), the \( CIDSD \) document of (6-1) above is represented by a tuple of the following two sequences of chapters:

(1) \( <CIDSD\_Chapter\_Framework_> : \) this element represents the sequence of chapters that provide the frameworks internal design specifications such that each chapter is dedicated for a certain framework. As explained later on, each framework's chapter will be divided into sections such that each section provides the internal design specifications for a certain component in that framework.

(2) \( <CIDSD\_Chapter\_SharedPool_> : \) this element represents the sequence of chapters that provide the shared pools internal design specifications such that each chapter is dedicated for a certain one of these shared pools. As explained later on, each
shared pool’s chapter will be divided into sections such that each section provides
the internal design specifications for a certain class in that shared pool.

\[
CIDSD \equiv \left[ \prod_{i=1}^{P} (CIDSD_{Chapter\_Framework_i}), \right.
\left. \prod_{k=0}^{S} (CIDSD_{Chapter\_SharedPool_k}) \right]
\] (6–12)

CISDs in (6–3) above do not involve the shared pools, while the internal design
specifications documents, CIDSDs does. The justification for this is that all shared enti-
ties are used internally by the components from the different frameworks and they are
not intended to be directly used by externals.

6.5.2.3.1 Explanation of (6–12): CDISD

In formula (6–13), the internal design specifications of the \(i^{th}\) framework (i.e.
\(CIDSD_{Chapter\_Framework_i}\), of (6–12) above) is represented by a set of sections such that
each section provides the internal design specifications for a certain component (i.e.
\(CIDSD_{Section\_Component_i}\)).

\[
CIDSD_{Chapter\_Framework_i} \equiv \left( \prod_{j=1}^{C_i} (CIDSD_{Section\_Component_j}) \right)
\] (6–13)

In formula (6–14), the internal design specifications of the \(k^{th}\) shared pool (i.e.
\(CIDSD_{Chapter\_SharedPool_k}\), of (6–12) above) is represented by a tuple of the following
two elements:

1. \( (CIDSD_{Section\_SharedClass_j})\): as explained in (6–16), this element represents the
first sequence of the sections of the \(k^{th}\) shared pool chapter. Each of these sections
is dedicated for the internal design specifications of a certain class in that shared
pool. As explained in the lexicon of this view (i.e. section 6.5.4), \( e_1 \) is used to represent the number of shared classes in the \( k^a \) shared pool.

\[
(2) \quad \{CIDSD\_Section\_OtherSharedEntity_{k}\}_{i=0}^{r_1}
\]

this element represents the second sequence of sections of the \( k^a \) shared pool chapter. Each of these sections is dedicated for the internal design specifications of a non-class entity in the \( k^a \) shared pool. As explained in the lexicon of this view (i.e. section 6.5.4), \( e_4 \) is used to represent the number of non-class shared entities in the \( k^a \) shared pool. It is left up to the developers to decide what non-class entities to be included in this chapter.

\[
CIDSD\_Chapter\_SharedPool_{k} = \begin{bmatrix}
    (CIDSD\_Section\_SharedClass_{k})_{i=0}^{r_1} \\
    (CIDSD\_Section\_OtherSharedEntity_{k})_{i=1}
\end{bmatrix}
\]  

(6-14)

6.5.2.3.2 Explanation of (6-13): \( CIDSD\_Section\_Component_{q} \)

In formula (6-15), the internal design specifications of the \( i^a \) component (i.e. \( CIDSD\_Section\_Component_{q} \) of (6-13) above) is represented by a tuple of the following two elements.

(1) \( CIDSD\_SubSection\_Main_{i} \): as explained in (6-17), this element represents the internal design specifications of the main section of the \( i^a \) component. The main section of any component is mainly a straight code that enables users to use the given component as a complete application rather than just invoking some of its classes. Parnas DISPLAY method [40] can be used to document the internal design of components main sections.
(2) \( (CIDS\_SubSection\_InternalClass_i)_j \): as explained in (6-18), this element represents the sequence of sub-sections that provide the internal design specifications of the internal classes of the \( i^a \) component. Each of these sub-sections is dedicated for one of these internal classes. The \( e_j \) is used to represent the number of internal classes in the \( i^a \) component. As explained later on, each internal class is composed of up to three parts: public, protected, and private. The public part can contain only public methods (access programs) that can be invoked by external users. The protected part has mainly protected methods, but still it may contain some protected objects. The private part has mainly private objects (i.e. data structure). The internal design of each of these methods whether public, protected, or even private can be specified "documented" using Parnas DISPLAY method [40] similar to the main section explained above.

\[
CIDS\_Section\_Component_i = \left[ \begin{array}{c}
CIDS\_SubSection\_Main_i \\
CIDS\_SubSection\_InternalClass_i
\end{array} \right]_{i=1}^{e_i} \tag{6-15}
\]

We should notice that the internal design specification of each component are divided into specifications for the main section and specifications for the internal classes of that component. This is consistent with how we viewed the implementation of components in the previous chapter's design and implementation view.

6.5.2.3.3 Explanation of (6-14): CIDS\_Chapter\_SharedPool_i

In formula (6-16), the internal design specifications of the \( k_j^a \) shared class (i.e. \( CIDS\_Section\_SharedClass_i \) of (6-14) above) is represented by a tuple of the following three elements:
(1) \( CIDSD\_SubSection\_SharedClass\_Public_{ij} \): as explained in (6–19), this element represents the sub-section that provides the internal design specifications of the public section of the \( k_j \)th shared class.

(2) \( CIDSD\_SubSection\_SharedClass\_Protected_{ij} \): as explained in (6–20), this element represents the sub-section that provides the internal design specifications of the protected section of the \( k_j \)th shared class.

(3) \( CIDSD\_SubSection\_SharedClass\_Private_{ij} \): as explained in (6–21), this element represents the sub-section that provides the internal design specifications of the private section of the \( k_j \)th shared class.

\[
CIDSD\_Section\_SharedClass_{ij} = \begin{bmatrix}
CIDSD\_SubSection\_SharedClass\_Public_{ij} \\
CIDSD\_SubSection\_SharedClass\_Protected_{ij} \\
CIDSD\_SubSection\_SharedClass\_Private_{ij}
\end{bmatrix}
\]  

(6–16)

### 6.5.2.3.4 Explanation of (6–15): \( CIDSD\_Section\_Component_{ij} \)

In formula (6–17), we show that Parnas DISPLAY method [39] can be used to specify the internal design of each component’s main part.

\[
CIDSD\_SubSection\_Main_{ij} = \left( \frac{\text{NumberOfDISPLAYs\(;\)ForMainPart\(;}\)Component}_{ij}}{i=1} \right)
\]  

(6–17)

In formula (6–18), the internal design specifications of the \( i_t \)th internal class of the \( i_j \)th component (i.e. \( CIDSD\_SubSection\_InternalClass_{ij} \) of (6–15) above) is represented by a tuple of the following three elements:

(1) \( CIDSD\_SubSection\_InternalClass\_Public_{ij} \): as explained in (6–22), this element represents the sub-section that provides the internal design specifications of the public section of the \( i_t \)th internal class of the \( i_j \)th component.
(2) $\text{CIDSD\_SubSection\_InternalClass\_Protected}_i$: as explained in (6–23), this element represents the sub-section that provides the internal design specifications of the protected section of the $i^{th}$ internal class of the $i^{th}$ component.

(3) $\text{CIDSD\_SubSection\_InternalClass\_Private}_i$: as explained in (6–24), this element represents the sub-section that provides the internal design specifications of the private section of the $i^{th}$ internal class of the $i^{th}$ component.

$$\text{CIDSD\_SubSection\_InternalClass}_i \equiv \begin{bmatrix} \text{CIDSD\_SubSection\_InternalClass\_Public}_i \\ \text{CIDSD\_SubSection\_InternalClass\_Protected}_i \\ \text{CIDSD\_SubSection\_InternalClass\_Private}_i \end{bmatrix} \quad (6–18)$$

6.5.2.3.5 Explanation of (6–16): $\text{CIDSD\_Section\_SharedClass}_i$

Parnas DISPLAY method can be used to specify the internal design of shared classes.

$\text{CIDSD\_SubSection\_SharedClass\_Public}_i$: this is similar to (6–22), i.e. public parts \hfill (6–19)

$\text{CIDSD\_SubSection\_SharedClass\_Protected}_i$: this is similar to (6–23), i.e. protected parts \hfill (6–20)

$\text{CIDSD\_SubSection\_SharedClass\_Private}_i$: this is similar to (6–24), i.e. private parts. \hfill (6–21)

6.5.2.3.6 Explanation of (6–18): $\text{CIDSD\_SubSection\_InternalClass}_i$

In formula (6–22), the internal design specifications for the public parts of internal classes can be done using Parnas DISPLAYs method. The internal design specifications for each internal class's public part is represented by

$$\bigcup_{d=1}^{\text{NumberOfDISPLAYsForPublicMethod}_i} \left\{\text{InternalClass\_PublicMethod\_DISPLAY}_d\right\}_{i\in\text{InternalClass\_PublicMethod\_DISPLAY}_i}.$$\hfill (6–18)

This element represents a set of sequences of DISPLAYs. Each of these sequences (i.e. $\left\{\text{InternalClass\_PublicMethod\_DISPLAY}_d\right\}_{i\in\text{InternalClass\_PublicMethod\_DISPLAY}_i}$) represents the subset of Parnas DISPLAYs used to specify the internal design of a certain public method. As per the the lexicon of this view (section 6.5.4), $M^*_i$ represents the
number of public methods in the $i^{th}$ class of the $j^{th}$ component. 

$\text{NumberOfDISPLAYsForPublicMethod}_{ijm}$ represents the number of Parnas DISPLAYs that are required to specify the internal design of the $m^{th}$ public method in the $i^{th}$ class of the $j^{th}$ component.

\[
\text{CIDSD\_SubSection\_InternalClass\_Public}_{ij} = \bigcup_{m=1}^{M_{ij}^{p}} \left\{ \text{NumberOfDISPLAYsForPublicMethod}_{ijm} \right\} \quad (6-22)
\]

In formula (6–23), the internal design specifications for the protected parts of internal classes can be done using Parnas DISPLAYs method similar to the public part above.

\[
\text{CIDSD\_SubSection\_InternalClass\_Protected}_{ij} = \bigcup_{m=0}^{M_{ij}^{p}} \left\{ \text{NumberOfDISPLAYsForProtectedMethod}_{ijm} \right\} \quad (6-23)
\]

In formula (6–24), the internal design specifications for the private parts of internal classes can be done using Parnas DISPLAYs method similar to the public and protected parts above. The internal design specifications for each internal class's private part is represented by a tuple of the following two elements:

\[
(1) \quad \bigcup_{m=0}^{M_{ij}^{p}} \left\{ \text{NumberOfDISPLAYsForPrivateMethod}_{ijm} \right\}: \text{similar to the above, this element represents a set of sequences of DISPLAYs. Each of these sequences (i.e.}
\]

\[
\left( \text{NumberOfDISPLAYsForPrivateMethod}_{ijm} \right) \text{represents the subset of Parnas DISPLAYs used to specify the internal design of a certain private method. As per the the lexicon of this view (section 6.5.4), } M_{ij}^{p} \text{ represents the number of private methods in the } i^{th} \text{ class of the } j^{th} \text{ component. } \text{NumberOfDISPLAYsForPrivateMethod}_{ijm} \text{ rep-}
\]

110
represents the number of Paramas DISPLAYs that are required to specify the internal design of the $m^n$ private method in the $i^n$ class of the $ij^n$ component.

(2) $InternalClass_{DataStructureSpecifications}_{ij}$: this element describes the data structure implemented in the $i^n$ class of the $ij^n$ component. Readers are reminded that a component's data structure is the composition of the data structures of that component's internal classes.

$$CIDSD\_{SubSection}\_{InternalClass}\_{Private}_{ij} = \left[ \bigcup_{n=0}^{N} \left\{ \sum_{d=1}^{\text{NumberOfDISPLAYsForPrivateMethod}_{ij}} InternalClass\_{PrivateMethod}\_{DISPLAY}_{ijd} \right\} \right]$$

(6-24)

6.5.3 The View Characteristics

The following are the main points that characterize this third view:

(1) A CO-Toolkit’s CGD document is an informal document. Therefore, it may contain various information and details about that CO-Toolkit. CO-Toolkits’ Developer are the ones who should decide what to contain in their CGDs.

(2) A CO-Toolkit’s CISD document must address each component in that CO-Toolkit. This is because of that a CO-Toolkit’s CISD reflect how each component should be used.

(3) A CO-Toolkit’s CISD should not address any internal classes of components as well as should not address any shared class from any shared pool. This is because of that a CO-Toolkit's CISD is intended for providing interface specifications and should not reveal and internal specifications.

(4) A CO-Toolkit’s CISD must address the main section and the internal classes of each component in that CO-Toolkit. Also, it must address the shared classes from the different shared pools. This is because of that a CO-Toolkit’s CISD reflects
the internal design of each component as well as the internal design of shared pools' classes and entities.

(5) The various documents (i.e. CGD, CISD, CIDSD, etc.) have to respect the characteristics of the other views that we introduced in the previous two chapters. This is because of that proper documents always reflect their systems' design, implementation, and interface specifications.

6.5.4 The View Lexicon

6.5.4.1 Variables And Acronyms

COToolkit: Represents any Component-Oriented Toolkit that can provide reusable components to be used to build new applications.

CO: Component Oriented.

6.5.4.1.1 CGD’s Variables And Acronyms

CGD: Represents the Component Guide Document as described in section 6.2.

CGD_DescriptiveOverview: Represents the first chapter of the proposed CGD document. This chapter should describe the various aspects of the CO-Toolkit under consideration.

CGD_PictorialOverview: Represents the second chapter of the proposed CGD document. This chapter should provide a set of pictorial views that describe several things such as the file structure used for the considered CO-Toolkit, the various relations across the frameworks and among their components (i.e. such as inheritance and association relations), and etc.

CGD_IndexedQuickReferenceTables: Represents the third chapter of the proposed CGD document. This chapter should provide a set of index table that can help users as well as maintainers to find out what service(s) is (are) provided by each component in the CO-Toolkit under consideration.

6.5.4.1.2 CISD’s Variables And Acronyms

CISD: Represents the Component Interface Specifications Document as described in section 6.3.

CISD_Chapter_Framework_i: Represents the CISD chapter that provides the interface specifications for the i-th framework.

CISD_Section_Component_i: Represents the section dedicated for the interface specifications of i-th component as a part of the CISD chapter for the i-th framework.
CISD_SubSection_CHARACTERISTICS\(i\): Represents the general characteristics of the \(i^{th}\) component.

CISD_SubSection_SYNTAX\(i\): Represents the signatures of the access programs of the \(i^{th}\) component.

CISD_SubSection_CANONICAL_REPRESENTATION\(i\): Represents the canonical representation for the \(i^{th}\) component.

CISD_SubSection_EQUIVALENCES\(i\): Represents the equivalences for the \(i^{th}\) component.

CISD_SubSection_VALUES\(i\): Represents the values returned by the access programs of the \(i^{th}\) component.

CISD_SubSection_LEXICON\(i\): Represents the lexicon for the interface specifications of the \(i^{th}\) component.

TypeSpecified\(i\): Represents the specification type of the \(i^{th}\) component.

PublicPrimitiveSystemDataTypes\(i\): Represents the set of primitive data types (i.e. int, boolean etc.) that appear in the interface specifications of the \(i^{th}\) component.

PublicConstantParameters\(i\): Represents the set of public constant that appear in the interface specifications of the \(i^{th}\) component.

PublicSpecificationTypes\(i\): Represents the set of types that appear in the interface specifications of the \(i^{th}\) component.

PublicSpecificationVariables\(i\): Represents the set of variables that appear in the interface specifications of the \(i^{th}\) component.

Public_ACCESS_ProgramsInComponent\(i\): Represents the set of public access programs (methods) that are contained in the \(i^{th}\) component.

InheritedPublic_ACCESS_ProgramsByComponent\(i\): Represents the set of access programs (methods) that are publicly inherited by the \(i^{th}\) component from other components.

CANONICAL_REPRESENTATION_OfComponent\(i\): Represents the canonical representation for the \(i^{th}\) component.

ReferencesTo_CANONICAL_REPRESENTATIONS_OFInheritedComponents\(i\): Represents a set of references to the canonical representations of the other components that are inherited from by the \(i^{th}\) component.

CANONICAL_REPRESENTATION_OfInternalClass\(i\): Represents the canonical representation for the \(i^{th}\) internal class of the \(i^{th}\) component.

EQUIVALENCES_OfComponent\(i\): Represents the set of equivalent states to the canonical one. An object of the \(i^{th}\) component goes into certain one of these equivalent states whenever a certain access program is invoked on that object.
ReferencesTo_EQUIVALENCES_OFInheritedComponents$q$: Represents a set of references to the equivalences of the other components that are inherited from by the $ij^s$ component.

VALUES_OfComponent$q$: Represents the set of values that are returned by the public access programs (methods) of the $ij^s$ component.

ReferencesTo_VALUES_OFInheritedComponents$q$: Represents a set of references to the values returned by the public access programs (methods) of the other components that are inherited from by the $ij^s$ component.

6.5.4.1.3 CIDSD Variables And Acronyms

CIDSD: Represents the Component Internal Design Specifications Document as described in section 6.4.

CIDSD_Chapter_Framework$q$: Represents the CIDSD chapter that provides the internal design specifications for the $i^s$ framework.

CIDSD_Chapter_SharedPool$k$: Represents the CIDSD chapter that provides the internal design specifications for the $k^s$ shared pool.

CIDSD_Section_Component$q$: Represents the section dedicated for the internal design specifications of $ij^s$ component as a part of the CIDSD chapter for the $i^s$ framework.

CIDSD_Section_SharedClass$k$: Represents the section dedicated for the internal design specifications of the $kj^s$ shared class as a part of the CIDSD chapter for the $k^s$ shared–pool.

CIDSD_Section_OtherSharedEntity$k$: Represents the section dedicated for the internal design of the $kj^s$shared non–class entities as a part of the CIDSD chapter for the $k^s$ shared–pool.

CIDSD_SubSection_Main$q$: Represents the sub-section that provides the internal design specifications of the main part of the $ij^s$ component.

CIDSD_SubSection_InternalClass$q$: Represents the sub-section that provides the internal design specifications for the $l^s$ internal class in the $ij^s$ component.

CIDSD_SubSection_SharedClass_Public$k$: Represents the sub-section that provides the internal design specifications of the public section of the $kj^s$ shared class.

CIDSD_SubSection_SharedClass_Protected$k$: Represents the sub-section that provides the internal design specifications of the protected section of the $kj^s$ shared class.

CIDSD_SubSection_SharedClass_Private$k$: as explained in (6–21), this element represents the internal design specifications of the private section of the $kj^s$ shared class.

CIDSD_SubSection_Main_DISPLAY$d$: Represents the $d^s$ DISPLAY of those used to specify the internal design of the Main part of the $ij^s$ component.

CIDSD_SubSection_InternalClass_Public$q$: Represents the sub-section that provides the internal design specifications of the public section of the $l^s$ internal class of the $ij^s$ component.
CHAPTER 6: THIRD VIEW: FORMALIZING HOW TO DOCUMENT CO-TOOLKITS

**CIDSD_SubSection_InternalClass_Protected**: Represents the sub-section that provides the internal design specifications of the protected section of the \( i \)\textsuperscript{th} internal class of the \( ij \)\textsuperscript{th} component.

**CIDSD_SubSection_InternalClass_Private**: Represents the sub-section that provides the internal design specifications of the private section of the \( i \)\textsuperscript{th} internal class of the \( ij \)\textsuperscript{th} component.

**InternalClass_PublicMethod_DISPLAY**\textsubscript{\( \text{decl} \)}: Represents the \( d \)\textsuperscript{th} Parnas DISPLAY of those required to specify the internal design of the \( m \)\textsuperscript{th} public method in the \( l \)\textsuperscript{th} internal class of the \( ij \)\textsuperscript{th} component.

**ReferencesTo_DISPLAYs_OFPublicMethodsInheritedByInternalClass**\textsubscript{\( \text{incl} \)}: Represents a set of references to the set of Parnas DISPLAYs that provide the internal specifications for those internal class methods that are inherited by the \( l \)\textsuperscript{th} internal class of the \( ij \)\textsuperscript{th} component.

**InternalClass_ProtectedMethod_DISPLAY**\textsubscript{\( \text{impl} \)}: Represents the \( d \)\textsuperscript{th} Parnas DISPLAY of those required to specify the internal design of the \( m \)\textsuperscript{th} protected method in the \( l \)\textsuperscript{th} internal class of the \( ij \)\textsuperscript{th} component.

**InternalClass_PrivateMethod_DISPLAY**\textsubscript{\( \text{incl} \)}: Represents the \( d \)\textsuperscript{th} Parnas DISPLAY of those required to specify the internal design of the \( m \)\textsuperscript{th} private method in the \( l \)\textsuperscript{th} internal class of the \( ij \)\textsuperscript{th} component.

**bldInternalClass_DataStructureSpecifications**\textsubscript{\( \text{incl} \)}: Represents the data structure implemented in the \( l \)\textsuperscript{th} class of the \( ij \)\textsuperscript{th} component.

### 6.5.4.2 Symbols And Constants

\( F \): Represents the number of frameworks in the given CO_toolkit.

\( S \): Represents the number of shared_pools.

\( C_i \): Represents the number of components in the \( i \)\textsuperscript{th} framework.

\( c_k^i \): Represents the number of shared_classes in the \( k \)\textsuperscript{th} shared_pool.

\( e^k_i \): Represents the number of other shared_entities in the \( k \)\textsuperscript{th} shared_pool.

\( c_{ij} \): Represents the number of local_classes used to construct the \( ij \)\textsuperscript{th} component.

\( M_{ij}^{\text{pu}} \): Represents the number of local_public_methods in the \( l \)\textsuperscript{th} class of the \( ij \)\textsuperscript{th} component.

\( M_{ij}^{\text{ps}} \): Represents the number of local_protected_methods in the \( l \)\textsuperscript{th} class of the \( ij \)\textsuperscript{th} component.

\( M_{ij}^{\text{pm}} \): Represents the number of local_private_methods in the \( l \)\textsuperscript{th} class of the \( ij \)\textsuperscript{th} component.

**NumberOfDISPLAYsForMainPartOfComponent**\textsubscript{\( \text{incl} \)}: Represents the number of Parnas DISPLAYs that are required to specify the internal design of the main part of the \( ij \)\textsuperscript{th} component.

**NumberOfDISPLAYsForPublicMethod**\textsubscript{\( \text{incl} \)}: Represents the number of Parnas DISPLAYs that are required to specify the internal design of the \( m \)\textsuperscript{th} public method in the \( l \)\textsuperscript{th} internal class of the \( ij \)\textsuperscript{th} component.

**NumberOfDISPLAYsForProtectedMethod**\textsubscript{\( \text{incl} \)}: Represents the number of Parnas DISPLAYs that are required to specify the internal design of the \( m \)\textsuperscript{th} protected method in the \( l \)\textsuperscript{th} internal class of the \( ij \)\textsuperscript{th} component.

115
**NumberOfDISPLAYsForPrivateMethod**\(_i,j,m,n\): Represents the number of Parnas DISPLAYs that are required to specify the internal design of the \(m^{th}\) private method in the \(i^{th}\) internal class of the \(j^{th}\) component.

### 6.6 Conclusion

In this chapter, we have formalized how to use "table based documentation methods" [7, 39, 40] to document CO-Toolkits.
CHAPTER 7

INITIAL CONDITIONS THAT MAKE THE THREE VIEWS CONSISTENT

7.1 Overview

CO-Toolkits have been viewed in this thesis from three separate perspectives: (i) their file structures (c.f. chapter 4), (ii) their design and implementation (c.f. chapter 5), and (iii) their documentation (c.f. chapter 6). The rules in these three views do not guarantee that the three views are describing the same system. However, they guarantee that it is not impossible that the three views are describing the same system. Satisfying the conditions that are presented in this chapter can improve the possibility of having the three views to be describing the same system. Some of these conditions relate all three views, while some others relate only pairs of views.

This chapter has six sections. The next section presents the conditions that should be satisfied by the three views together. The third section presents the conditions that should be satisfied by the file structure view (i.e. the view presented in chapter 4), and the design and implementation view (i.e. the view presented in chapter 5). The fourth section presents the conditions that should be satisfied by the design and implementation view, and the documentation view (i.e. the view presented in chapter 6). The fifth section
presents the conditions that should be satisfied by the file structure view, and the documentation view. The last section describes this chapter’s conclusions. All of these conditions should be satisfied in order to make the three views consistent.

7.2 Conditions That Relate All Three Views

This section presents the conditions that should be satisfied by the three views at the same time. The following two auxiliary functions are used for convenience in this section in order to maintain simple formulas. Also, these two functions are used in section 7.4.

\[ \text{CARD}(\text{FLAG}, X) = \text{This is an auxiliary function that returns the number of elements in the set } X. \]  
\[ \text{LENGTH}(\text{FLAG}, Y) = \text{This function returns the number of elements in the sequence } Y. \]

<table>
<thead>
<tr>
<th>FLAG</th>
<th>Type of Elements in Set X</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Each element in X is a “Set of Files”.</td>
</tr>
<tr>
<td>2</td>
<td>Each element in X is a “Framework”.</td>
</tr>
<tr>
<td>3</td>
<td>Each element in X is a “Component”.</td>
</tr>
<tr>
<td>4</td>
<td>Each element in X is a “Shared Pool”.</td>
</tr>
<tr>
<td>5</td>
<td>Each element in X is a “Class”</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FLAG</th>
<th>Type of Elements in Sequence Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Each element in Y is a “Chapter”.</td>
</tr>
<tr>
<td>2</td>
<td>Each element in Y is a “Section”.</td>
</tr>
<tr>
<td>3</td>
<td>Each element in Y is a “SubSection”.</td>
</tr>
</tbody>
</table>

(1) Concerning the following formulas:

** Formula (4–1) \( \text{CO} \_{\text{Files}} = \bigcup_{i=1}^{\text{F}} \text{Fr} \_{\text{Files}}_i \bigcup_{k=0}^{\text{S}} \text{Sh} \_{\text{Files}}_k \) from the first view,

** Formula (5–1) \( \text{COToolkit} = \bigcup_{i=1}^{\text{F}} \text{Framework}_i \bigcup_{k=0}^{\text{S}} \text{SharedPool}_k \) from the second view,
Formula (6-3) \( CISD = \left( \bigcup_{i=1}^{F} \text{CISD}_i \right) \) from the third view's CISD, and

** Formula (6-12) \( CIDSD = \left[ \bigcup_{i=1}^{F} \text{CIDSD}_i \right] \) from the third view's CIDSD, the following conditions should be satisfied:

1. \( F \) (i.e., the number of frameworks) should have the same value in the three views. More precisely, the predicate \((P1 \land P2 \land P3)\) should be TRUE, where \(P1\), \(P2\), and \(P3\) are defined next. The justification for requiring this condition is simple. If the three views do not have the same number of frameworks, then either they do not describe the same system, or they fail their attempt to provide a correct description for the same system.

2. \( P1 \equiv \left( \text{CARD}(1, \bigcup_{i=1}^{F} \text{Fr Files}_i) = \text{CARD}(2, \bigcup_{i=1}^{F} \text{Framework}_i) \right) \): This predicate is to ensure that in the number of Frameworks' sets of files in the view for CO-Toolkits' File Structures (i.e., the first view that is presented in chapter 4) is the same as the number of Frameworks in the view for CO-Toolkits' Design and Implementation (i.e., the second view that is presented in chapter 5).

3. \( P2 \equiv \left( \text{CARD}(1, \bigcup_{i=1}^{F} \text{Fr Files}_i) = \text{LENGTH}(1, \bigcup_{i=1}^{F} \text{CISD}_i) \right) \): This predicate is to ensure that the number of Frameworks' sets of files in the first view is the same as the number of Frameworks' Chapters in the CISD document of the view for CO-Toolkits' Documentation (i.e., the third view that is presented in chapter 6).

4. \( P3 \equiv \left( \text{CARD}(1, \bigcup_{i=1}^{F} \text{Fr Files}_i) = \text{LENGTH}(1, \bigcup_{i=1}^{F} \text{CIDSD}_i) \right) \): This predicate is to ensure that the number of Frameworks' sets of files in the first view is the same as the number of Frameworks' Chapters in the CIDSD document of the third view.


ii— $S$ (i.e. the number of shared pools) should have the same value in the three views. More precisely, the predicate ($PS1 \land PS2$) should be TRUE, where $PS1$, and $PS2$ are defined next. The justification for requiring this condition follows from the justification made above for $F$.

- $PS1 = \left( \text{CARD} \left( 1, \bigcup_{k=0}^{s} \text{Sh}\_Files_{k} \right) = \text{CARD} \left( 4, \bigcup_{k=0}^{s} \text{SharedPool}_{k} \right) \right)$: This predicate is to ensure that the number of Shared Pools’ sets of files in the first view (c.f. chapter 4) is the same as the number of Shared Pools in the second view (c.f. chapter 5).

- $PS2 = \left( \text{CARD} \left( 1, \bigcup_{k=0}^{s} \text{Sh}\_Files_{k} \right) = \text{LENGTH} \left( 1, \{ \text{CIDSD}\_Chapter\_SharedPool_{k} \} \right) \right)$:

This predicate is to ensure that the number of Shared Pools’ sets of files in the first view is the same as the number of Shared Pools’ Chapters in the CIDSD document of the third view (c.f. chapter 6).

(2) Concerning the following formulas:

** Formula (4–2) $Fr\_Files_{i} = \bigcup_{j=1}^{c_{i}} C\_Files_{j}$ from the first view,

** Formula (5–2) $Framework_{i} = \bigcup_{j=1}^{c_{i}} Component_{j}$ from the second view,

** Formula (6–4) $CISD\_Chapter\_Framework_{i} = \left( CISD\_Section\_Component_{j} \right)$ from the third view’s CISD, and

** Formula (6–13) $CISD\_Chapter\_Framework_{i} = \left( CISD\_Section\_Component_{j} \right)$ from the third view’s CIDSD, the following condition should be satisfied:

- For all $i$, $C_{i}$ that represent the number of components in the $i^{th}$ framework, should have the same value in the three views. More precisely, the predicate $(\forall i, (1 \leq i \leq F) \Rightarrow (PC1 \land PC2 \land PC3))$ should be TRUE, where $PC1$, $PC2$, and
PC3 are defined next. The justification for requiring this condition follows from the justification above for requiring to have the same value for $F$ in the three views.

\[- \quad PC1 \equiv \left( \text{CARD} \left( 1, \bigcup_{j=1}^{c_i} C_{Files_j} \right) = \text{CARD} \left( 3, \bigcup_{j=1}^{c_i} Component_j \right) \right) : \text{This predicate is to ensure that the number of Components' sets of files of the } i^{th} \text{ Framework in the first view (c.f. chapter 4) is the same as the number of Components in the } i^{th} \text{ Framework in the second view (c.f. chapter 5)} \]

\[- \quad PC2 \equiv \left( \text{CARD} \left( 1, \bigcup_{j=1}^{c_i} C_{Files_j} \right) = \text{LENGTH} \left( 2, \left( \text{CISD\_Section\_Component}_j \right) \right) \right) : \text{This predicate is to ensure that the number of Components' sets of files of the } i^{th} \text{ Framework in the first view is the same as the number of Components' Sections in the } i^{th} \text{ Framework's Chapter in the CISD document of the third view (c.f. chapter 6)} \]

\[- \quad PC3 \equiv \left( \text{CARD} \left( 1, \bigcup_{j=1}^{c_i} C_{Files_j} \right) = \text{LENGTH} \left( 2, \left( \text{CIDSD\_Section\_Component}_j \right) \right) \right) : \text{This predicate is to ensure that the number of Components' sets of files of the } i^{th} \text{ Framework in the first view is the same as the number of Components' Sections in the } i^{th} \text{ Framework's Chapter in the CIDSD document of the third view (c.f. chapter 6)} \]

### 7.3 Conditions That Relate The First Two Views

This section presents the conditions (i.e. in addition to those in section 7.2) that should be satisfied at the same time by the first two views (c.f. chapters 4, and 5). These initial conditions are required in order to make these two views consistent. The following auxiliary predicate is used for convenience in order to simplify the formulas that are presented in this section.

$UsedToImplement(X,Y) \equiv \text{This is an auxiliary predicate that returns TRUE if the Source Code of the Sub-System } Y \text{ is provided in the Set of Files } X.$
(1) Concerning formula (4–1) (i.e. $CO_{Files} = \bigcup_{i=1}^{F} Fr_{Files_i} \cup \bigcup_{k=0}^{S} Sh_{Files_k}$) from the first view, and formula (5–1) (i.e.$CO_{Toolkit} = \bigcup_{i=1}^{F} Framework_i \cup \bigcup_{k=0}^{S} SharedPool_k$) from the second view, the following conditions should be satisfied:

\[ i - \forall i, (1 \leq i \leq F) \Rightarrow (UsedToImplement(Fr_{Files_i}, Framework_i) = \text{TRUE}) \] This predicate is to ensure that each Framework has been implemented in its own set of files.

\[ ii - \forall k, (1 \leq i \leq S) \Rightarrow (UsedToImplement(Sh_{Files_k}, SharedPool_k) = \text{TRUE}) \] This predicate is to ensure that each Share Pool has been implemented in its own set of files.

(2) Concerning formula (4–2) (i.e. $Fr_{Files_i} = \bigcup_{j=1}^{C_i} C_{Files_j}$) from the first view, and formula (5–2) (i.e. $Framework_i = \bigcup_{j=1}^{C_i} Component_j$) from the second view, the following condition should be satisfied:

\[ i - \forall i, (1 \leq i \leq F) \Rightarrow \forall j, (1 \leq j \leq C_i) \Rightarrow (UsedToImplement(C_{Files_j}, Component_j) = \text{TRUE}) \] This predicate is to ensure that each component has been implemented in its own set of files.

(3) Concerning formula (4–4) (i.e. $C_{Files_i} = [topFile_i, intermediateFile_i, internalFile_i]$) from the first view, and formula (5–4) $Component_i = [Interface_i, Implementation_i]$ and its sub-formulas (5–5) through (5–17) from the second view, the following conditions should be satisfied:

\[ i - \forall i, (1 \leq i \leq F) \Rightarrow \forall j, (1 \leq j \leq C_i) \Rightarrow (UsedToImplement(topFile_i, MainPart_j) = \text{TRUE}) \] This predicate is to ensure that the implementation of the main part of each component is provided in the top file of that component. The term $MainPart_i$ is addressed in the sub-formula (5–10) of chapter 5.

\[ ii - \forall i, (1 \leq i \leq F) \Rightarrow \forall j, (1 \leq j \leq C_i) \Rightarrow (UsedToImplement(InternalFile_i, Interface_i) = \text{TRUE}) \] This predicate is to ensure that the interface of each component is provided in the intermediate file of that component.
iii - \( \forall i, (1 \leq i \leq F) \Rightarrow \forall j, (1 \leq j \leq C_i) \Rightarrow (\text{UsedToImplement}(\text{InternalFile}_i, \text{Implementation}_j) = \text{TRUE}) \)

This predicate is to ensure that the implementation of each component is provided in the internal file of that component.

### 7.4 Conditions That Relate The Second And Third Views

This section presents the conditions (i.e. in addition to those in section 7.2) that should be satisfied at the same time by the second view (c.f. chapter 5) and third view (c.f. chapter 6). These initial conditions are required in order to make these two views consistent.

There are three documents considered in the third view (i.e. CGD, CISD, and CIDSD). We are concerned about the conditions that relate the second view and the third view's CISD and CIDSD documents since the CGD document is an informal one.

#### 7.4.1 Conditions That Relate The Second View And Third View's CISD Document

This section presents the conditions (i.e. in addition to those in section 7.2) that should be satisfied at the same time by the second view (c.f. chapter 5), and the CISD document of the documentation view (c.f. chapter 6: section 6.5.2.2). The following auxiliary predicate is used for convenience throughout this sub-section in order to maintain simple formulas.

\( \text{ProvidesInterfaceSpecifications}(X, Y) = \text{This is an auxiliary predicate that returns TRUE if the interface specifications of the Sub-System Y are provided in the portion X of the CISD document.} \)

(1) Concerning formula (5-1) (i.e. \( COToolkit = \bigcup_{i=1}^{F} \text{Framework}_i \bigcup \bigcup_{k=0}^{S} \text{SharedPool}_k \)) from the second view, and formula (6-3) (i.e. \( \text{CISD} \equiv \langle \text{CISD}_ Chapter \ Framework \rangle \)) from the third view's CISD, the following condition should be satisfied:
i— \( \forall i, (1 \leq i \leq F) \Rightarrow (\text{ProvidesInterfaceSpecifications} (\text{CISD Chapter Framework}_i, \text{Framework}_i) = \text{TRUE}) \):

This predicate is to ensure that the interface specifications for each framework is provided in the CISD' Chapter that is dedicated for that framework.

(2) Concerning formula (5–2) (i.e. \( \text{Framework}_i = \bigcup_{j=1}^{c_i} \text{Component}_j \)) from the second view and the formula (6–4) (i.e. \( \text{CISD Chapter Framework}_i = (\bigcup_{j=1}^{c_i} \text{CISD Section Component}_j) \)) from the third view's CISD, the following condition should be satisfied:

\[ i— \forall i, (1 \leq i \leq F) \Rightarrow \forall j, (1 \leq j \leq c_i) \Rightarrow \]

\( (\text{ProvidesInterfaceSpecifications} (\text{CISD Section Component}_j, \text{Component}_j) = \text{TRUE}) \):

This predicate is to ensure that the interface specification for each component is provided in the CISD' Section that is dedicated for that component.

7.4.2 Conditions That Relate The Second View And Third View's CIDSD Document

This section presents the conditions (i.e. in addition to those in section 7.2) that should be satisfied at the same time by the second view (c.f. chapter 5) and third view's CIDSD document (c.f. chapter 6, section 6.5.2.3.1). The following auxiliary predicate is used (i.e. in addition to those two functions in section 7.2) for convenience throughout this sub-section in order to maintain simple formulas.

\( \text{ProvidesDesignSpecifications} (X, Y) \Rightarrow \) This is an auxiliary predicate that returns TRUE if the design and implementation specifications of the Sub–System Y are provided in the portion X of the CIDSD document.

(1) Concerning the following formulas:

** Formula (5–10) \( \text{InternalClassesSet}_Y = \bigcup_{i=1}^{c_Y} \text{InternalClass}_Y \) from the second view, and
** Formula (6–15) \( CIDSD\_Section\_Component_q = \left[ \begin{array}{c} CIDSD\_SubSection\_Main_q, \\
\sum_{i=1}^{e_i} \langle CIDSD\_SubSection\_InternalClass_i \rangle 
\end{array} \right] \) from

the third view's CIDSD, the following condition should be satisfied:

\( e_q \) that represents the number of internal classes in the \( i \)-th component, should have the same value in the second view and the third view's CIDSD documents. More precisely, the next predicate should be TRUE. The justification for this is that the number of internal classes in each component should be the same as the number of CIDSD's Sub-Sections that are dedicated for that component.

\[ \forall i, (1 \leq i \leq F) \Rightarrow \forall j, (1 \leq j \leq C_i) \Rightarrow
\begin{align*}
\text{CARD}(6, InternalClassSet_q) &= \text{LENGTH}\left(3, \sum_{i=1}^{e_q} \langle CIDSD\_SubSection\_InternalClass_i \rangle \right)
\end{align*}

(2) Concerning the following formulas:

** Formula (5–3) \( SharedPool_k = \left[ \begin{array}{c}
\text{SharedClassesSet}_k, \\
\text{OtherSharedEntitiesSet}_k
\end{array} \right] \), where

\[ \text{SharedClassesSet}_k = \bigcup_{j=0}^{e_k} \text{SharedClass}_{k_j}, \text{ and } \text{OtherSharedEntitiesSet}_k = \bigcup_{j=1}^{e_k} \text{SharedEntity}_{k_j} \]

from the second view, and

** Formula (6–14) \( CIDSD\_Chapter\_SharedPool_k = \left[ \begin{array}{c}
\langle CIDSD\_Section\_SharedClass_{k_j} \rangle, \\
\langle CIDSD\_Section\_OtherSharedEntity_{k_j} \rangle
\end{array} \right] \) from

the third view's CIDSD, the following condition should be satisfied:

\( e_k \) that represents the number of shared classes in the \( k \)-th shared-pool should have the same value in both the second view and the third view's CIDSD document. More precisely, the next predicate should be TRUE. The justification for this is that the number of shared classes in each Shared Pool should be the same as the number of CIDSD's
Sections that are dedicated for the shared classes of that Shared Pool.

\[ \forall k, (0 \leq k \leq S) \Rightarrow \]
\[ \left[ \text{CARD} \left( S, \bigcup_{j=0}^{c_i^k} \text{SharedClass}_k \right) \right] = \text{LENGTH} \left( 2, \left( \text{CIDSD Section SharedClass}_k \right) \right) \]

(3) Concerning formula (5-1) (i.e. \( COToolkit = \bigcup_{i=1}^{p} \text{Framework}_i \bigcup_{k=0}^{S} \text{SharedPool}_k \)) from the second view, and formula (6-12) (i.e. \( CIDSD = \left[ \left( \text{CIDSD Chapter Framework}_k \right) \bigcup_{i=1}^{p} \bigcup_{s=0}^{S} \right] \)) from the third view’s CIDSD, the following conditions should be satisfied:

\( i \quad \forall i, (1 \leq i \leq F) \Rightarrow \)

\( \left( \text{ProvidesDesignSpecifications}(\text{CIDSD Chapter Framework}_k, \text{Framework}_i) = \text{TRUE} \right) \):

This predicate is to ensure that the design specifications for each framework is provided in the CIDSD’ Chapter that is dedicated for that Framework.

\( ii \quad \forall k, (1 \leq i \leq S) \Rightarrow \)

\( \left( \text{ProvidesDesignSpecifications}(\text{CIDSD SharedPool}_k, \text{SharedPool}_k) = \text{TRUE} \right) \):

This predicate is to ensure that the design specifications for each Shared Pool is provided in the CIDSD’ Chapter that is dedicated for that Shared Pool.

(4) Concerning formula (5-2) (i.e. \( \text{Framework}_i \bigcup_{j=1}^{c_i} \text{Component}_j \)) from the second view, and formula (6-13) (i.e. \( \text{CIDSD Chapter Framework}_i = \left( \text{CIDSD Section Component}_j \right) \)) from the third view’s CIDSD, the following condition should be satisfied:

\( i \quad \forall i, (1 \leq i \leq F) \Rightarrow \forall j, (1 \leq j \leq c_i) \Rightarrow \)

\( \left( \text{ProvidesDesignSpecifications}(\text{CIDSD Section Component}_j, \text{Component}_j) = \text{TRUE} \right) \):

This predicate is to ensure that the design specifications for each component is provided in the CIDSD’ Section that is dedicated for that component.
(5) Concerning formula (5–10) (i.e. $\text{Implementation}_q = \begin{bmatrix} \text{MainPart}_q, \\ \text{InternalClassesSet}_q \end{bmatrix}$) where

$\text{InternalClassesSet}_q = \bigcup_{i=1}^{c_q} \text{InternalClass}_{q_i}$) from the second view, and formula (6–15) (i.e. $\text{CIDSD\_Section\_Component}_q = \begin{bmatrix} \text{CIDSD\_SubSection\_Main}_q, \\ \left(\bigcup_{i=1}^{c_q} \text{CIDSD\_SubSection\_InternalClass}_{q_i} \right) \end{bmatrix}$) from the third view’s CIDSD document, the following conditions should be satisfied:

$i-$ $\forall i, (1 \leq i \leq F) \Rightarrow \forall j, (1 \leq j \leq C_i) \Rightarrow$

(ProvidesDesignSpecifications($\text{CIDSD\_SubSection\_Main}_q, \text{MainPart}_i$) = TRUE):

This predicate is to ensure that the design specifications for each component’s main part is provided in the CIDSD’ Sub–Section that is dedicated for that main part.

$ii-$ $\forall i, (1 \leq i \leq F) \Rightarrow \forall j, (1 \leq j \leq C_i) \Rightarrow$

(ProvidesDesignSpecifications($\text{CIDSD\_SubSection\_InternalClass}_q, \text{InternalClass}_{q_j}$) = TRUE):

This predicate is to ensure that the design specifications for each internal class is provided in the CIDSD’ Sub–Section that is dedicated for that internal class.

### 7.5 Conditions That Relate The First And Third Views

The conditions that need to be satisfied in order to ensure the consistency of the file structure view (i.e. the first view presented in chapter 4), and the documentation view (i.e. the third view presented in chapter 6) are:

(1) Satisfying the conditions that relate the three views (c.f. section 7.2).

(2) Satisfying the conditions that relate the first two views (c.f. see section 7.3).

(3) Satisfying the conditions that relate the second and the third views (c.f. sections 7.4, 7.4.1, and 7.4.2).
7.6 Conclusions

In this chapter, we have presented the conditions that should be satisfied by either the three views or by only pairs of these views at the same time. These conditions are required in order to make these three views consistent.

(1) Section 7.2 presents the conditions that should be satisfied by the three views together.

(2) Section 7.3 presents the conditions that should be satisfied by the file structure view and the implementation view.

(3) Section 7.4 presents the conditions that should be satisfied by the implementation view and the documentation view.

(4) Section 7.5 presents the conditions that should be satisfied by the file structure view and the documentation view.
CHAPTER 8

LIMITATIONS, CONCLUSIONS, AND FUTURE WORK

8.1 Overview

This chapter has four sections. The next section describes the limitations of our work. The third section describes the main conclusions about this research. The last section describes future work that can potentially improve and extend the outcome of this research.

8.2 Limitations Of This Research

The limitations of this research can be summarized as follows:

1. In the view for CO-Toolkits' file structures (c.f. chapter 4), we limited the file hierarchy within each framework to only three levels: top, intermediate, and internal. Although, this can be seen as a restriction, we think that this restriction makes it easier for us to formalize the set of rules for file structures that we derived based on the assumptions for good design practice (c.f. sections 3.3 and 3.3.2).
(2) Given that our prototype systems were intended to just demonstrate that formalized rules can be used to build tools, we did not provide any front ends for these prototypes just to keep them simple. Also, we think that existing front ends such DATRIX from Bell Canada can be customized and integrated with our prototypes.

8.3 Conclusions

By this research, we have shown the following:

(1) Rules for organizing, designing, and implementing CO–Toolkits can be derived based on a set of initial assumptions for good design practice (c.f. chapter 3).

(2) These rules can be formalized using an appropriate mathematical terminology and notation (c.f. chapters 4, 5, and 6).

(3) These formalized rules can be used to develop tools (c.f. Appendixes A, and B).

(4) Such tools can be used to check CO–Toolkits’ code against these formalized rules and report any violations back to developers (c.f. Appendixes A2–A4, and B2–B4).

(5) Our views are intended to be general and language–independent. However, the following things need to be considered when using our views:

- The view for file structures is good for languages that support the #include directive such as C, C++. For languages that do not support such directive (i.e. JAVA, FORTRAN, etc.), some of the rules needs to be modified to fit the equivalent directive (i.e. if there is any).
ii– The view for implementation structures is good for object-oriented languages. For example, it can not be used to check code that is written in C.

iii– Given that the documentation view is focusing on the interface specifications and the internal design specifications, then it can be used to document code that is written in any language.

(6) Experience shows that there is no other place that tried to make formal statement of rules for good design.

(7) Given that we are looking at a large set of badly defined artifacts and trying to find general rules, there may be no simple general rules in our field – unlike nature – because languages are often poorly designed.

8.4 Future Work

This section describes future works that can potentially extend this research:

(1) As our formal models are not complete, further work can be conducted in the future to extend and improve the set of rules presented in this thesis.

(2) Investigating the possibility of simplifying the mathematical notation and terminology that we used to formalize our rules described in chapters 4 through 6.

(3) Investigating the integration of existing front ends such as the DATRIX tool from Bell Canada into our prototypes.

(4) Investigating the use of design patterns in documenting CO–Toolkits. According to Fayad and Schmidt [19], "Patterns represent recurring solutions to software development problems within a particular context".
(5) In [28], Lakos provided a set of arbitrary and language-dependent rules that address components construction. Investigating how to formalize these rules can be performed in further research.
APPENDIX A

A PROTOTYPE FOR CO-TOOLKITS' FILE STRUCTURES

(1) Appendix A1: A Prototype For CO-Toolkits' File Structures:
   An Overview --------------------------------------------- P135–188

(2) Appendix A2: A Prototype For CO-Toolkits' File Structures:
   Example#1 – An Artificial Example ---------------------- P189–192

(3) Appendix A3: A Prototype For CO-Toolkits' File Structures:
   Example#2 – An Artificial Example — --------------------- P193–206

(4) Appendix A4: A Prototype For CO-Toolkits' File Structures:
   Example#3 – ACE Sample File Structure ------------------ P207–220
Appendix A1

A Prototype For CO–Toolkits’s File Structures: An Overview

A1.1 Prototype Description

Java has been used to build this prototype based on the view for CO–Toolkits’ file structures (c.f. chapter 4). The prototype’s data structure represents the view structure of section 4.2.2. Each predicate from the view’s characteristics section (c.f. section 4.2.3) is implemented as a method that returns a boolean value (i.e. TRUE or FALSE). For convenience, we named each of these methods by its formula number as in chapter 4.

Given that the purpose of building this prototype is to show that this view can be used to build a tool, we kept this prototype simple. All of its inputs are built–in. Currently, three built–in examples are provided, one of them is on ACE file structure (c.f. Appendixes A2–A4).

To run this prototype:

1. The twelve files listed in the next section should be in a directory named V1_FileStructure_R3. In Java, path and file names are case sensitive.
2. Run the command `javac –deprecation V1_FileStructure_R3/*.java`
3. Run the command `java V1_FileStructure_R3.CheckFileStructure N`
   N is the argument. It represents the example number of choice (i.e. 1 for example 1, 2 for example 2, and 3 for the ACE example).
The prototype operation goes through the following steps:

(1) It allocates its internal data structure based on the view structure of section 4.2.2. To perform this step, it requires the number of frameworks (i.e. \( F \)), the number of shared pools (i.e. \( S \)), the number of components in each framework (i.e. \( C[I] \)), and the number of shared files in each shared pool (\( SC[I] \)).

(2) It updates its internal data structures with the file names and their extensions. For now, we use .top for top files, .inm for intermediate files, and .inl for internal files.

(3) It updates its internal data structure with the file "#include" relations.

(4) It checks its internal data structure by sequentially invoking its methods that represent the various predicate (i.e. rules) in the view characteristic section (c.f. section 4.2.3). Each of these methods reports any violation to its embedded rule.

A1.2 Source Code

A1.2.1 CheckFileStructure.java

This file contains the implementation of all predicates that represent the characteristics of the File Structure’s View. These predicates are named by their formula numbers as in chapter 4. For example, predicate4–20 is the implementation of characteristic (4–20).

Next is the source code listing for this file:

```java
package V1_FileStructure_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#1 For CO_Toolkits File Structures — Chapter 4 of My Ph.D. Thesis
 */
public class CheckFileStructure_R3 {
```
private static int Ptrl=0;
private static boolean LoopOccurred=false;
private static boolean MultiplePathOccurred=false;
private static boolean MatrixExtracted=false;
private static int[][] InclusionMatrix= new int[((Globals.MAX_SP-1) * (Globals.MAX_SF-1)) + ((Globals.MAX_FR-1) * (Globals.MAX_CO-1)) + 1][][][((Globals.MAX_SP-1) * (Globals.MAX_CO-1)) + 1][][][((Globals.MAX_FR-1) * (Globals.MAX_CO-1)) + 1];
private static int M; // for the InclusionMatrix dimension [MxM]
private static tmpFileNode[] ListFiles= new tmpFileNode[(((Globals.MAX_FR-1)*(Globals.MAX_CO-1))*(Globals.MAX_SF-1)) + ((Globals.MAX_SP-1)*(Globals.MAX_SF-1)) + 1];
private static int[] ListNodes= new int[(((Globals.MAX_FR-1)*(Globals.MAX_CO-1))*(Globals.MAX_SF-1)) + 1];
private static int[] Path1 = new int[(((Globals.MAX_FR-1)*(Globals.MAX_CO-1))*(Globals.MAX_SF-1)) + 1];
private static int[] Path2 = new int[(((Globals.MAX_FR-1)*(Globals.MAX_CO-1))*(Globals.MAX_SF-1)) + 1];
private static int Ptrl=0;
private static int Ptrl-2=0;
private static int StartNode;
private static int LastNode;
private static int recpt=0; private static String output;
private static String violation; public CheckFileStructure_R3() {
    super();
}

protected static void displayFileStructure() {
    output = "FILES STRUCTURE: As Extracted From The Current Example";
    FileStructure.reportFileStructure(output);
    FileStructure.display();
    output = "FILE STRUCTURE display is DONE";
    FileStructure.reportFileStructure(output);
    output = "="
    FileStructure.reportFileStructure(output);
}

private static void displayInclusionMatrix() {
    output = "INCLUSION MATRIX: Coordinates (x,y) represent FileNodeNumbers x and y\n";
    output = output + "\n";
    for (int i=1; i<=M; i++){
        output = output + "\n" + i;
    }
    output = output + "\n";
    for (int i=1; i<=M; i++){
        output = output + i + "\n";
    }
    for (int j=1; j<=M; j++){
        output = output + j + "\n" + InclusionMatrix[i][j];
    }
    output = output + "\n";
}
output=output+"__________

reportInclusionMatrix(output);
}

private static void displayListNodes() {
    output="n"
    output=output+"ListNodes[]:"
    for (int i=1; i<Ptr; i++) {
        output=output+ "u" + ListNodes[i];
    }
    output=output+"n"
    reportOutput(output);
}

private static void displayLoop (int listSize) {
    output="n\nVIOLATION(Predicate4_13): FileInclusionLOOP has been detected: \n"
    int lp=1;
    output=output+"u"
    while (lp < listSize) {
        output=output+ListNodes[lp]+ " l-->
        lp=lp+1;
    }
    output=output+ ListNodes[lp];
    output=output+ "n\nDETAILS:\n"
        for (int lpp=1; lpp<=listSize; lpp++){
            output=output+"\nNodeNumber("+ListNodes[lpp]+")u"
                "FileName("+ListFiles[ListNodes[lpp]].Name+":")+ListFiles[ListNodes[lpp]].Type+"l"
                "Framework("+ListFiles[ListNodes[lpp]].FR_Number+ ")l"
                "Component("+ListFiles[ListNodes[lpp]].FR_Number+",")+ListFiles[ListNodes[lpp]].CO_Number+"
        }
    reportViolation(output);
    return;
}

private static void displayMultiplePath (int Path1Size, int Path2Size) {
    output="n\nVIOLATION(Predicate4_12:OnlyIntermediateLevelMayHaveMultiplePaths) has been detected: ",
    int lp1=1;
    output=output+"n\n"
    while (lp1 < Path1Size ) {
        output=output+Path1[lp1]+ " l-->
        lp1=lp1+1;
    }
    output=output+Path1[lp1];
    int lp2=1;
    output=output+"n\n"
    while (lp2 < Path2Size ) {
        output=output+Path2[lp2]+ " l-->
        lp2=lp2+1;
    }
}
 Appendix A1: A Prototype For CO-Toolkits’s File Structures: An Overview

```java
output = output + Path2[i][2];
reportViolation(output);
return;
}
private static void displayPath1()
{
    output = "\nPath1[]:"
    for (int i = 1; i <= Ptrl; i++)
    {
        output = output + "\t" + Path1[i];
    }
    output = output + "\n"
    reportOutput(output);
}
private static void displayPath2()
{
    output = "\nPath2[]:"
    for (int i = 1; i <= Ptrl2; i++)
    {
        output = output + "\t" + Path2[i];
    }
    output = output + "\n"
    reportOutput(output);
}
private static void example_1()
{
    FileStructure.initializeFor_Example_1();
}
private static void example_1_OLD()
{
    FileStructure.initializeFor_Example_1();
}
private static void example_2()
{
    FileStructure.initializeFor_Example_2();
}
private static void example_3_ACE()
{
    FileStructure.initializeFor_Example_3();
}
private static void extractInclusionMatrix()
{
    // Calculate M: the dimension of the InclusionMatrix[MxM]
    M = 0;
    for (int i = 1; i <= FileStructure.F; i++)
    {
        M += 3 * FileStructure.C[i];
    }
    int ExtM = 0;
    for (int i = 1; i <= FileStructure.S; i++)
    {
        ExtM = ExtM + FileStructure.SF[i];
    }
    M += ExtM; int tmpCount = 0;
    for (int i = 1; i <= FileStructure.F; i++)
    {
        for (int j = 1; j <= FileStructure.C[i]; j++)
        {
            for (int k = 1; k <= 3; k++)
            {
```
int tmpFileNodeNumber = FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].ID.FileNodeNumber;
ListFiles[++tmpCount] = new tmpFileNode();
ListFiles[tmpCount].NodeNumber = FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].ID.FileNodeNumber;
ListFiles[tmpCount].Name = FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].Name;
ListFiles[tmpCount].Type = FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].Type;
ListFiles[tmpCount].FR_Number = i;
ListFiles[tmpCount].CO_Number = j;
int tmp = 0;
while (FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].IncludeList[++tmp] > 0) {
    int tmpIncludedFileNodeNumber = FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].IncludeList[tmp];
    if (tmpIncludedFileNodeNumber <= M) {
        InclusionMatrix[tmpFileNodeNumber][tmpIncludedFileNodeNumber] = 1;
    } else {
        String tmpFN = FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].Name + "." + FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].Type;
        System.out.println("ERROR(Predicate4_13): File " + tmpFN + " with FileNode(" + tmpFileNodeNumber + ") from Framework(" + i + ") and Component(" + i + "," + j + ") includes FileNode(" + tmpIncludedFileNodeNumber + ") that has no existing file");
    }
}
}

for (int i = 1; i <= FileStructure.S; i++) {
    for (int j = 1; j <= FileStructure.SF[i]; j++) {
        int tmpFileNodeNumber = FileStructure.CO_Sh_Files[i].SharedFileList[j].ID.FileNodeNumber;
        ListFiles[++tmpCount] = new tmpFileNode();
        ListFiles[tmpCount].NodeNumber = FileStructure.CO_Sh_Files[i].SharedFileList[j].ID.FileNodeNumber;
        ListFiles[tmpCount].Name = FileStructure.CO_Sh_Files[i].SharedFileList[j].Name;
        ListFiles[tmpCount].Type = FileStructure.CO_Sh_Files[i].SharedFileList[j].Type;
        int tmp = 0;
        while (FileStructure.CO_Sh_Files[i].SharedFileList[j].IncludeList[++tmp] > 0) {
            int tmpIncludedFileNodeNumber = FileStructure.CO_Sh_Files[i].SharedFileList[j].IncludeList[tmp];
            if (tmpIncludedFileNodeNumber <= M) {
                InclusionMatrix[tmpFileNodeNumber][tmpIncludedFileNodeNumber] = 1;
            } else {
                String tmpFN = FileStructure.CO_Sh_Files[i].SharedFileList[j].Name + "." + FileStructure.CO_Sh_Files[i].SharedFileList[j].Type;
                System.out.println("ERROR(Predicate4_13): SharedFile " + tmpFN + " with FileNode(" + tmpFileNodeNumber + ") from SharedPool(" + i + ") includes FileNode(" + tmpIncludedFileNodeNumber + ") that has

140
no existing file
 System.out.println("\nThis illegal FileNode will be ignored");
        }
    }
    }
    } MatrixExtracted=true;
    return;
}
public static void initializeFileStructure() {
    if (Globals.EXAMPLE_NUM == 1) {
        exExample_10();
    } else {
        if (Globals.EXAMPLE_NUM==2){
            exExample_20();
        }else{
            if (Globals.EXAMPLE_NUM==3){
                exExample_3_ACE();
            }
        else { System.out.println("Wrong Example Nomber.It has to be 1, 2, or 3");
            return;
        }
    }
}

public static void main(String args[]) {
    Globals.EXAMPLE_NUM= Integer.parseInt(args[0]);
    initializeFileStructure();
displayFileStructure(): predicate4_5_GoodFileStructure();
}
protected static boolean predicate4_10() {
    boolean returnvalue=false;
    output="\nChecking for Predicate 4_10: ComponentsMainFilesShouldBeDistinct\n";
    for (int i=1; i<=FileStructure.F; i++) {
        for (int j=1; j<=FileStructure.C[i]; j++) {
            for (int k=1; k<=3; k++) {
                String tmpName=FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].FileName;
                String tmpType=FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].FileType;
                String tmpFile=tmpName+"."+tmpType;
                for (int jj=1; jj<=FileStructure.C[i]; jj++) {
                    if (jj==i) { break; }
                }
                for (int kk=1; kk<=3; kk++) {
                    String tmp2Name=FileStructure.CO_Fr_Files[i].ComponentList[jj].FileList[kk].FileName;
                    String tmp2Type=FileStructure.CO_Fr_Files[i].ComponentList[jj].FileList[kk].FileType;
                    String tmp2File=tmp2Name+"."+tmp2Type;
                    if ((tmp2Name==tmpName)&&((tmp2Type==tmpType)) {
                        output=output+"\nVIOLATION(Predicate4_10): File "+tmpFile+" belongs to Components("+i+","+jj+") and Component("+i+","+jj+") at the same time";
                    }
                    }
                }
            }
        }
    }
}

141
returnValue=false;
{
}
}
}
}
}
}
}
}
}
}
}
}
}

output=output+"\nChecking for Predicate 4_10 is complete — STATUS: "+ returnValue;
output=output+"\n="

reportViolation(output);
return returnValu e;
}

protected static boolean predicate4_11 () {
    reportViolation("\nChecking for Predicate 4_11: GoodUseOfFiles = predicates 2_14, 2_15, 2_16, and 2_17");
    boolean returnValue1=predicate4_12();
    boolean returnValue2=predicate4_13();
    boolean returnValue3=predicate4_14();
    boolean returnValue4=predicate4_15();
    boolean returnValue=returnValue1 &&returnValue2 &&returnValue3 &&returnValue4;
    reportViolation("\nChecking for Predicate 4_11 is complete — STATUS: "+ returnValue);
    reportViolation("="

    return (returnValue);
}

protected static boolean predicate4_12() {
    boolean returnValu e=true;
    reportViolation("\nChecking for Predicate 4_12: InclusionLoopsMayNotExist");
    if (!MatrixExtracted){
        extractInclusionMatrix();
        // displayInclusionMatrix();
    }
}

recursiveCheckForLoops(1,1); if (LoopOccurred) {returnValue=false;}

reportViolation("\nChecking for Predicate 4_12 is complete — STATUS: "+ return Value);
reportViolation("="

    return returnValu e;
}

protected static boolean predicate4_13 () {
    reportViolation("\nChecking for Predicate 4_13: GoodUseOfTop_Files = predicates 2_18, 2_19, and 2_20");
    boolean returnValue1=predicate4_17();
    boolean returnValue2=predicate4_18();
    boolean returnValue3=predicate4_19();
    boolean returnValue=returnValue1 && returnValue2 && returnValue3;
    reportViolation("\nChecking for Predicate 4_13 is complete — STATUS: "+ returnValue);
    reportViolation("="

142
return (returnValue);
}

protected static boolean predicate4_14 () {
    reportViolation("\nChecking for Predicate 4_14: GoodUseOf_intermediate_Files = predicates 2_21 and 2_22\n");
    boolean returnValuel=predicate4_20();
    reportViolation("\nChecking for Predicate 4_14 is complete — STATUS: "+returnValuel);
    reportViolation("\n");
    return (returnValuel);
}

protected static boolean predicate4_15 () {
    reportViolation("\nChecking for Predicate 4_15: GoodUseOf_internal_Files = predicates 2_23 and 2_24\n");
    boolean returnValue2=predicate4_21();
    boolean returnValue2=predicate4_22();
    boolean returnValuel && returnValue2;
    reportViolation("\nChecking for Predicate 4_15 is complete — STATUS: "+returnValue2);
    return (returnValue2);
}

protected static boolean predicate4_16 () {
    reportViolation("\nChecking for Predicate 4_16: GoodUseOf_shared_Files = predicates 2_25 and 2_26\n");
    boolean returnValuel=predicate4_23();
    reportViolation("\nChecking for Predicate 4_16 is complete — STATUS: "+returnValuel);
    return (returnValuel);
}

protected static boolean predicate4_17 () {
    boolean returnValuel=true; reportViolation("\nChecking for predicate 4_17: NoFileMayIncludeA.top_File\n");
    tmpFileNode[] topNodeList = new tmpFileNode[Globals.MAX_FR * Globals.MAX_CO];
    int tmp=0;
    for (int i=1; i<=FileSystem.F; i++){
        for (int j=1; j<=FileSystem.C[i]; j++){
            for (int k=1; k<=3; k++)
                if (FileSystem.CO_Fr_Files[i].ComponentList[j].FileType=="top"){
                    topNodeList[++tmp] = new tmpFileNode();
                    topNodeList[tmp].NodeNumber=FileSystem.CO_Fr_Files[i].ComponentList[j].FileL-
                    ist[k].ID.FileNameNumber;
                    topNodeList[tmp].Name=FileSystem.CO_Fr_Files[i].ComponentList[j].FileName;
                    topNodeList[tmp].Type=FileSystem.CO_Fr_Files[i].ComponentList[j].FileType;
                    topNodeList[tmp].CO_Number=j;
                    topNodeList[tmp].FR_Number=i;
                }
            }
        }
    }
}
 Appendix A1: A Prototype For CO-Toolkits's File Structures: An Overview

```java
} // Checking
for (int ii=1; ii<=tmp; ii++) { // index ii for topNodeList[]
    for (int i=1; i<=FileStructure.F[i++]; // index i for frameworks
        for (int j=1; j<=FileStructure.C[i][j++]; // index j for components
            for (int k=1; k<=3; k++) { // index k for files
                int tmpFileNodeNumber=FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].ID.FileNodeNumber;
                String tmpFileName=FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].FileName;
                String tmpFileType=FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].FileType;
                int tmpFrameworkNumber=i;
                int tt=0;
                while (FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].IncludeList[++tt] > 0){
                    int tmpIncludedFileNodeNumber=FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].IncludeList[tt];
                    if (tmpIncludedFileNodeNumber==topNodeList[ii].NodeNumber) {
                        reportViolation("\nVIOLATION(Predicate4_18): Top File "+topNodeList[ii].Name+"."+topNodeList[ii].Type+
                            " from Framework("+topNodeList[ii].FR_Number+")) is included by "+tmpFileName+");
                        returnValue=false;
                    }
                }
            }
        }
    }
}
for (int i=1; i<=FileStructure.S;i++){
    for (int j=1; j<=FileStructure.SF[i][j++];
        int tmpFileNodeNumber_sh=FileStructure.CO_Sh_Files[i].SharedFileList[j].ID.FileNodeNumber;
        String tmpFileName_sh=FileStructure.CO_Sh_Files[i].SharedFileList[j].FileName;
        String tmpFileType_sh=FileStructure.CO_Sh_Files[i].SharedFileList[j].FileType;
        int tmpSharedPoolNumber=i;
        int tmpp=0;
        while (FileStructure.CO_Sh_Files[i].SharedFileList[j].IncludeList[++tmpp]>0){
            if (FileStructure.CO_Sh_Files[i].SharedFileList[j].IncludeList[tmpp]== topNodeList[ii].NodeNumber){
                reportViolation("\nVIOLATION(Predicate4_18): Top File "+topNodeList[ii].Name+"."+topNodeList[ii].Type+
                            " from Framework("+topNodeList[ii].FR_Number+")) is included by the shared file "+tmpFileName_sh+".
                            "+tmpFileType_sh+" from SharedPool("+tmpSharedPoolNumber+"));
                          returnValue=false;
            }
        }
    }
}
```
protected static boolean predicate4_18 () {
    boolean return_value = true;
    reportViolation("unChecking for predicate 4_18: FromSame_Fr_A_top_File-
    MayIncludeOnlyIts_intermediate_Files\n");
    for (int i = 1; i <= FileStructure.F; i++) {
        for (int j = 1; j <= FileStructure.C[i]; j++) {
            for (int k = 1; k <= 3; k++) {
                if (FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].FileType == "top") {
                    int tmp_file_node_number = FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].ID.FileNode-
                    Number;
                    String tmp_file_name = FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].FileName;
                    String tmp_file_type = FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].FileType;
                    int tmp_framework_number = i;
                    int tmp = 0;
                    while (FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].IncludeList[++tmp] > 0) {
                        int tmp_included_file_number = FileStructure.CO_Fr_Files[i].ComponentList[j].FileL-
                        ist[k].IncludeList[tmp];
                        boolean found2 = false;
                        for (int ii = 1; ii <= FileStructure.F; ii++) {
                            if (found2) { break; }
                            for (int jj = 1; jj <= FileStructure.C[ii]; jj++) {
                                if (found2) { break; }
                                for (int kk = 1; kk <= 3; kk++) {
                                    if (found2) { break; }
                                    if (FileStructure.CO_Fr_Files[ii].ComponentList[jj].FileList[kk].ID.FileNode-
                                    Number == tmp_included_file_number) {
                                        found2 = true;
                                        int tmp2_file_node_number = FileStructure.CO_Fr_Files[ii].ComponentList[jj].FileL-
                                        ist[kk].ID.FileNodeNumber;
                                        String tmp2_file_name = FileStructure.CO_Fr_Files[ii].ComponentList[jj].FileL-
                                        ist[kk].FileName;
                                        String tmp2_file_type = FileStructure.CO_Fr_Files[ii].ComponentList[jj].FileList[kk].Fi-
                                        leType;
                                        int tmp2_framework_number = ii;
                                        if (tmp2_framework_number != tmp_framework_number) {
                                            reportViolation("unVIOLATION(Predicate4_18): Top File "+tmp_file_name+"."+tmp_file-
                                            type+" from Framework("+tmp_framework_number+")) includes File "+tmp2_file_name+"."+tmp2_file-
                                            type+" from Framework("+tmp2_framework_number+"));
                                            return_value = false;
                                            break;
                                        }
                                        if (tmp2_file_node_number == tmp_file_node_number) {
                                            reportViolation("unVIOLATION(Predicate4_18): Top File "+tmp_file_name+"."+tmp_file-
                                            type+" from Framework("+tmp_framework_number+")) includes itself");
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
returnValue=false;
break;
}
if ((tmp2FrameworkNumber==tmpFrameworkNumber) &&
    (tmp2FileName==tmpFileName) && (tmp2FileType=="lnl")){
    reportViolation("\nVIOLATION(Predicate4_18): Top File \"+tmpFileName+\"."+tmpFi-
    leType+" from Framework("+tmpFrameworkNumber+) includes its Internal File\");
    returnValue=false;
    break;
}
if ((tmp2FrameworkNumber!=tmpFrameworkNumber) && (tmp2FileName !=tmpFile-
    Name)){
    reportViolation("\nVIOLATION(Predicate4_18): Top File \"+tmpFileName+\"."+tmpFi-
    leType+" from Framework("+tmpFrameworkNumber+) includes File \"+tmp2FileName+\"."+tmp2File-
    Type+" from Framework("+tmp2FrameworkNumber+)\”);
    returnValue=false;
    break;
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
} }

int NumberOfSharedFiles = tmp;

// Checking
for (int i=1; i<=FileStructure.F; i++)
for (int j=1; j<FileStructure.C[i]; j++)
for (int k=1; k<=3; k++)
    if (FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].FileType == "top")
        int tmpFileNumber = FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].ID.FileNodeNumber;
    String tmpFileName = FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].FileName;
    String tmpFileType = FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].FileType;
    int tmpFrameworkNumber = i;
    int tmppp = 0;
while (FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].IncludeList[++tmppp] > 0) {
    // find out if this file node is a shared one
    int tmSharedNode_check = FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].IncludeList[tmppp];
    int tt = 0;
    boolean found = false;
    while (tt <= tmp){
        if (found) {break;}
        tt = tt + 1; if (tt > tmp) {break;}
        if (sharedNodeList[tt].NodeNumber == tmSharedNode_check){
            found = true;
            // check if tmpFrameworkNumber belongs to the permitted set of Frameworks to access the
            SharedPool of the detected shared file
            int tmSharedPoolNumber = sharedNodeList[tt].FR_Number;
            int tmppp = 0;
            boolean found2 = false;
            while (FileStructure.CO_Sh_Files[tmSharedPoolNumber].PermittedFrameworks[++tmppp] > 0) {
                if (found2) {break;}
                if (FileStructure.CO_Sh_Files[tmSharedPoolNumber].PermittedFrameworks[tmppp] == tmpFrameworkNumber) {
                    found2 = true;
                }
            }
            if (!found2) {
                reportViolation("VIOLATION(Predicate4.19): Top File " + tmpFileName + ";" + tmpFileType + " from Framework(" + tmpFrameworkNumber + ") includes a shared file from a non-accessible SharedPool(" + tmSharedPoolNumber + ");
            return Value = false;
        }
    }
}
protected static boolean predicate4_20 ( ) {
    boolean return_value=true; reportViolation("\nChecking for predicate 4_20: An_intermediate_fileMayIn-
cludeOnlyIts_internal_FileAndMayIncludeOther_intermediate_FilesFromSame_Fr\n");
    for (int i=1; i<=FileStructure.F; i++)
        for (int j=1; j<=FileStructure.C[i]; j++)
            for (int k=1; k<=3; k++)
                if (FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].FileType=="inm"){
                    int tmpFileNumber=FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].ID.FileNo-
                    deNumber;
                    int tmpFrameworkNumber=i;
                    String tmpFileName=FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].FileName;
                    String tmpFile=FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].FileType;
                    int tmp=0;
                    while (FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].IncludeList[++tmp]>0){
                        int tmpIncludedFileNumber=FileStructure.CO_Fr_Files[i].ComponentList[j].FileL-
                        ist[k].IncludeList[tmp];
                        boolean found=false;
                        for (int ii=1; ii<=FileStructure.F; ii++)
                            if (found) { break; }
                        for (int jj=1; jj<=FileStructure.C[ii]; jj++)
                            if (found) { break; }
                        for (int kk=1; kk<=3; kk++)
                            if (found) { break; }
                        int tmp2FileNumber=FileStructure.CO_Fr_Files[ii].ComponentList[jj].FileL-
                        ist[kk].ID.FileNodeNumber;
                        int tmp2FrameworkNumber=ii;
                        String tmp2FileName=FileStructure.CO_Fr_Files[ii].ComponentList[jj].FileList[kk].File-
                        Name;
                        String tmp2File=FileStructure.CO_Fr_Files[ii].ComponentList[jj].FileList[kk].File-
                        Type;
                        if (tmp2FileNumber==tmpIncludedFileNumber){
                            found=true;
                            if (tmp2FileNumber==tmpFileNumber){
                                reportViolation("\nVIOLATION(Predicate4_20): Intermediate File "+tmpFile-
                                Name+"."+tmpFileType+" from Framework("+tmpFrameworkNumber+"") includes itself");
                                return_value=false;
                            }
                        }
                    }
                }
    return return_value;
}
if ((tmp2FileName==tmpFileName) && (tmp2FrameworkNumber==tmpFrameworkNumber) && (tmp2FileType=="top")){
    reportViolation("\nVIOLATION(Predicate4_20): Intermediate File "+tmpFileName+":"+tmpFileType+" from Framework("+tmpFrameworkNumber+")) includes its top file "+tmp2FileName+":"+tmp2FileType+";
    returnValue=false;
}
if ((tmp2FrameworkNumber==tmpFrameworkNumber) && (tmp2FileName!=tmpFileName) && (tmp2FileType=="inl")){
    reportViolation("\nVIOLATION(Predicate4_20): Intermediate File "+tmpFileName+":"+tmpFileType+" from Framework("+tmpFrameworkNumber+")) includes the non-intermediate "+tmp2FileName+":"+tmp2FileType+" from the same framework");
    returnValue=false;
}
if (tmp2FrameworkNumber!=tmpFrameworkNumber){
    reportViolation("\nVIOLATION(Predicate4_20): Intermediate File "+tmpFileName+":"+tmpFileType+" from Framework("+tmpFrameworkNumber+")) includes "+tmp2FileName+":"+tmp2FileType+" from Framework("+tmpFrameworkNumber+"))");
    returnValue=false;
}
}
if (!found) {
    boolean found_sh=false;
    for (int ii=1; ii<=FileStructure.S; ii++){
        if (found_sh) { break; }
        for (int jj=1; jj<=FileStructure.SF[ii]; jj++){
            if (found_sh) { break; }
            if (FileStructure.CO_Sh_Files[ii].SharedFileList[jj].ID.FileNodeNumber==tmpIncludedFileNumber) {
                found_sh=true;
                String tmpFileName_sh=FileStructure.CO_Sh_Files[ii].SharedFileList[jj].FileName;
                String tmpFileType_sh=FileStructure.CO_Sh_Files[ii].SharedFileList[jj].FileType;
                reportViolation("\nVIOLATION(Predicate4_20): Intermediate file "+tmpFileName+":"+tmpFileType+" from Framework("+tmpFrameworkNumber+")) includes "+tmpFileName+":"+tmpFileType+" from SharedPool("+ii+"));
                returnValue=false;
}
    }
}
if (found_sh){
    reportViolation("\nERROR(Predicate4_20): Intermediate File "+tmpFileName+"."+tmpFileType+" includes FileNode("+tmpIncludedFileNumber+")) that is not found in any Framework or Sha-
protected static boolean predicate4_21() {
    boolean returnVal = true;
    reportViolation("nChecking for predicate 4_21: An_internal_FileMayNotIncludeAnyFile");
    tmpFileNode[] InternalNodeList = new tmpFileNode[Globals.MAX_FR * Globals.MAX_CO];
    // Checking
    for (int i = 1; i <= FileStructure.F; i++) { // index i for frame
        for (int j = 1; j <= FileStructure.C[i]; j++) { // index j for components
            for (int k = 1; k <= 3; k++) { // index k for files
                if (FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].FileType == "inl") {
                    int tt = 0;
                    while (FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].IncludeList[++tt] > 0) {
                    }
                }
            }
        }
    }
    returnVal = false;
}

protected static boolean predicate4_22() {
    boolean returnVal = true;
    reportViolation("nChecking for predicate 4_22: NoFileMayIncludeAn_Internal_FileExceptIts_intermediate/File\n");
    tmpFileNode[] InternalNodeList = new tmpFileNode[Globals.MAX_FR * Globals.MAX_CO];
    int tmp = 0;
    for (int i = 1; i <= FileStructure.F; i++) {
        for (int j = 1; j <= FileStructure.C[i]; j++) {

    reportViolation("nChecking for predicate 4_21 is complete — STATUS:" + returnVal);
    return returnVal;
}

protected static boolean predicate4_22() {
    boolean returnVal = true;
    reportViolation("nChecking for predicate 4_22: NoFileMayIncludeAn_Internal_FileExceptIts_intermediate_File\n");
    tmpFileNode[] InternalNodeList = new tmpFileNode[Globals.MAX_FR * Globals.MAX_CO];
    int tmp = 0;
    for (int i = 1; i <= FileStructure.F; i++) {
        for (int j = 1; j <= FileStructure.C[i]; j++) {

    reportViolation("nChecking for predicate 4_21 is complete — STATUS:" + returnVal);
    return returnVal;
}

protected static boolean predicate4_22() {
    boolean returnVal = true;
    reportViolation("nChecking for predicate 4_22: NoFileMayIncludeAn_Internal_FileExceptIts_intermediate_File\n");
    tmpFileNode[] InternalNodeList = new tmpFileNode[Globals.MAX_FR * Globals.MAX_CO];
    int tmp = 0;
    for (int i = 1; i <= FileStructure.F; i++) {
        for (int j = 1; j <= FileStructure.C[i]; j++) {

    reportViolation("nChecking for predicate 4_21 is complete — STATUS:" + returnVal);
    return returnVal;
}

protected static boolean predicate4_22() {
    boolean returnVal = true;
    reportViolation("nChecking for predicate 4_22: NoFileMayIncludeAn_Internal_FileExceptIts_intermediate_File\n");
    tmpFileNode[] InternalNodeList = new tmpFileNode[Globals.MAX_FR * Globals.MAX_CO];
    int tmp = 0;
    for (int i = 1; i <= FileStructure.F; i++) {
        for (int j = 1; j <= FileStructure.C[i]; j++) {

    reportViolation("nChecking for predicate 4_21 is complete — STATUS:" + returnVal);
    return returnVal;
}
for (int k=1; k<=3; k++){
    if (FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].FileType=="int"){
        InternalNodeList[++tmp] = new tmpFileNode();
        InternalNodeList[tmp].ID.FileNodeNumber=FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].FileName;
        InternalNodeList[tmp].Name=FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].FileType;
        InternalNodeList[tmp].CO_Number=j;
        InternalNodeList[tmp].FR_Number=i;
    }
    }
}

// Checking
for (int ii=1; ii<tmp; ii++) {  // index ii for InternalNodeList[]
    for (int i=1; i<=FileStructure.F; i++) {  // index i for frameworks
        for (int j=1; j<=FileStructure.C[i]; j++) {  // index j for components
            for (int k=1; k<=3; k++) {  // index k for files
                int tt=0;
                while (FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].IncludeList[++tt] > 0){
                    if (FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].IncludeList[tt] == InternalNodeList[ii].NodeNumber) &&
                        ((FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].FileName != InternalNodeList[ii].Name) ||
                        ((FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].FileName == InternalNodeList[ii].Name) &&
                        (FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].FileType != "int")))
                    {
                        reportViolation("\nVIOLATION(Predicate 4_22): Internal File "+InternalNodeList[ii].Name++" for File "+FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].FileName++" is included by "+FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].FileType++" from Framework("+++") \nComponent("+++");
                        returnValue=false;
                    }
                }
            }
        }
    }
}

reportViolation("\nChecking for predicate 4_22 is complete — STATUS:"+returnValue);
return returnValue;
}

protected static boolean predicate4_23() {
    output="\nChecking for Predicate4_23: A_shared_FileMayIncludeOnlyOther_shared_FilesFromSame_SPn";
boolean returnValue=true;
for (int i=1; i<=FileStructure.S; i++)
    // extract the shared files list of FileNodes for the ith shared pool
    int tmp=0;
    int[] tmpNodeList=new int[FileStructure.SF[i]+1];
    for (int j=1; j<=FileStructure.SF[i]; j++)
        tmpNodeList[++]tmp=FileStructure.CO_Sh_Files[i].SharedFileList[j].ID.FileNodeNumber;
    }
    reportViolation(""");
    for (int sf=1; sf<=FileStructure.SF[i]; sf++)
        int ttt=0;
        while (FileStructure.CO_Sh_Files[i].SharedFileList[sf].IncludeList[++ttt]>0)
            int ttt=1;
            while ((ttt++tmp) && (FileStructure.CO_Sh_Files[i].SharedFileList[sf].IncludeList[ttt]++tmpNodeList[ttt])
                ttt=ttt+1;
                if (ttt++tmp)
                    output=output+"nVIOLATION(Predicate4.23): FileNode("+FileStructure.CO_Sh_Files[i].SharedFileList[sf].ID.FileNodeNumber+
                        ") from SharedPool("+i") includes FileNode("+FileStructure.CO_Sh_Files[i].SharedFileList[sf].IncludeList[ttt]+
                        ") from another SharedPool or Framework";
                    returnValue=false;
                    }
    }
    output=output+"nChecking for Predicate4.23 is complete —— STATUS:"+returnValue;
    reportViolation(output);
    return returnValue;
}
protected static boolean predicate4_5_GoodFileStructure() {
    reportViolation("nChecking for GoodFileStructure_predicate4_5: GoodFileStructure = predicates 2_6,
        2_7, and 2_8");
    boolean returnValue1=predicate4_6();
    boolean returnValue2=predicate4_7();
    boolean returnValue3=predicate4_8();
    boolean returnValue=returnValue1 && returnValue2 && returnValue3;
    reportViolation("nChecking for GoodFileStructure_predicate4_5 is complete —— STATUS:"+returnValue);
    return returnValue;
}
protected static boolean predicate4_6() {
    boolean returnValue=true;
    reportViolation("nChecking for Predicate 4_6: OnlyFourFileCategories\n");
    for (int i=1; i<=FileStructure.F; i++)
        for (int j=1; j<=FileStructure.C[i]; j++)
            152
for (int k=1; k<=3; k++)
    String tmpType=FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].FileType;
    String tmpName=FileStructure.CO_Fr_Files[i].ComponentList[j].FileList[k].FileName;
    if ((tmpType!="top" ) & &(tmpType!="inn") & &(tmpType!="int") )
        reportViolation("\nVIOLATION(Predicate4_6): File "+tmpName+"."+tmpType+ " from Framework("+i+") uses an invalid File Extension");
    returnValue=false;
}

for (int i=1; i<=FileStructure.S.F[i]; i++)
    for (int j=1; j<=FileStructure.S.F[i].SharedFileList[j].FileType;
        String tmpName=FileStructure.CO_Fr_Files[i].SharedFileList[j].FileName;
        if (tmpType!="s")
            reportViolation("\nVIOLATION(Predicate4_6): File "+tmpName+"."+tmpType+ " from Shared-Pool("+i+") uses an invalid File Extension");
    returnValue=false;
}

reportViolation("\nChecking for Predicate 4_6 is complete — STATUS: "+returnValue);

protected static boolean predicate4_7() {
    reportViolation("\nChecking for Predicate 4_7: GoodDistinctionOfFiles= Predicates 2_9, and 2_10");
    boolean returnValue1=predicate4_9();
    boolean returnValue2=predicate4_10();
    boolean returnValue=returnValue1& &returnValue2;
    reportViolation("\nChecking for Predicate 4_7 is complete — STATUS: "+returnValue);
    return (returnValue);
}

protected static boolean predicate4_8() {
    reportViolation("\nChecking for Predicate 4_8: GoodFileInclusionRelation = predicates 2_11, 2_12, and 2_13");
    boolean returnValue1=predicate4_11();
    boolean returnValue2=predicate4_12();
    boolean returnValue=returnValue1 & & returnValue2;
    reportViolation("\nChecking for Predicate 4_8 is complete — STATUS: "+returnValue);
    return (returnValue);
}
protected static boolean predicate4_9() {
    boolean returnValue=true;
    reportViolation("nChecking for Predicate 4_9: Fr_SetsOfFilesShouldBeDistinct\n");
    for (int i=1; i<=FileSystem.F; i++) {
        for (int j=1; j<=FileSystem.C[i]; j++) {
            for (int k=1; k<=3; k++) {
                String tmpName=FileSystem.CO_Fr_Files[i].ComponentList[jj].FileList[k].FileName;
                String tmpType=FileSystem.CO_Fr_Files[i].ComponentList[jj].FileList[k].FileType;
                String tmpFile=tmpName+"."+tmpType;
                for (int ii=1; ii<=FileSystem.F; ii++) {
                    for (int jj=1; jj<=FileSystem.C[ii]; jj++) {
                        if ((ii==i) && (jj==j)) {
                            break;
                        }
                        for (int kk=1; kk<=3; kk++) {
                            String tmp2Name=FileSystem.CO_Fr_Files[ii].ComponentList[jj].FileList[kk].FileName;
                            String tmp2Type=FileSystem.CO_Fr_Files[ii].ComponentList[jj].FileList[kk].FileType;
                            String tmp2File=tmp2Name+"."+tmp2Type;
                            if ((tmp2Name.equals(tmpName)) && (tmp2Type.equals(tmp2Type))) {
                                reportViolation("nVIOLATION(Predicate4_9): File \"+
                                tmpFile\" belongs to Framework("\"+i+"\") and Framework("+
                                ii+") at the same time");
                                returnValue=false;
                            }
                        }
                    }
                }
            }
        }
    }
    reportViolation("nChecking for Predicate 4_9 is complete — STATUS: \"+returnValue\n");
    reportViolation("n================================================================\n");
    return returnValue;
}
private static void recursiveCheckForLoops (int x, int y) {
    if (x>M) {return;}
    if (x==y) {y=y+1;}
    if (x>=M) && (y>M) {return;}
    while((y<=M) && ((x+y)%1) {y=y+1;}
    if (x==y) {
        y=y+1;
    }
    while((y<=M) && (InclusionMatrix[x][y]==0)) {y=y+1;}
}
if (y>M) {
    if ((Ptr==0) && (x<=M)) {
        recursiveCheckForLoops(x+1,1);
    } else {
        Ptr=Ptr-1;
    }
while ((Ptr>0) && (ListNodes[Ptr+1]+1 > M)) {Ptr=Ptr-1; }
if (Ptr>0) {
    if (x<=M) {
        recursiveCheckForLoops(ListNodes[Ptr], ListNodes[Ptr+1]+1);
    }

} else {
    if (x<=M) {
        recursiveCheckForLoops(ListNodes[Ptr+1]+1, 1);
    }

} 
else {
    if (Ptr!=1) {ListNodes[++Ptr]=x; ListNodes[++Ptr]=y;}
    else {ListNodes[++Ptr]=y;}
    // compress
    if (Ptr>=4) {
        if (ListNodes[Ptr-1]==ListNodes[Ptr-2]) {
            ListNodes[Ptr-1]=ListNodes[Ptr];
            Ptr=Ptr-1;
        }
    }
    // cleanup
    int tmpPtr=Ptr-1;
    while ((tmpPtr>1) && (ListNodes[Ptr]!=ListNodes[tmpPtr])) {tmpPtr=tmpPtr-1;}
    if (tmpPtr>1) {
        Ptr=Ptr-1;
        if ((y<=M) && (x<=M)) {
            recursiveCheckForLoops(ListNodes[Ptr], ListNodes[Ptr+1]+1);
        }
    }
    // check for loops
    if (Ptr>=2) {
        if (ListNodes[Ptr]==ListNodes[1]) {
            LoopOcurred=true;
            displayLoop(Ptr);
            Ptr=Ptr-1;
            if (x<=M) {
                recursiveCheckForLoops(ListNodes[Ptr], ListNodes[Ptr+1]+1);
            }
        }
    }
}

}
private static void recursiveCheckForMultiplePaths (int x, int y, int pathID) {
    if (x>M) return; if ((x==M) && (y>M)) return;
    while((y<=M) && ((InclusionMatrix[x][y]==0) || (x==y))) y=y+1;
    if (pathID==1) {
        if (y<=M) {
            if (Ptr1==0) { StartNode=x; }
            Path1[++Ptr1]=x;
            Path1[++Ptr1]=y;
            //Compress
            if (Ptr1>2) {
                if (Path1[Ptr1-1]==Path1[Ptr1-2]) {
                    Ptr1=Ptr1-1;
                    Path1[Ptr1]=Path1[Ptr1+1];
                }
            //clean up any redundancies as well as skip any loops
                int tmpPtr1=Ptr1-1;
                while (((tmpPtr1>=1) && (Path1[Ptr1]!=Path1[tmpPtr1])) || tmpPtr1=tmpPtr1-1;){
                    if (tmpPtr1>=1) {Ptr1=Ptr1-1;}
                }
            }
            if (Ptr1==2) {
                recpnt=1;
                recursiveCheckForMultiplePaths(Path1[Ptr1], Path1[Ptr1+1]+1,1);
            }
            if (Ptr1<2) {
                StartNode=StartNode+1;
                recpnt=2;
                if (StartNode==M) {recursiveCheckForMultiplePaths(StartNode,1,1);}
                else {return;}
            }
            if (Ptr1>2) { // Path1 has more than two nodes ... ready .. move to Path2
                if (StartNode==M) {
                    recpnt=3;
                    recursiveCheckForMultiplePaths(StartNode,1,2);
                    recpnt=32;
                    recursiveCheckForMultiplePaths(Path1[Ptr1],Path1[Ptr1+1]+1,1);
                }
                else {return;}
            }
        }
    }
}
else if (Ptr1<2)
    StartNode=StartNode+1;
    Ptr1=0;
    recpt=5;
    if (StartNode<=M) recursiveCheckForMultiplePaths(StartNode,1,1);
    else {return;}
}
}
else if (ptrID==2)
    if(y<=M){
        Path2[++Ptr2]=x;
        Path2[++Ptr2]=y;
        //compress
        if(Ptr2>=2){
            if(Path2[Ptr2-1]==Path2[Ptr2-2]){
                Ptr2=Ptr2-1;
                Path2[Ptr2]=Path2[Ptr2+1];
            }
        //clean up and skip any loops
        int tmpPtr2=Ptr2-1;
        while ((tmpPtr2>=1) && (Path2[Ptr2]!=Path2[tmpPtr2])){tmpPtr2=tmpPtr2-1;}
        if (tmpPtr2>1){Ptr2=Ptr2-1;}
        }
    // ensure that Path2 and Path1 are distinct in the middle
    if(Ptr2>=2){
        if(Path2[2]==Path1[2]){
            Ptr2=1;
            recpt=6;
            recursiveCheckForMultiplePaths(Path2[Ptr2],Path2[Ptr2+1]+1,2);
        }
    }
    //check for multiple paths
    if (Ptr2>=2)
        if((Path1[1]==Path2[1])&&(Path1[Ptr1]==Path2[Ptr2])){
            MultiplePathOccured=true;
            displayMultiplePath(Ptr1,Ptr2);
            Ptr2=Ptr2-1;
        }
    recpt=7;
    recursiveCheckForMultiplePaths(Path2[Ptr2],Path2[Ptr2+1]+1,2);
}
// forward
if(Ptr2<2){
    Ptr2=0;
    // StartNode=StartNode+1;
// Pr1=0;
// recpnt=8;
// if (StartNode<=M){
// recursiveCheckForMultiplePaths(StartNode,1,1);
// }
// else{
// return;
// }
// }
else {// pathID=2 & y>m
  if(Ptr2>0){Ptr2=Ptr2-1;}
  if(Ptr2>0){
    recpnt=9;
    recursiveCheckForMultiplePaths(Path2[Ptr2], Path2[Ptr2+1]+1,2);
  }
  else{ // Pr2<=0
    Ptr2=0;
    // StartNode=StartNode+1;
    // Pr1=0;
    // recpnt=10;
    // if (StartNode<=M){
    //   recursiveCheckForMultiplePaths(StartNode,1,1);
    // }
    // else{
    //   return;
    // }
    // }
    // }
    // }
}

public static void reportInclusionMatrix(String output){
  System.out.println("\n"+output+"\n");
  return;
}
public static void reportOutput(String output){
  System.out.println("\n"+output+"\n");
  return;
}
public static void reportViolation(String output){
  System.out.println("\n"+output+"\n");
  return;
}
A1.2.2  Globals.java

This file contains the class that is used to provide all global variables used in the implementation of this prototype. Next is the source code listing for this file:

```java
package V1_FileStructure_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#1 For CO_Toolkits File Structures — Chapter 4 of My Ph.D. Thesis
 */

public class Globals {
    public static int EXAMPLE_NOM;    // number of the example
    public static int MAX_FR=20;      // maximum number of frameworks -1
    public static int MAX_CO=50;      // maximum number of components -1 per framework
    public static int MAX_FL=4;       // maximum number of files -1 per component
    public static int MAX_SP=3;       // maximum number of shared pools -1
    public static int MAX_SF=30;      // maximum number of shared files per shared pool -1
    public static int MAX_INCL= 20;   // maximum number of files that can be included by a single included files
    public static int NodeCount=0;
    public Globals() { }
}
```

A1.2.3  tmpFileNode.java

This file contains the class that is used to represent an internal temporary file node. Next is the source code listing for this file:

```java
package V1_FileStructure_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#1 For CO_Toolkits File Structures — Chapter 4 of My Ph.D. Thesis
 */
```
public class tmpFileNode {
    int NodeNumber;
    String Name;
    String Type;
    int CO_Number;
    int FR_Number;
    public tmpFileNode() {
    }
}

A1.2.4 FileID.java

This file contains the class that is used to represent a file ID. Next is the source code listing for this file:

package V1_FileStructure_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#1 For CO_Toolkits File Structures — Chapter 4 of My Ph.D. Thesis
 */
public class FileID {
    int ParentFrameworkNodeNumber;
    int ParentComponentNodeNumber;
    int FileNodeNumber; // FileNodeNumber must be unique
    // — no other file (whether shared or not) can have the same number
    public FileID() { super(); }
}

A1.2.5 FileNode.java

This file contains the class that is used to represent a FileNode in the File Structure.

Next is the source code listing for this file:

package V1_FileStructure_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 */

160
A1.2.6 ComponentID.java

This file contains the class that is used to represent a component ID. Next is the source code listing for this file:

```java
package V1_FileStructure_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#1 For CO_Toolkits File Structures — Chapter 4 of My Ph.D. Thesis
 */
public class ComponentID {
    int ParentFrameworkNodeNumber;
    int ComponentNodeNumber;    // this must be a unique number for each component/**
    public ComponentID() {
    }
}
```

A1.2.7 ComponentNode.java

This file contains the class that is used to represent a component node. Next is the source code listing for this file:

```java
package V1_FileStructure_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
```
A1.2.8 FrameworkNode.java

This file contains the class that is used to represent a framework node. Next is the source code listing for this file:

```java
package V1_FileStructure_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#1 For CO_Toolkits File Structures — Chapter 4 of My Ph.D. Thesis
 */
public class FrameworkNode {
    int FrameworkNodeNumber;
    // this must be a unique number for each framework
    ComponentNode ComponentList[] = new ComponentNode[Globals.MAX_CO];
    public FrameworkNode() {super();}
}
```

A1.2.9 SharedFileID.java

This file contains the class that is used to represent a shared file ID. Next is the source code listing for this file:

```java
package V1_FileStructure_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 */
```
**Appendix A1: A Prototype For CO-Toolkits's File Structures: An Overview**

* Hamilton, Ontario  
* January 1999  
* Proto-type Tool View#1 For CO_Toolkits File Structures — Chapter 4 of My Ph.D. Thesis  
*

```java
public class SharedFileID {
    int ParentSharedPoolNodeNumber;
    int FileNodeNumber;    // FileNodeNumber must be unique
    // — no other file (whether shared or not) can have the same number
    public SharedFileID() { super(); }
}

A1.2.10 SharedFileNode.java

This file contains the class that is used to represent a shared file node in the File Structure. Next is the source code listing for this file:

```java
package V1_FileStructure_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#1 For CO_Toolkits File Structures — Chapter 4 of My Ph.D. Thesis
 */
public class SharedFileNode {
    SharedFileID ID=new SharedFileID();
    String FileName= null;
    String FileType= null;
    int includeList[] = new int[Globals.MAX_INCL];
    public SharedFileNode() { super(); }
}

A1.2.11 SharedPoolNode.java

This file contains the class that is used to represent a shared file node. Next is the source code listing for this file:

```java
package V1_FileStructure_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group

163
Appendix A1: A Prototype For CO-Toolkits's File Structures: An Overview

* Department of Electrical & Computer Engineering
* McMaster University
* Hamilton, Ontario
* January 1999
* Proto-type Tool View#1 For CO_Toolkits File Structures — Chapter 4 of My Ph.D. Thesis

public class SharedPoolNode {
    int SharedPoolNodeNumber;
    SharedFileNode[] SharedFileList = new SharedFileNode[Globals.MAX_SF];
    int[] PermittedFrameworks = new int[Globals.MAX_FR];
    public SharedPoolNode() {super();}
}

A1.2.12 FileStructure.java

This file contains the implementation of the File Structure. Users should notice that the three examples are imbedded in this class. Next is the source code listing for this file:

package V1_FileStructure_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#1 For CO_Toolkits File Structures — Chapter 4 of My Ph.D. Thesis
 */

public class FileStructure {
    public static int F;     // NumberOfFrameworks;
    public static int[] C = new int[Globals.MAX_FR];  // to hold number of components per framework
    public static int S;     // NumberOfSharedPools;
    public static int[] SF = new int[Globals.MAX_SP];  // to hold number of shared files per shared pool
    private static String output;
    public static FrameworkNode[] CO_Fr_Files=new FrameworkNode[Globals.MAX_FR];
    public static SharedPoolNode[] CO_Sh_Files = new SharedPoolNode[Globals.MAX_SP];
    public static void display() {
        displayFrameworks();
        displaySharedPools();
    }
    public static void displayFrameworks() {
        output="";
        for (int i=1; i<= F; i++) {
            output=output+"\n\nFramework("+i+\n";
for (int j=1; j<=C[i]; j++)
    output=output+"\tComponent("+i+","+j+")\n";
for (int k=1; k<=3; k++){
    output=output+"\t"+CO_Fr_Files[i].ComponentList[j].FileList[k].ID.FileNodeNumber+":"+CO_Fr_Files[i].ComponentList[j].FileList[k].FileName+"\n";
    int tmp=1;
    while (CO_Fr_Files[i].ComponentList[j].FileList[k].IncludeList[tmp] > 0) {
        output=output+"\t"+CO_Fr_Files[i].ComponentList[j].FileList[k].IncludeList[tmp++];
    }
    output=output+"\n";
}
reportFileStructure(output);
}

public static void displaySharedPools() {
    output="";
    for (int i=1; i<=S; i++) {
        output=output+"\nSharedPool("+i+")\n";
        for (int j=1; j<=SF[i]; j++){
            output=output+"\t"+CO_Sh_Files[i].SharedFileList[j].ID.FileNodeNumber+":"+CO_Sh_Files[i].SharedFileList[j].FileType+"\n";
            int tmp=1;
            while (CO_Sh_Files[i].SharedFileList[j].IncludeList[tmp] > 0) {
                output=output+"\t"+CO_Sh_Files[i].SharedFileList[j].IncludeList[tmp++];
            }
        }
        output=output+"\nPermitted Frameworks to this SharedPool are: ";
        int ttt=0;
        while (CO_Sh_Files[i].PermittedFrameworks[++ttt]>0) {
            output=output+"\tFramework("+CO_Sh_Files[i].PermittedFrameworks[ttt]+")\n";
        }
        output=output+"\n";
    }
    reportFileStructure(output);
}

private static void frameworks_Structure_Example_1() {
    int F=2;
    C[1]=2;
    C[2]=2;// initialize the array structure ... new array then go over each element and initialize it.
    for (int i=1; i<=F; i++){
        CO_Fr_Files[i] = new FrameworkNode();
        CO_Fr_Files[i].FrameworkNodeNumber=i;
        return;
    }
}
for (int j=1; j<=C[i]; j++)
    CO_Fr_Files[i].ComponentList[j]=new ComponentNode();
    CO_Fr_Files[i].ComponentList[j].ID.ComponentNodeNumber=j;
    CO_Fr_Files[i].ComponentList[j].ID.ParentFrameworkNodeNumber=i;
    for (int k=1; k<=3; k++)
        CO_Fr_Files[i].ComponentList[j].FileList[k]=new FileNode();
        CO_Fr_Files[i].ComponentList[j].FileList[k].ID.FileNodeNumber=++Globals.NodeCount;
        CO_Fr_Files[i].ComponentList[j].FileList[k].ID.ParentComponentNodeNumber=j;
    }

// File names Assignments
// Component (1,1)
    CO_Fr_Files[1].ComponentList[1].FileList[1].FileName="file11";
    CO_Fr_Files[1].ComponentList[1].FileList[1].FileType="top";
    CO_Fr_Files[1].ComponentList[1].FileList[2].FileType="inm";
    CO_Fr_Files[1].ComponentList[1].FileList[3].FileName="file11";
    CO_Fr_Files[1].ComponentList[1].FileList[3].FileType="inl";
// Component (1,2)
    CO_Fr_Files[1].ComponentList[2].FileList[1].FileName="file12";
    CO_Fr_Files[1].ComponentList[2].FileList[1].FileType="top";
    CO_Fr_Files[1].ComponentList[2].FileList[2].FileType="inm";
    CO_Fr_Files[1].ComponentList[2].FileList[3].FileType="inl";
// Component (2,1)
    CO_Fr_Files[2].ComponentList[1].FileList[1].FileName="file21";
    CO_Fr_Files[2].ComponentList[1].FileList[1].FileType="top";
    CO_Fr_Files[2].ComponentList[1].FileList[2].FileType="inm";
    CO_Fr_Files[2].ComponentList[1].FileList[3].FileName="file21";
    CO_Fr_Files[2].ComponentList[1].FileList[3].FileType="inl";
// Component (2,2)
    CO_Fr_Files[2].ComponentList[2].FileList[1].FileType="top";
    CO_Fr_Files[2].ComponentList[2].FileList[3].FileType="inl";
// Assign Include lists
// Component (1,1) CO_Fr_Files[1].ComponentList[1].FileList[1].IncludeList[1]=2;
Appendix A1: A Prototype For CO-Toolkits's File Structures: An Overview

```java
private static void frameworks_Structure_Example_20() {
    F=3;
    C[1]=1;
    C[2]=1;
    C[3]=2;
    // initialize the array structure ... new array then go over each element and initialize it.
    for (int i=1; i<=F; i++) {
        CO_Fr_Files[i] = new FrameworkNode();
        CO_Fr_Files[i].FrameworkNodeNumber=i;
        for (int j=1; j<=C[i]; j++) {
            CO_Fr_Files[i].ComponentList[j] = new ComponentNode();
            CO_Fr_Files[i].ComponentList[j].ID.ComponentNodeNumber=j;
            CO_Fr_Files[i].ComponentList[j].ID.ParentFrameworkNodeNumber=i;
            for (int k=1; k<=3; k++) {
                CO_Fr_Files[i].ComponentList[j].FileList[k] = new FileInfo();
                CO_Fr_Files[i].ComponentList[j].FileList[k].ID.FileNodeNumber = ++Globals.NodeCount;
                CO_Fr_Files[i].ComponentList[j].FileList[k].ID.ParentComponentNodeNumber=j;
            }
        }
    }
    // File names Assignments
    // Component(1,1)
    CO_Fr_Files[1].ComponentList[1].FileList[1].FileName="file11";
    CO_Fr_Files[1].ComponentList[1].FileList[1].FileType="top";
    CO_Fr_Files[1].ComponentList[1].FileList[2].FileType="inm";
    CO_Fr_Files[1].ComponentList[1]+="top";
    CO_Fr_Files[1].ComponentList[1].FileList[2].FileType="inm";
    CO_Fr_Files[1].ComponentList[1]+="top";
    CO_Fr_Files[1].ComponentList[1].FileList[2].FileType="inm";
    CO_Fr_Files[1].ComponentList[1]+="top";
    CO_Fr_Files[1].ComponentList[1].FileList[2].FileType="inm";
    CO_Fr_Files[1].ComponentList[1]+="top";
    CO_Fr_Files[1].ComponentList[1].FileList[2].FileType="inm";
    CO_Fr_Files[1].ComponentList[1]+="top";
    CO_Fr_Files[1].ComponentList[1].FileList[2].FileType="inm";
    CO_Fr_Files[1].ComponentList[1]+="top";
    CO_Fr_Files[1].ComponentList[1].FileList[2].FileType="inm";
    CO_Fr_Files[1].ComponentList[1]+="top";
    CO_Fr_Files[1].ComponentList[1].FileList[2].FileType="inm";
    CO_Fr_Files[1].ComponentList[1]+="top";
    CO_Fr_Files[1].ComponentList[1].FileList[2].FileType="inm";
    CO_Fr_Files[1].ComponentList[1]+="top";
    CO_Fr_Files[1].ComponentList[1].FileList[2].FileType="inm";
    CO_Fr_Files[1].ComponentList[1]+="top";
    CO_Fr_Files[1].ComponentList[1].FileList[2].FileType="inm";
    // Component(1,2)
    // Component(2,1)
    // Component(2,2)
    }
```
CO_Fr_Files[1].ComponentList[1].FileList[3].FileName="file1";
CO_Fr_Files[1].ComponentList[1].FileList[3].FileType="inl";
// Component (2,1)
CO_Fr_Files[2].ComponentList[1].FileList[1].FileName="file21";
CO_Fr_Files[2].ComponentList[1].FileList[1].FileType="top";
CO_Fr_Files[2].ComponentList[1].FileList[2].FileType="inn";
CO_Fr_Files[2].ComponentList[1].FileList[3].FileName="file21";
CO_Fr_Files[2].ComponentList[1].FileList[3].FileType="inl";
// Component(3,1)
CO_Fr_Files[3].ComponentList[1].FileList[1].FileName="file31";
CO_Fr_Files[3].ComponentList[1].FileList[1].FileType="top";
CO_Fr_Files[3].ComponentList[1].FileList[2].FileName="file31";
CO_Fr_Files[3].ComponentList[1].FileList[2].FileType="inn";
CO_Fr_Files[3].ComponentList[1].FileList[3].FileName="file31";
CO_Fr_Files[3].ComponentList[1].FileList[3].FileType="inl";
// Component (3,2)
CO_Fr_Files[3].ComponentList[2].FileList[1].FileType="top";
CO_Fr_Files[3].ComponentList[2].FileList[2].FileType="inn";
CO_Fr_Files[3].ComponentList[2].FileList[3].FileType="inl";
//Assign Include lists
//Component (1,1)
// CO_Fr_Files[1][1][1].IncludeList[]={2, 3, 18, 14, -1}
// CO_Fr_Files[1][1][2].IncludeList[]={1, 3, -1}
// CO_Fr_Files[1][1][3].IncludeList[]={1, -1}
//Component (2,1)
// CO_Fr_Files[2][1][1].IncludeList[]={7, 2, -1}
// CO_Fr_Files[2][1][2].IncludeList[]={8, 7, 30, -1}
private static void frameworks_Structure_Example_30() {
    F=13;
    C[1]=9;  //ASX
    C[2]=2;  //CORBA
    C[3]=3;  //Collections
    C[4]=10; //Concurrency
    C[5]=5;  //Connection
    C[6]=5;  //IPC, only those components involved in the include are considered
    C[7]=2;  //Log_Msg
    C[8]=1;  //Memory, only one component is involved in include
    C[9]=1;  //Misc, only one component has include
}
//C[10]=7;  //Name_Service, this framework does not have includes, it will be ignored
C[10]=2;  //OS, previously C[11]
C[12]=1;  //System_V_IPC, previously 13, only one component is considered
//C[14]=2;  //Timer, this does not have includes
C[13]=1;  //Token_Services, previously C[15], only one component has include
// initialize the array structure ... new array then go over each element and initialize it.
for (int i=1; i<=F; i++)
    CO_Fr_Files[i] = new FrameworkNode();
    CO_Fr_Files[i].FrameworkNodeNumbers=i;
for (int j=1; j<=C[i]; j++)
    CO_Fr_Files[i].ComponentList[j]= new ComponentNode();
    CO_Fr_Files[i].ComponentList[j].ID = ComponentNodeNumber=j;
    CO_Fr_Files[i].ComponentList[j].ID.ParentFrameworkNodeNumber=i;
    for (int k=1; k<=3; k++)
        CO_Fr_Files[i].ComponentList[j].FileList[k]=new FileNode();
    CO_Fr_Files[i].ComponentList[j].FileList[k].ID.FileNodeNumber=k++Globals.NodeCount;
    CO_Fr_Files[i].ComponentList[j].FileList[k].ID.ParentComponentNodeNumber=j;
    CO_Fr_Files[i].ComponentList[j].FileList[k].IncludeList[1]=-1;

} frameworks_Structure_Example_3_FileNames();
frameworks_Structure_Example_3Includes();

private static void frameworks_Structure_Example_3_FileNames() {
// File names Assignments
// Component(1,1)
    CO_Fr_Files[1].ComponentList[1].FileList[1].FileName="IO_Cntl_Msg";
    CO_Fr_Files[1].ComponentList[1].FileList[1].FileType="top";
    CO_Fr_Files[1].ComponentList[1].FileList[2].FileName="IO_Cntl_Msg";
    CO_Fr_Files[1].ComponentList[1].FileList[2].FileType="inm";
    CO_Fr_Files[1].ComponentList[1].FileList[3].FileName="IO_Cntl_Msg";
    CO_Fr_Files[1].ComponentList[1].FileList[3].FileType="inl";
// Component(1,2)
    CO_Fr_Files[1].ComponentList[2].FileList[1].FileName="Map_Manager";
    CO_Fr_Files[1].ComponentList[2].FileList[1].FileType="top";
    CO_Fr_Files[1].ComponentList[2].FileList[2].FileName="Map_Manager";
    CO_Fr_Files[1].ComponentList[2].FileList[2].FileType="inm";
    CO_Fr_Files[1].ComponentList[2].FileList[3].FileName="Map_Manager";
    CO_Fr_Files[1].ComponentList[2].FileList[3].FileType="inl";
// Component(1,3)
    CO_Fr_Files[1].ComponentList[3].FileList[1].FileName="Message_Block";
    CO_Fr_Files[1].ComponentList[3].FileList[1].FileType="top";
    CO_Fr_Files[1].ComponentList[3].FileList[2].FileType="inm";
    CO_Fr_Files[1].ComponentList[3].FileList[3].FileName="Message_Block";
    CO_Fr_Files[1].ComponentList[3].FileList[3].FileType="inl";
// Component(1,4)
CO_Fr_Files[1].ComponentList[4].FileList[1].FileName = "Message_Queue";
CO_Fr_Files[1].ComponentList[4].FileList[1].FileType = "top";
CO_Fr_Files[1].ComponentList[4].FileList[2].FileType = "inm";
CO_Fr_Files[1].ComponentList[4].FileList[3].FileType = "inl";

// Component(1,5)
CO_Fr_Files[1].ComponentList[5].FileList[1].FileName = "Module";
CO_Fr_Files[1].ComponentList[5].FileList[1].FileType = "top";
CO_Fr_Files[1].ComponentList[5].FileList[2].FileName = "Module";
CO_Fr_Files[1].ComponentList[5].FileList[2].FileType = "inm";
CO_Fr_Files[1].ComponentList[5].FileList[3].FileName = "Module";
CO_Fr_Files[1].ComponentList[5].FileList[3].FileType = "inl";

// Component(1,6)
CO_Fr_Files[1].ComponentList[6].FileList[1].FileName = "Multiplexor";
CO_Fr_Files[1].ComponentList[6].FileList[1].FileType = "top";
CO_Fr_Files[1].ComponentList[6].FileList[2].FileName = "Multiplexor";
CO_Fr_Files[1].ComponentList[6].FileList[2].FileType = "inm";
CO_Fr_Files[1].ComponentList[6].FileList[3].FileName = "Multiplexor";
CO_Fr_Files[1].ComponentList[6].FileList[3].FileType = "inl";

// Component(1,7)
CO_Fr_Files[1].ComponentList[7].FileList[1].FileName = "Stream";
CO_Fr_Files[1].ComponentList[7].FileList[1].FileType = "top";
CO_Fr_Files[1].ComponentList[7].FileList[2].FileName = "Stream";
CO_Fr_Files[1].ComponentList[7].FileList[2].FileType = "inm";
CO_Fr_Files[1].ComponentList[7].FileList[3].FileName = "Stream";
CO_Fr_Files[1].ComponentList[7].FileList[3].FileType = "inl";

// Component(1,8)
CO_Fr_Files[1].ComponentList[8].FileList[1].FileName = "Stream_Modules";
CO_Fr_Files[1].ComponentList[8].FileList[1].FileType = "top";
CO_Fr_Files[1].ComponentList[8].FileList[2].FileType = "inm";
CO_Fr_Files[1].ComponentList[8].FileList[3].FileType = "inl";

// Component(1,9)
CO_Fr_Files[1].ComponentList[9].FileList[1].FileName = "Task";
CO_Fr_Files[1].ComponentList[9].FileList[1].FileType = "top";
CO_Fr_Files[1].ComponentList[9].FileList[2].FileType = "inm";
CO_Fr_Files[1].ComponentList[9].FileList[3].FileName = "Task";
CO_Fr_Files[1].ComponentList[9].FileList[3].FileType = "inl";

// Component (2,1)
CO_Fr_Files[2].ComponentList[1].FileList[1].FileName = "CORBA_Handler";
CO_Fr_Files[2].ComponentList[1].FileList[1].FileType = "top";
CO_Fr_Files[2].ComponentList[1].FileList[2].FileName = "CORBA_Handler";
CO_Fr_Files[2].ComponentList[1].FileList[2].FileType = "inm";
Appendix A1: A Prototype For CO-Toolkits's File Structures: An Overview

CO_Fr_Files[2].ComponentList[1].FileList[3].FileName="CORBA_Handler";
CO_Fr_Files[2].ComponentList[1].FileList[3].FileType="inl";
// Component (2,2)
CO_Fr_Files[2].ComponentList[2].FileList[1].FileName="CORBA_Ref";
CO_Fr_Files[2].ComponentList[2].FileList[1].FileType="top";
CO_Fr_Files[2].ComponentList[2].FileList[2].FileName="CORBA_Ref";
CO_Fr_Files[2].ComponentList[2].FileList[3].FileName="CORBA_Ref";
CO_Fr_Files[2].ComponentList[2].FileList[3].FileType="inl";
// Component(3,1)
CO_Fr_Files[3].ComponentList[1].FileList[1].FileName="Set";
CO_Fr_Files[3].ComponentList[1].FileList[1].FileType="top";
CO_Fr_Files[3].ComponentList[1].FileList[2].FileType="inm";
CO_Fr_Files[3].ComponentList[1].FileList[3].FileName="Set";
CO_Fr_Files[3].ComponentList[1].FileList[3].FileType="inl";
// Component (3,2)
CO_Fr_Files[3].ComponentList[2].FileList[1].FileType="top";
CO_Fr_Files[3].ComponentList[2].FileList[2].FileType="inm";
CO_Fr_Files[3].ComponentList[2].FileList[3].FileType="inl";
// Component (3,3)
CO_Fr_Files[3].ComponentList[3].FileList[1].FileName="SSString";
CO_Fr_Files[3].ComponentList[3].FileList[1].FileType="top";
CO_Fr_Files[3].ComponentList[3].FileList[2].FileName="SSString";
CO_Fr_Files[3].ComponentList[3].FileList[2].FileType="inm";
CO_Fr_Files[3].ComponentList[3].FileList[3].FileName="SSString";
CO_Fr_Files[3].ComponentList[3].FileList[3].FileType="inl";
// Component(4,1)
CO_Fr_Files[4].ComponentList[1].FileList[1].FileName="Activation_Queue";
CO_Fr_Files[4].ComponentList[1].FileList[1].FileType="top";
CO_Fr_Files[4].ComponentList[1].FileList[2].FileType="inm";
CO_Fr_Files[4].ComponentList[1].FileList[3].FileName="Activation_Queue";
CO_Fr_Files[4].ComponentList[1].FileList[3].FileType="inl";
// Component (4,2)
CO_Fr_Files[4].ComponentList[2].FileList[1].FileType="top";
CO_Fr_Files[4].ComponentList[2].FileList[3].FileType="inl";
// Component (4,3)
CO_Fr_Files[4].ComponentList[3].FileList[1].FileType="top";

172
Appendix A1: A Prototype For CO-Toolkits's File Structures: An Overview

CO_Fr_Files[4].ComponentList[3].FileList[2].FileType="inn";
CO_Fr_Files[4].ComponentList[3].FileList[3].FileType="inl";
// Component(4,4)
CO_Fr_Files[4].ComponentList[4].FileList[1].FileName="Process_Manager";
CO_Fr_Files[4].ComponentList[4].FileList[1].FileType="top";
CO_Fr_Files[4].ComponentList[4].FileList[2].FileType="inn";
CO_Fr_Files[4].ComponentList[4].FileList[3].FileType="inl";
// Component (4,5)
CO_Fr_Files[4].ComponentList[5].FileList[1].FileName="Synch";
CO_Fr_Files[4].ComponentList[5].FileList[1].FileType="top";
CO_Fr_Files[4].ComponentList[5].FileList[2].FileType="inn";
CO_Fr_Files[4].ComponentList[5].FileList[3].FileName="Synch";
CO_Fr_Files[4].ComponentList[5].FileList[3].FileType="inl";
// Component (4,6)
CO_Fr_Files[4].ComponentList[6].FileList[1].FileName="Synch_Options";
CO_Fr_Files[4].ComponentList[6].FileList[1].FileType="top";
CO_Fr_Files[4].ComponentList[6].FileList[2].FileType="inn";
CO_Fr_Files[4].ComponentList[6].FileList[3].FileName="Synch_Options";
CO_Fr_Files[4].ComponentList[6].FileList[3].FileType="inl";
// Component (4,7)
CO_Fr_Files[4].ComponentList[7].FileList[1].FileName="Synch_T";
CO_Fr_Files[4].ComponentList[7].FileList[1].FileType="top";
CO_Fr_Files[4].ComponentList[7].FileList[2].FileName="Synch_T";
CO_Fr_Files[4].ComponentList[7].FileList[2].FileType="inn";
CO_Fr_Files[4].ComponentList[7].FileList[3].FileName="Synch_T";
CO_Fr_Files[4].ComponentList[7].FileList[3].FileType="inl";
// Component (4,8)
CO_Fr_Files[4].ComponentList[8].FileList[1].FileName="Thread";
CO_Fr_Files[4].ComponentList[8].FileList[1].FileType="top";
CO_Fr_Files[4].ComponentList[8].FileList[2].FileName="Thread";
CO_Fr_Files[4].ComponentList[8].FileList[2].FileType="inn";
CO_Fr_Files[4].ComponentList[8].FileList[3].FileName="Thread";
CO_Fr_Files[4].ComponentList[8].FileList[3].FileType="inl";
// Component(4,9)
CO_Fr_Files[4].ComponentList[9].FileList[1].FileName="Thread_Manager";
CO_Fr_Files[4].ComponentList[9].FileList[1].FileType="top";
CO_Fr_Files[4].ComponentList[9].FileList[2].FileName="Thread_Manager";
CO_Fr_Files[4].ComponentList[9].FileList[2].FileType="inn";
CO_Fr_Files[4].ComponentList[9].FileList[3].FileName="Thread_Manager";
CO_Fr_Files[4].ComponentList[9].FileList[3].FileType="inl";
// Component (4,10)

173
// Component (6, 2)
CO_Fr_Files[6].ComponentList[2].FileList[1].FileName = "SOCK_Acceptor";
CO_Fr_Files[6].ComponentList[2].FileList[1].FileType = "inm";
CO_Fr_Files[6].ComponentList[2].FileList[2].FileName = "SOCK_Acceptor";
CO_Fr_Files[6].ComponentList[2].FileList[2].FileType = "inm";
CO_Fr_Files[6].ComponentList[2].FileList[3].FileName = "SOCK_Acceptor";
CO_Fr_Files[6].ComponentList[2].FileList[3].FileType = "inl";

// Component (6, 3)
CO_Fr_Files[6].ComponentList[3].FileList[1].FileName = "SOCK_Connector";
CO_Fr_Files[6].ComponentList[3].FileList[1].FileType = "top";
CO_Fr_Files[6].ComponentList[3].FileList[2].FileName = "SOCK_Connector";
CO_Fr_Files[6].ComponentList[3].FileList[2].FileType = "inm";
CO_Fr_Files[6].ComponentList[3].FileList[3].FileName = "SOCK_Connector";
CO_Fr_Files[6].ComponentList[3].FileList[3].FileType = "inl";

// Component (6, 4)
CO_Fr_Files[6].ComponentList[4].FileList[1].FileName = "SOCK_Stream";
CO_Fr_Files[6].ComponentList[4].FileList[1].FileType = "top";
CO_Fr_Files[6].ComponentList[4].FileList[2].FileName = "SOCK_Stream";
CO_Fr_Files[6].ComponentList[4].FileList[2].FileType = "inm";
CO_Fr_Files[6].ComponentList[4].FileList[3].FileName = "SOCK_Stream";
CO_Fr_Files[6].ComponentList[4].FileList[3].FileType = "inl";

// Component (6, 5)
CO_Fr_Files[6].ComponentList[5].FileList[1].FileName = "SPIPE_Connector";
CO_Fr_Files[6].ComponentList[5].FileList[1].FileType = "top";
CO_Fr_Files[6].ComponentList[5].FileList[2].FileName = "SPIPE_Connector";
CO_Fr_Files[6].ComponentList[5].FileList[2].FileType = "inm";
CO_Fr_Files[6].ComponentList[5].FileList[3].FileName = "SPIPE_Connector";
CO_Fr_Files[6].ComponentList[5].FileList[3].FileType = "inl";

// Component (7, 1)
CO_Fr_Files[7].ComponentList[1].FileList[1].FileName = "Log_Msg";
CO_Fr_Files[7].ComponentList[1].FileList[1].FileType = "top";
CO_Fr_Files[7].ComponentList[1].FileList[2].FileName = "Log_Msg";
CO_Fr_Files[7].ComponentList[1].FileList[2].FileType = "inm";
CO_Fr_Files[7].ComponentList[1].FileList[3].FileName = "Log_Msg";
CO_Fr_Files[7].ComponentList[1].FileList[3].FileType = "inl";

// Component (7, 2)
CO_Fr_Files[7].ComponentList[2].FileList[1].FileName = "Log_Record";
CO_Fr_Files[7].ComponentList[2].FileList[1].FileType = "top";
CO_Fr_Files[7].ComponentList[2].FileList[2].FileName = "Log_Record";
CO_Fr_Files[7].ComponentList[2].FileList[2].FileType = "inm";
CO_Fr_Files[7].ComponentList[2].FileList[3].FileName = "Log_Record";
CO_Fr_Files[7].ComponentList[2].FileList[3].FileType = "inl";

// Component (8, 1)
CO_Fr_Files[8].ComponentList[1].FileList[1].FileName = "Malloc";
CO_Fr_Files[8].ComponentList[1].FileList[1].FileType = "top";
CO_Fr_Files[8].ComponentList[1].FileList[2].FileName = "Malloc";
CO_Fr_Files[8].ComponentList[1].FileList[2].FileType="inm";
CO_Fr_Files[8].ComponentList[1].FileList[3].FileName="Malloc";
CO_Fr_Files[8].ComponentList[1].FileList[3].FileType="inl";

// Component (9,1)
CO_Fr_Files[9].ComponentList[1].FileList[1].FileName="Dynamic";
CO_Fr_Files[9].ComponentList[1].FileList[1].FileType="top";
CO_Fr_Files[9].ComponentList[1].FileList[2].FileName="Dynamic";
CO_Fr_Files[9].ComponentList[1].FileList[2].FileType="inm";
CO_Fr_Files[9].ComponentList[1].FileList[3].FileName="Dynamic";
CO_Fr_Files[9].ComponentList[1].FileList[3].FileType="inl";

// Component (10,1)
CO_Fr_Files[10].ComponentList[1].FileList[1].FileName="ACE";
CO_Fr_Files[10].ComponentList[1].FileList[1].FileType="top";
CO_Fr_Files[10].ComponentList[1].FileList[2].FileName="ACE";
CO_Fr_Files[10].ComponentList[1].FileList[2].FileType="inm";
CO_Fr_Files[10].ComponentList[1].FileList[3].FileName="ACE";
CO_Fr_Files[10].ComponentList[1].FileList[3].FileType="inl";

// Component (10,2)
CO_Fr_Files[10].ComponentList[2].FileList[1].FileName="OS";
CO_Fr_Files[10].ComponentList[2].FileList[1].FileType="top";
CO_Fr_Files[10].ComponentList[2].FileList[2].FileName="OS";
CO_Fr_Files[10].ComponentList[2].FileList[2].FileType="inm";
CO_Fr_Files[10].ComponentList[2].FileList[3].FileName="OS";
CO_Fr_Files[10].ComponentList[2].FileList[3].FileType="inl";

// Component(11,1)

// Component(11,2)

// Component(11,3)

// Component(11,4)
// Component(11,5)
// Component(11,6)
// Component(11,7)
CO_Fr_Files[11].ComponentList[7].FileList[1].FileType="top";
CO_Fr_Files[11].ComponentList[7].FileList[3].FileType="inl";
// Component(11,8)
// Component(11,9)
// Component(12,1)
CO_Fr_Files[12].ComponentList[1].FileList[1].FileName="SV_Semaphore_Complex";
CO_Fr_Files[12].ComponentList[1].FileList[1].FileType="top";
CO_Fr_Files[12].ComponentList[1].FileList[2].FileName="SV_Semaphore_Complex";
CO_Fr_Files[12].ComponentList[1].FileList[2].FileType="inm";
CO_Fr_Files[12].ComponentList[1].FileList[3].FileName="SV_Semaphore_Complex";
CO_Fr_Files[12].ComponentList[1].FileList[3].FileType="inl";
177
private static void frameworks_Structure_Example_3_Includes() {
        // Assign Include lists
        // Component (1.1) No included files
        // Component (1.2)
        // Component (1.3)
        // Component (1.4)
        // Component (1.5)
Appendix A1: A Prototype For CO-Toolkit's File Structures: An Overview


//Component (1,7) CO_Fr_Files[1].ComponentList[7].FileList[1].IncludeList[1]=14;

// CO_Fr_Files[1][8][2].IncludeList[]=26,24,22,-1


Appendix A1: A Prototype For CO-Toolkit's File Structures: An Overview

Appendix A1: A Prototype For CO-Toolkit's File Structures: An Overview

CO_Fr_Files[7].ComponentList[1].FileList[1].IncludeList[8]=1;
public static void initializeFor_Example_1() {
    frameworks_Structure_Example_1();
}

184
sharedPools_Structure_Example_10; 
}

public static void initializeFor_Example_1_OLD() 
{ 
    frameworks_Structure_Example_10; 
    sharedPools_Structure_Example_10; 
}

public static void initializeFor_Example_20() 
{ 
    frameworks_Structure_Example_20; 
    sharedPools_Structure_Example_20; 
}

public static void initializeFor_Example_30() 
{ 
    frameworks_Structure_Example_30; 
    sharedPools_Structure_Example_30; 
}

public static void reportFileStructure(String output) 
{ 
    System.out.println("\n"+output+"\n"); 
    return; 
}

private static void sharedPools_Structure_Example_10() 
{ 
    S=2; 
    SF[1]=2; 
    SF[2]=1; 
    // initialize the array structure ... new array then go over each element and initialize it. 
    for (int i=1;i<=S;i++) 
    { 
        CO_Sh_Files[i]=new SharedPoolNode(); 
        CO_Sh_Files[i].SharedPoolNodeNumber=i; 
        for (int j=1;j<=SF[i];j++) 
        { 
            CO_Sh_Files[i].SharedFileList[j]=new SharedFileNode(); 
            CO_Sh_Files[i].SharedFileList[j].ID=ParentSharedPoolNodeNumber=i; 
            CO_Sh_Files[i].SharedFileList[j].ID.FileNumber=++Globals.NodeCount; 
        }
    }
    //Assignments — Permitted frameworks 
    // SharedPool(1) 
    CO_Sh_Files[1].PermittedFrameworks[1]=1; 
    CO_Sh_Files[1].PermittedFrameworks[2]=2; 
    CO_Sh_Files[1].PermittedFrameworks[3]=1; 
    // SharedPool(2) 
    CO_Sh_Files[2].PermittedFrameworks[1]=1; 
    //Assignments — File names 
    // SharedPool(1) 
    CO_Sh_Files[1].SharedFileList[1].FileName="sh_file1"; 
    CO_Sh_Files[1].SharedFileList[1].FileType="s"; 
    CO_Sh_Files[1].SharedFileList[2].FileName="sh_file2"; 
    CO_Sh_Files[1].SharedFileList[2].FileType="s";
Appendix A1: A Prototype For CO-Toolkits’s File Structures: An Overview

// SharedPool(2)
CO_Sh_Files[2].SharedFileList[1].FileName="sh_file21";
CO_Sh_Files[2].SharedFileList[1].FileType="s";
//Assign Include lists
//SharedPool(1)
CO_Sh_Files[1].SharedFileList[1].IncludeList[1]=14;
CO_Sh_Files[1].SharedFileList[2].IncludeList[1]=1;
//SharedPools(2)
CO_Sh_Files[2].SharedFileList[1].IncludeList[1]=1;
}
private static void sharedPools_Structure_Example_1_OLD() {
    S=2;
    SF[1]=2;
    SF[2]=1;
    // initialize the array structure ... new array then go over each element and initialize it.
    for (int i=1;i<=S;i++){
        CO_Sh_Files[i]=new SharedPoolNode();
        CO_Sh_Files[i].SharedPoolNodeNumber=i;
        for (int j=1;j<=SF[i];j++){
            CO_Sh_Files[i].SharedFileList[j]=new SharedFileNode();
            CO_Sh_Files[i].SharedFileList[j].ID.ParentSharedPoolNodeNumber=i;
            CO_Sh_Files[i].SharedFileList[j].ID.FileNodeNumber+=Globals.NodeCount;
        }
    }
    //Assignments —— Permitted frameworks
    //SharedPool(1)
    CO_Sh_Files[1].PermittedFrameworks[1]=1;
    CO_Sh_Files[1].PermittedFrameworks[2]=2;
    CO_Sh_Files[1].PermittedFrameworks[3]=1;
    //SharedPool(2)
    CO_Sh_Files[2].PermittedFrameworks[1]=1;
    //Assignments —— File names
    //SharedPool(1)
    CO_Sh_Files[1].SharedFileList[1].FileName="sh_file11";
    CO_Sh_Files[1].SharedFileList[1].FileType="s";
    CO_Sh_Files[1].SharedFileList[2].FileName="sh_file12";
    CO_Sh_Files[1].SharedFileList[2].FileType="s";
    //SharedPool(2)
    CO_Sh_Files[2].SharedFileList[1].FileName="sh_file21";
    CO_Sh_Files[2].SharedFileList[2].FileType="s";
    //Assign Include lists
    //SharedPool(1)
    CO_Sh_Files[1].SharedFileList[1].IncludeList[3]=1;
}
//SharedPools(2)

private static void sharedPools_Structure_Example_20()
{
    S=2;
    SF[1]=2;
    SF[2]=1;
    // initialize the array structure ... new array then go over each element and initialize it.
    for (int i=1;i<=S;i++)
    {
        CO_Sh_Files[i]=new SharedPoolsNode();
        CO_Sh_Files[i].SharedPoolNumber=i;
        for (int j=1;j<=SF[i];j++)
        {
            CO_Sh_Files[i].SharedFileList[j]=new SharedFileNode();
            CO_Sh_Files[i].SharedFileList[j].ID.ParentSharedPoolNodeNumber=i;
            CO_Sh_Files[i].SharedFileList[j].ID.FileNumber=++Globals.NodeCount;
        }
    }

    //Assignments — Permitted frameworks
    // SharedPool(1)
    CO_Sh_Files[1].PermittedFrameworks[1]=1;
    CO_Sh_Files[1].PermittedFrameworks[2]=2;
    CO_Sh_Files[1].PermittedFrameworks[3]=-1;

    // SharedPool(2)
    CO_Sh_Files[2].PermittedFrameworks[1]=2;
    CO_Sh_Files[2].PermittedFrameworks[2]=3;

    //Assignments — File names
    // SharedPool(1)
    CO_Sh_Files[1].SharedFileList[1].FileName="sh_file11";
    CO_Sh_Files[1].SharedFileList[1].FileType="s";
    CO_Sh_Files[1].SharedFileList[2].FileName="sh_file12";
    CO_Sh_Files[1].SharedFileList[2].FileType="s";

    // SharedPool(2)
    CO_Sh_Files[2].SharedFileList[1].FileName="sh_file21";
    CO_Sh_Files[2].SharedFileList[1].FileType="s";

    //Assign Include lists
    //SharedPool(1)
    CO_Sh_Files[1].SharedFileList[1].IncludeList[3]=1;
// SharedPools(2)
}
private static void sharedPools_Structure_Example_30 {
  S = 1;
  SF[1] = 7;
  // initialize the array structure ... new array then go over each element and initialize it.
  for (int i = 1; i <= S; i++) {
    CO_Sh_Files[i] = new SharedPoolNode();
    CO_Sh_Files[i].SharedPoolNodeNumber = i;
    for (int j = 1; j <= SF[i]; j++) {
      CO_Sh_Files[i].SharedFileList[j] = new SharedFileNode();
      CO_Sh_Files[i].SharedFileList[j].ID.PartialSharedPoolNodeNumber = i;
      CO_Sh_Files[i].SharedFileList[j].ID.FileNodeNumber = ++Globals.NodeCount;
      CO_Sh_Files[i].SharedFileList[j].IncludeList[1] = -1;
    }
  }
  // Assignments —— Permitted frameworks
  // SharedPool(1)
  CO_Sh_Files[1].PermittedFrameworks[1] = 1;
  CO_Sh_Files[1].PermittedFrameworks[2] = 2;
  CO_Sh_Files[1].PermittedFrameworks[3] = 10;
  CO_Sh_Files[1].PermittedFrameworks[4] = -1;
  // Assignments —— File names
  // SharedPool(1)
  CO_Sh_Files[1].SharedFileList[1].FileName = "config";
  CO_Sh_Files[1].SharedFileList[1].FileType = "s";
  CO_Sh_Files[1].SharedFileList[2].FileName = "Service_Config";
  CO_Sh_Files[1].SharedFileList[2].FileType = "s";
  CO_Sh_Files[1].SharedFileList[3].FileName = "Pipe";
  CO_Sh_Files[1].SharedFileList[3].FileType = "s";
  CO_Sh_Files[1].SharedFileList[4].FileName = "CORBA";
  CO_Sh_Files[1].SharedFileList[4].FileType = "s";
  CO_Sh_Files[1].SharedFileList[5].FileName = "daemon";
  CO_Sh_Files[1].SharedFileList[5].FileType = "s";
  CO_Sh_Files[1].SharedFileList[6].FileName = "Service_Object";
  CO_Sh_Files[1].SharedFileList[6].FileType = "s";
  CO_Sh_Files[1].SharedFileList[7].FileName = "Service_Repository";
  CO_Sh_Files[1].SharedFileList[7].FileType = "s";
  // Assign Include lists
  // SharedPool(1)
  // None
}
}
Appendix A2

A Prototype For CO–Toolkits’ File Structures: Example#1 – An Artificial Example

A2.1 Overview

This Appendix presents the first artificial example that has been used to demonstrate our prototype for CO–Toolkits’ file structures. Section A2.3 shows that this example passes the prototype test (i.e. each predicate returns STATUS=true).

A2.2 Example#1: File Structure

FILE STRUCTURE: As Extracted From The Current Example

Framework(1)
Component(1,1)
  FileNode(1):  file11.top  Includes FileNodes:  2  13  15
  FileNode(2):  file11.inm  Includes FileNodes:  3  5
  FileNode(3):  file11.inl  Includes FileNodes:
Component(1,2)
  FileNode(4):  file12.top  Includes FileNodes:  5
  FileNode(5):  file12.inm  Includes FileNodes:  6
  FileNode(6):  file12.inl  Includes FileNodes:
Framework(2)
Component(2,1)
  FileNode(7):  file21.top  Includes FileNodes:  8
  FileNode(8):  file21.inm  Includes FileNodes:  9
  FileNode(9):  file21.inl  Includes FileNodes:
Component(2,2)
  FileNode(10):  file22.top  Includes FileNodes:  11  13  14
  FileNode(11):  file22.inm  Includes FileNodes:  12  8
  FileNode(12):  file22.inl  Includes FileNodes:
A2.3 Example#1: Output Report

Checking for GoodFileSystem_predicate4_5: GoodFileSystem = predicates 4_6, 4_7, and 4_8

Checking for Predicate 4_6: OnlyFourFileCategories

Checking for Predicate 4_5 is complete — STATUS: true

_____________________________________________________

Checking for Predicate 4_7: GoodDistinctionOfFile_4_9, and 4_10

Checking for Predicate 4_9: Fr_SetsOfFilesShouldBeDistinct

Checking for Predicate 4_9 is complete — STATUS: true

_____________________________________________________

Checking for Predicate 4_10: ComponentsMainFilesShouldBeDistinct
Checking for Predicate 4_10 is complete — STATUS: true

_____________________________________________________

Checking for Predicate 4_7 is complete — STATUS: true

_____________________________________________________

Checking for Predicate 4_8: GoodFileInclusionRelation = predicates 4_11, 4_12, and 4_13

Checking for Predicate 4_11: GoodUseOfFile_4_14, 4_15, 4_16, and 4_17

Checking for Predicate 4_12: InclusionLoopsMayNotExist

Checking for Predicate 4_12 is complete — STATUS: true

_____________________________________________________

Checking for Predicate 4_13: GoodUseOf_top_Files = predicates 4_18, 4_19, and 4_20
Checking for predicate 4_17: NoFileMayIncludeA_top_File

Checking for predicate 4_17 is complete —— STATUS: true

Checking for predicate 4_18: FromSame_Fr_A_top_FileMayIncludeOnlyIts_intermediate_File

Checking for predicate 4_18 is complete —— STATUS: true

Checking for predicate 4_19: A_top_FileMayInclude_shared_FilesFromPermittedSharedPools

Checking for predicate 4_19 is complete —— STATUS: true

Checking for Predicate 4_13 is complete —— STATUS: true

Checking for Predicate 4_14: GoodUseOf_intermediate_Files = predicates 4_21 and 4_22

Checking for predicate 4_20: An_intermediate_fileMayIncludeOnlyIts_internal_FileAndMayIncludeOther_intermediate_FilesFromSame_Fr

Checking for predicate 4_20 is complete —— STATUS: true

Checking for Predicate 4_14 is complete —— STATUS: true

Checking for Predicate 4_15: GoodUseOf_internal_Files = predicates 4_23 and 4_24

Checking for predicate 4_21: An_internal_FileMayNotIncludeAnyFile

Checking for predicate 4_21 is complete —— STATUS: true

Checking for predicate 4_22: NoFileMayIncludeAn_Internal_FileExceptIts_intermediate_File

Checking for predicate 4_22 is complete —— STATUS: true

Checking for Predicate 4_15 is complete —— STATUS: true

Checking for Predicate 4_11 is complete —— STATUS: true

Checking for Predicate 4_12: InclusionLoopsMayNotExist
Checking for Predicate 4.12 is complete — STATUS: true

Checking for Predicate 4.8 is complete — STATUS: true

Checking for GoodFileStructure_predicate4.5 is complete — STATUS: true
Appendix A3

VA Prototype For CO–Toolkits’ File Structures: Example#2 – An Artificial Example

A3.1 Overview

This Appendix presents the second artificial example that has been used to demonstrate our prototype for CO–Toolkits’ file structures. Section A3.3 presents the rules violations reported by this prototype for this example.

A3.2 Example#2: File Structure

FILE STRUCTURE: As Extracted From The Current Example

Framework(1)
  Component(1,1)
    FileNode(1): file11.top includes FileNodes: 2 3 18 14
    FileNode(2): file11.inm includes FileNodes: 1 3
    FileNode(3): file11.inl includes FileNodes: 1

Framework(2)
  Component(2,1)
    FileNode(4): file21.top includes FileNodes: 7 2
    FileNode(5): file21.inm includes FileNodes: 8 7 30
    FileNode(6): file21.inl includes FileNodes: 8

Framework(3)
  Component(3,1)
    FileNode(7): file31.top includes FileNodes: 8 19 11 14
    FileNode(8): file31.inm includes FileNodes: 9 10 11
    FileNode(9): file31.inl includes FileNodes: 10
  Component(3,2)
    FileNode(10): file12.top includes FileNodes: 5 8 9
    FileNode(11): file12.inm includes FileNodes: 12 10 8
    FileNode(12): file12.inl includes FileNodes: 7 4 8

SharedPool(1)
  FileNode(13): sh_file11.a includes FileNodes: 11 16
  FileNode(14): sh_file12.a includes FileNodes: 14 16
  Permitted Frameworks to this SharedPool are: Framework(1) Framework(2)

SharedPool(2)
  FileNode(15): sh_file21.a includes FileNodes: 13 14
  Permitted Frameworks to this SharedPool are: Framework(2) Framework(3)

FILE STRUCTURE display is DONE
A3.3 Example#2: Output Report

Checking for GoodFileStructure.predicate_4.5: GoodFileStructure = predicates 4.6, 4.7, and 4.8.
Checking for Predicate 4.6: OnlyFourFileCategories
Checking for Predicate 4.6 is complete — STATUS: true

Checking for Predicate 4.7: GoodDistinctionOfFileset = Predicates 4.9, and 4.10
Checking for Predicate 4.9: Pr_SetOfFilesShouldBeDistinct
Checking for Predicate 4.9 is complete — STATUS: true

Checking for Predicate 4.10: ComponentsMainFilesShouldBeDistinct
Checking for Predicate 4.10 is complete — STATUS: true

Checking for Predicate 4.7 is complete — STATUS: true

Checking for Predicate 4.8: GoodFileInclusionRelation = predicates 4.11, 4.12, and 4.13
Checking for Predicate 4.11: GoodUseOfFiles = predicates 4.14, 4.15, 4.16, and 4.17
Checking for Predicate 4.12: InclusionLoopsMayNotExist
ERROR(Predicate_13): File file11.top with FileNode(1) from Framework(1) and Component(1.1) includes FileNode(18) that has no existing file
ERROR(Predicate_13): File file12.inn with FileNode(5) from Framework(2) and Component(2.1) includes FileNode(30) that has no existing file
ERROR(Predicate_13): File file31.top with FileNode(7) from Framework(3) and Component(3.1) includes FileNode(19) that has no existing file
ERROR(Predicate_13): SharedFile sh_file11.top with FileNode(13) from SharedPool(1) includes FileNode(16) that has no existing file
This illegal FileNode will be ignored
ERROR(Predicate_13): SharedFile sh_file12.inn with FileNode(4) from SharedPool(1) includes FileNode(16) that has no existing file
This illegal FileNode will be ignored

VIOLATION(Predicate_13): FileInclusionLOOP has been detected:
1 —> 2 —> 1
DETAILS:
NodeNumber(1) FileNode(file11.top) Framework(1) Component(1.1)
NodeNumber(2) FileNode(file11.inn) Framework(1) Component(1.1)
NodeNumber(1) FileNode(file11.top) Framework(1) Component(1.1)

VIOLATION(Predicate_13): FileInclusionLOOP has been detected:
1 —> 2 —> 3 —> 1
DETAILS:
NodeNumber(1) FileNode(file11.top) Framework(1) Component(1.1)
NodeNumber(2) FileNode(file11.inn) Framework(1) Component(1.1)
NodeNumber(3) FileNode(file11.inn) Framework(1) Component(1.1)
NodeNumber(1) FileNode(file11.top) Framework(1) Component(1.1)
NodeNumber(1) FileNode(file11.inn) Framework(1) Component(1.1)

VIOLATION(Predicate_13): FileInclusionLOOP has been detected:
1 —> 3 —> 1
DETAILS:
NodeNumber(1) FileNode(file11.top) Framework(1) Component(1.1)
NodeNumber(3) FileNode(file11.inn) Framework(1) Component(1.1)
NodeNumber(1) FileNode(file11.top) Framework(1) Component(1.1)

VIOLATION(Predicate_13): FileInclusionLOOP has been detected:
2 —> 1 —> 2
DETAILS:
NodeNumber(2) FileNode(file11.inn) Framework(1) Component(1.1)
NodeNumber(1) FileNode(file11.top) Framework(1) Component(1.1)
NodeNumber(2) FileNode(file11.inn) Framework(1) Component(1.1)
Appendix A3: A Prototype For CO-Toolkits: File Structures: Example#2 – An Artificial Example
Appendix A3: A Prototype For CO-Toolkits' File Structures: Example #2 – An Artificial Example

VIOLATION Predicate: FileInclusion LOOP has been detected:

5 \rightarrow 7 \rightarrow 11 \rightarrow 8 \rightarrow 10 \rightarrow 5

DETAILS:

NodeNumber(5) FileName(file21.ss) Framework(2) Component(2.1)
NodeNumber(7) FileName(file31.aop) Framework(3) Component(3.1)
NodeNumber(11) FileName(file12.ss) Framework(3) Component(3.2)
NodeNumber(10) FileName(file12.aop) Framework(3) Component(3.2)
NodeNumber(5) FileName(file21.ss) Framework(2) Component(2.1)

VIOLATION Predicate: FileInclusion LOOP has been detected:

5 \rightarrow 7 \rightarrow 11 \rightarrow 10 \rightarrow 5

DETAILS:

NodeNumber(5) FileName(file21.ss) Framework(2) Component(2.1)
NodeNumber(7) FileName(file31.aop) Framework(3) Component(3.1)
NodeNumber(11) FileName(file12.ss) Framework(3) Component(3.2)
NodeNumber(10) FileName(file12.aop) Framework(3) Component(3.2)
NodeNumber(5) FileName(file21.ss) Framework(2) Component(2.1)

VIOLATION Predicate: FileInclusion LOOP has been detected:

5 \rightarrow 7 \rightarrow 11 \rightarrow 12 \rightarrow 8 \rightarrow 9 \rightarrow 10 \rightarrow 5

DETAILS:

NodeNumber(5) FileName(file21.ss) Framework(2) Component(2.1)
NodeNumber(7) FileName(file31.aop) Framework(3) Component(3.1)
NodeNumber(11) FileName(file12.ss) Framework(3) Component(3.2)
NodeNumber(12) FileName(file12.aop) Framework(2) Component(2.1)
NodeNumber(10) FileName(file12.aop) Framework(3) Component(3.2)
NodeNumber(5) FileName(file21.ss) Framework(2) Component(2.1)

VIOLATION Predicate: FileInclusion LOOP has been detected:

5 \rightarrow 7 \rightarrow 11 \rightarrow 12 \rightarrow 8 \rightarrow 10 \rightarrow 5

DETAILS:

NodeNumber(5) FileName(file21.ss) Framework(2) Component(2.1)
NodeNumber(7) FileName(file31.aop) Framework(3) Component(3.1)
NodeNumber(11) FileName(file12.ss) Framework(3) Component(3.2)
NodeNumber(12) FileName(file12.aop) Framework(2) Component(2.1)
NodeNumber(10) FileName(file12.aop) Framework(3) Component(3.2)
NodeNumber(5) FileName(file21.ss) Framework(2) Component(2.1)

VIOLATION Predicate: FileInclusion LOOP has been detected:

5 \rightarrow 8 \rightarrow 9 \rightarrow 10 \rightarrow 5

DETAILS:

NodeNumber(5) FileName(file21.ss) Framework(2) Component(2.1)
NodeNumber(8) FileName(file31.aop) Framework(3) Component(3.1)
NodeNumber(9) FileName(file12.aop) Framework(3) Component(3.2)
NodeNumber(5) FileName(file21.ss) Framework(2) Component(2.1)

VIOLATION Predicate: FileInclusion LOOP has been detected:

5 \rightarrow 8 \rightarrow 11 \rightarrow 10 \rightarrow 5

DETAILS:

NodeNumber(5) FileName(file21.ss) Framework(2) Component(2.1)
NodeNumber(8) FileName(file31.aop) Framework(3) Component(3.1)
NodeNumber(9) FileName(file12.aop) Framework(3) Component(3.2)
NodeNumber(5) FileName(file21.ss) Framework(2) Component(2.1)

VIOLATION Predicate: FileInclusion LOOP has been detected:

5 \rightarrow 8 \rightarrow 9 \rightarrow 10 \rightarrow 5 \rightarrow 7

DETAILS:

NodeNumber(7) FileName(file31.aop) Framework(3) Component(3.1)
NodeNumber(8) FileName(file31.aop) Framework(3) Component(3.1)
NodeNumber(9) FileName(file12.aop) Framework(3) Component(3.2)
NodeNumber(5) FileName(file21.ss) Framework(2) Component(2.1)
NodeNumber(7) FileName(file31.aop) Framework(3) Component(3.1)

VIOLATION Predicate: FileInclusion LOOP has been detected:

7 \rightarrow 8 \rightarrow 10 \rightarrow 5 \rightarrow 7

DETAILS:

NodeNumber(7) FileName(file31.aop) Framework(3) Component(3.1)
Appendix A3: A Prototype For CO-Toolkits' File Structures: Example#2 – An Artificial Example

NodeNumber(8) FileName(file31.cpp) Framework(3) Component(3.1)
NodeNumber(10) FileName(file32.cpp) Framework(3) Component(3.2)
NodeNumber(11) FileName(file21.cpp) Framework(2) Component(2.1)
NodeNumber(7) FileName(file31.cpp) Framework(3) Component(3.1)

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:
71-> 81-> 91-> 111-> 101-> 51-> 7
DETAILS:
NodeNumber(7) FileName(file31.cpp) Framework(3) Component(3.1)
NodeNumber(8) FileName(file31.cpp) Framework(3) Component(3.1)
NodeNumber(11) FileName(file12.cpp) Framework(3) Component(3.2)
NodeNumber(10) FileName(file12.cpp) Framework(3) Component(3.2)
NodeNumber(5) FileName(file21.cpp) Framework(2) Component(2.1)
NodeNumber(7) FileName(file31.cpp) Framework(3) Component(3.1)

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:
71-> 81-> 91-> 111-> 121-> 41-> 7
DETAILS:
NodeNumber(7) FileName(file31.cpp) Framework(3) Component(3.1)
NodeNumber(8) FileName(file31.cpp) Framework(3) Component(3.1)
NodeNumber(11) FileName(file12.cpp) Framework(3) Component(3.2)
NodeNumber(12) FileName(file12.cpp) Framework(3) Component(3.2)
NodeNumber(4) FileName(file21.cpp) Framework(2) Component(2.1)
NodeNumber(7) FileName(file31.cpp) Framework(3) Component(3.1)

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:
71-> 81-> 91-> 111-> 121-> 7
DETAILS:
NodeNumber(7) FileName(file31.cpp) Framework(3) Component(3.1)
NodeNumber(8) FileName(file31.cpp) Framework(3) Component(3.1)
NodeNumber(11) FileName(file12.cpp) Framework(3) Component(3.2)
NodeNumber(12) FileName(file12.cpp) Framework(3) Component(3.2)
NodeNumber(7) FileName(file31.cpp) Framework(3) Component(3.1)

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:
71-> 111-> 91-> 101-> 51-> 7
DETAILS:
NodeNumber(7) FileName(file31.cpp) Framework(3) Component(3.1)
NodeNumber(11) FileName(file12.cpp) Framework(3) Component(3.2)
NodeNumber(8) FileName(file31.cpp) Framework(3) Component(3.1)
NodeNumber(9) FileName(file31.cpp) Framework(3) Component(3.1)
NodeNumber(5) FileName(file21.cpp) Framework(2) Component(2.1)
NodeNumber(7) FileName(file31.cpp) Framework(3) Component(3.1)

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:
71-> 111-> 101-> 51-> 7
DETAILS:
NodeNumber(7) FileName(file31.cpp) Framework(3) Component(3.1)
NodeNumber(11) FileName(file12.cpp) Framework(3) Component(3.2)
NodeNumber(8) FileName(file31.cpp) Framework(3) Component(3.1)
NodeNumber(10) FileName(file12.cpp) Framework(3) Component(3.2)
NodeNumber(5) FileName(file21.cpp) Framework(2) Component(2.1)
NodeNumber(7) FileName(file31.cpp) Framework(3) Component(3.1)

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:
71-> 111-> 101-> 51-> 7
DETAILS:
NodeNumber(7) FileName(file31.cpp) Framework(3) Component(3.1)
NodeNumber(11) FileName(file12.cpp) Framework(3) Component(3.2)
NodeNumber(12) FileName(file12.cpp) Framework(3) Component(3.2)
NodeNumber(5) FileName(file21.cpp) Framework(2) Component(2.1)
NodeNumber(7) FileName(file31.cpp) Framework(3) Component(3.1)

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:
71-> 111-> 121-> 7
DETAILS:
NodeNumber(7) FileName(file31.cpp) Framework(3) Component(3.1)
NodeNumber(11) FileName(file12.cpp) Framework(3) Component(3.2)
NodeNumber(12) FileName(file12.cpp) Framework(3) Component(3.2)
NodeNumber(7) FileName(file31.cpp) Framework(3) Component(3.1)
Appendix A3: A Prototype For CO-Toolkits' File Structures: Example#2 – An Artificial Example

VIOLATION(Predicated_12): FileInclusionLOOP has been detected:
7 — 11 — 12 — 8 — 9 — 10 — 5 — 7
DETAILS:
NodeNumber(7) FileName(file31 top) Framework(3) Component(3,1)
NodeNumber(11) FileName(file12 login) Framework(3) Component(3,2)
NodeNumber(12) FileName(file12 login) Framework(3) Component(3,2)
NodeNumber(8) FileName(file31 login) Framework(3) Component(3,1)
NodeNumber(9) FileName(file31 login) Framework(3) Component(3,1)
NodeNumber(10) FileName(file31 login) Framework(3) Component(3,1)
NodeNumber(5) FileName(file21 login) Framework(2) Component(2,1)
NodeNumber(7) FileName(file31 login) Framework(3) Component(3,1)

VIOLATION(Predicated_13): FileInclusionLOOP has been detected:
7 — 11 — 12 — 8 — 9 — 10 — 5 — 7
DETAILS:
NodeNumber(7) FileName(file31 top) Framework(3) Component(3,1)
NodeNumber(11) FileName(file12 login) Framework(3) Component(3,2)
NodeNumber(12) FileName(file12 login) Framework(3) Component(3,2)
NodeNumber(8) FileName(file31 login) Framework(3) Component(3,1)
NodeNumber(9) FileName(file31 login) Framework(3) Component(3,1)
NodeNumber(10) FileName(file31 login) Framework(3) Component(3,1)
NodeNumber(5) FileName(file21 login) Framework(2) Component(2,1)
NodeNumber(7) FileName(file31 login) Framework(3) Component(3,1)

VIOLATION(Predicated_13): FileInclusionLOOP has been detected:
8 — 9 — 10 — 5 — 7 — 11 — 8
DETAILS:
NodeNumber(8) FileName(file31 login) Framework(3) Component(3,1)
NodeNumber(9) FileName(file31 login) Framework(3) Component(3,1)
NodeNumber(10) FileName(file31 login) Framework(3) Component(3,1)
NodeNumber(5) FileName(file21 login) Framework(2) Component(2,1)
NodeNumber(7) FileName(file31 login) Framework(3) Component(3,1)
NodeNumber(11) FileName(file12 login) Framework(3) Component(3,2)
NodeNumber(12) FileName(file12 login) Framework(3) Component(3,2)
NodeNumber(8) FileName(file31 login) Framework(3) Component(3,1)

VIOLATION(Predicated_13): FileInclusionLOOP has been detected:
8 — 9 — 10 — 5 — 7 — 11 — 8
DETAILS:
NodeNumber(8) FileName(file31 login) Framework(3) Component(3,1)
NodeNumber(9) FileName(file31 login) Framework(3) Component(3,1)
NodeNumber(10) FileName(file31 login) Framework(3) Component(3,1)
NodeNumber(5) FileName(file21 login) Framework(2) Component(2,1)
NodeNumber(7) FileName(file31 login) Framework(3) Component(3,1)

VIOLATION(Predicated_13): FileInclusionLOOP has been detected:
8 — 9 — 10 — 5 — 7 — 11 — 8
DETAILS:
NodeNumber(8) FileName(file31 login) Framework(3) Component(3,1)
NodeNumber(9) FileName(file31 login) Framework(3) Component(3,1)
NodeNumber(10) FileName(file31 login) Framework(3) Component(3,1)
NodeNumber(5) FileName(file21 login) Framework(2) Component(2,1)
NodeNumber(7) FileName(file31 login) Framework(3) Component(3,1)

VIOLATION(Predicated_13): FileInclusionLOOP has been detected:
8 — 9 — 10 — 5 — 7 — 11 — 8
DETAILS:
NodeNumber(8) FileName(file31 login) Framework(3) Component(3,1)
NodeNumber(9) FileName(file31 login) Framework(3) Component(3,1)
NodeNumber(10) FileName(file31 login) Framework(3) Component(3,1)
NodeNumber(5) FileName(file21 login) Framework(2) Component(2,1)
NodeNumber(7) FileName(file31 login) Framework(3) Component(3,1)
Appendix A3: A Prototype For CO-Tools' File Structures: Example#2 – An Artificial Example

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:

81 -> 101 -> 51 -> 71 -> 111 -> 8

DETAILS:

NodeNumber(8) FileName(file11.asm) Framework(3) Component(3.1)
NodeNumber(10) FileName(file12.top) Framework(3) Component(3.2)
NodeNumber(5) FileName(file21.asm) Framework(2) Component(2.1)
NodeNumber(7) FileName(file31.top) Framework(3) Component(3.1)
NodeNumber(11) FileName(file12.asm) Framework(3) Component(3.2)
NodeNumber(8) FileName(file31.asm) Framework(3) Component(3.1)

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:

81 -> 101 -> 51 -> 71 -> 111 -> 121 -> 8

DETAILS:

NodeNumber(8) FileName(file11.asm) Framework(3) Component(3.1)
NodeNumber(10) FileName(file12.top) Framework(3) Component(3.2)
NodeNumber(5) FileName(file21.asm) Framework(2) Component(2.1)
NodeNumber(7) FileName(file31.top) Framework(3) Component(3.1)
NodeNumber(11) FileName(file12.asm) Framework(3) Component(3.2)
NodeNumber(12) FileName(file12.asm) Framework(3) Component(3.2)
NodeNumber(8) FileName(file31.asm) Framework(3) Component(3.1)

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:

81 -> 101 -> 51 -> 8

DETAILS:

NodeNumber(8) FileName(file11.asm) Framework(3) Component(3.1)
NodeNumber(10) FileName(file12.top) Framework(3) Component(3.2)
NodeNumber(5) FileName(file21.asm) Framework(2) Component(2.1)
NodeNumber(8) FileName(file31.asm) Framework(3) Component(3.1)

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:

81 -> 101 -> 8

DETAILS:

NodeNumber(8) FileName(file11.asm) Framework(3) Component(3.1)
NodeNumber(10) FileName(file12.top) Framework(3) Component(3.2)
NodeNumber(8) FileName(file31.asm) Framework(3) Component(3.1)

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:

81 -> 111 -> 8

DETAILS:

NodeNumber(8) FileName(file11.asm) Framework(3) Component(3.1)
NodeNumber(11) FileName(file12.asm) Framework(3) Component(3.2)
NodeNumber(8) FileName(file31.asm) Framework(3) Component(3.1)

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:

81 -> 111 -> 101 -> 51 -> 71 -> 8

DETAILS:

NodeNumber(9) FileName(file11.asm) Framework(3) Component(3.1)
NodeNumber(11) FileName(file12.asm) Framework(3) Component(3.2)
NodeNumber(10) FileName(file12.top) Framework(3) Component(3.2)
NodeNumber(5) FileName(file21.asm) Framework(2) Component(2.1)
NodeNumber(7) FileName(file31.top) Framework(3) Component(3.1)
NodeNumber(8) FileName(file31.asm) Framework(2) Component(2.1)

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:

81 -> 111 -> 101 -> 51 -> 8

DETAILS:

NodeNumber(9) FileName(file11.asm) Framework(3) Component(3.1)
NodeNumber(11) FileName(file12.asm) Framework(3) Component(3.2)
NodeNumber(10) FileName(file12.top) Framework(3) Component(3.2)
NodeNumber(5) FileName(file21.asm) Framework(2) Component(2.1)
NodeNumber(8) FileName(file31.asm) Framework(3) Component(3.1)

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:

81 -> 111 -> 101 -> 8

DETAILS:

NodeNumber(9) FileName(file11.asm) Framework(3) Component(3.1)
NodeNumber(11) FileName(file12.asm) Framework(3) Component(3.2)
NodeNumber(10) FileName(file12.top) Framework(3) Component(3.2)
NodeNumber(8) FileName(file31.asm) Framework(3) Component(3.1)

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:

81 -> 111 -> 121 -> 41 -> 71 -> 8

DETAILS:

NodeNumber(8) FileName(file11.asm) Framework(3) Component(3.1)
NodeNumber(11) FileName(file12.asm) Framework(3) Component(3.2)
NodeNumber(12) FileName(file12.asm) Framework(3) Component(3.2)
NodeNumber(4) FileName(file21.top) Framework(2) Component(2.1)
NodeNumber(7) FileName(file31.top) Framework(3) Component(3.1)
NodeNumber(8) FileName(file31.asm) Framework(3) Component(3.1)
Appendix A3: A Prototype For CO-Toolkits’ File Structures: Example#2 – An Artificial Example

VIOLATION(Predicate_13): FileInclusion.LOOP has been detected:
101-> 51-> 71-> 81-> 91-> 10
DETAILS:
  NodeNumber(10) File Name(file12.top) Framework(3) Component(3.2)
  NodeNumber(10) File Name(file12.imm) Framework(2) Component(2.1)
  NodeNumber(10) File Name(file12.imm) Framework(3) Component(3.1)
  NodeNumber(10) File Name(file12.imm) Framework(3) Component(3.1)
  NodeNumber(10) File Name(file12.top) Framework(3) Component(3.2)

VIOLATION(Predicate_13): FileInclusion.LOOP has been detected:
101-> 51-> 71-> 81-> 11-> 10
DETAILS:
  NodeNumber(10) File Name(file12.top) Framework(3) Component(3.2)
  NodeNumber(10) File Name(file12.imm) Framework(2) Component(2.1)
  NodeNumber(10) File Name(file12.imm) Framework(3) Component(3.1)
  NodeNumber(10) File Name(file12.imm) Framework(3) Component(3.1)
  NodeNumber(10) File Name(file12.top) Framework(3) Component(3.2)

VIOLATION(Predicate_13): FileInclusion.LOOP has been detected:
101-> 51-> 71-> 11-> 81-> 91-> 10
DETAILS:
  NodeNumber(10) File Name(file12.top) Framework(3) Component(3.2)
  NodeNumber(10) File Name(file12.imm) Framework(2) Component(2.1)
  NodeNumber(10) File Name(file12.imm) Framework(3) Component(3.1)
  NodeNumber(10) File Name(file12.imm) Framework(3) Component(3.1)
  NodeNumber(10) File Name(file12.top) Framework(3) Component(3.2)

VIOLATION(Predicate_13): FileInclusion.LOOP has been detected:
101-> 51-> 71-> 11-> 81-> 10
DETAILS:
  NodeNumber(10) File Name(file12.top) Framework(3) Component(3.2)
  NodeNumber(10) File Name(file12.imm) Framework(2) Component(2.1)
  NodeNumber(10) File Name(file12.imm) Framework(3) Component(3.1)
  NodeNumber(10) File Name(file12.imm) Framework(3) Component(3.1)
  NodeNumber(10) File Name(file12.top) Framework(3) Component(3.2)

VIOLATION(Predicate_13): FileInclusion.LOOP has been detected:
101-> 51-> 71-> 11-> 81-> 10
DETAILS:
  NodeNumber(10) File Name(file12.top) Framework(3) Component(3.2)
  NodeNumber(10) File Name(file12.imm) Framework(2) Component(2.1)
  NodeNumber(10) File Name(file12.imm) Framework(3) Component(3.1)
  NodeNumber(10) File Name(file12.top) Framework(3) Component(3.2)

VIOLATION(Predicate_13): FileInclusion.LOOP has been detected:
101-> 51-> 71-> 11-> 121-> 81-> 91-> 10
DETAILS:
  NodeNumber(10) File Name(file12.top) Framework(3) Component(3.2)
  NodeNumber(10) File Name(file12.imm) Framework(2) Component(2.1)
  NodeNumber(10) File Name(file12.imm) Framework(3) Component(3.1)
  NodeNumber(10) File Name(file12.imm) Framework(3) Component(3.1)
  NodeNumber(10) File Name(file12.top) Framework(3) Component(3.2)

VIOLATION(Predicate_13): FileInclusion.LOOP has been detected:
101-> 51-> 71-> 11-> 121-> 81-> 10
DETAILS:
  NodeNumber(10) File Name(file12.top) Framework(3) Component(3.2)
  NodeNumber(10) File Name(file12.imm) Framework(2) Component(2.1)
  NodeNumber(10) File Name(file12.imm) Framework(3) Component(3.1)
  NodeNumber(10) File Name(file12.imm) Framework(3) Component(3.1)
  NodeNumber(10) File Name(file12.top) Framework(3) Component(3.2)

VIOLATION(Predicate_13): FileInclusion.LOOP has been detected:
101-> 51-> 81-> 91-> 10
DETAILS:
  NodeNumber(10) File Name(file12.top) Framework(3) Component(3.2)
  NodeNumber(10) File Name(file12.imm) Framework(2) Component(2.1)
  NodeNumber(10) File Name(file12.imm) Framework(3) Component(3.1)
  NodeNumber(10) File Name(file12.imm) Framework(3) Component(3.1)
  NodeNumber(10) File Name(file12.top) Framework(3) Component(3.2)
Appendix A3: A Prototype For CO-Toolkits' File Structures: Example#2 – An Artificial Example

NodeNumber(8) FileNumber(file31.ann) Framework(3) Component(3,1)
NodeNumber(10) FileNumber(file12.cpp) Framework(3) Component(3,2)
NodeNumber(5) FileNumber(file11.cpp) Framework(2) Component(2,1)
NodeNumber(7) FileNumber(file31.ann) Framework(3) Component(3,1)
NodeNumber(11) FileNumber(file12.cpp) Framework(3) Component(3,2)

VIOLATION(Predicate_13): FileInclusionLOOP has been detected:
11-> 12 l-> 8 l-> 11
DETAILS:
NodeNumber(11) FileNumber(file12.cpp) Framework(3) Component(3,2)
NodeNumber(12) FileNumber(file12.cpp) Framework(3) Component(3,2)
NodeNumber(8) FileNumber(file31.ann) Framework(3) Component(3,1)
NodeNumber(11) FileNumber(file12.cpp) Framework(3) Component(3,2)

VIOLATION(Predicate_13): FileInclusionLOOP has been detected:
12 l-> 4 l-> 7 l-> 8 l-> 11 l-> 12
DETAILS:
NodeNumber(12) FileNumber(file12.cpp) Framework(3) Component(3,2)
NodeNumber(4) FileNumber(file21.cpp) Framework(2) Component(2,1)
NodeNumber(7) FileNumber(file31.ann) Framework(3) Component(3,1)
NodeNumber(8) FileNumber(file11.cpp) Framework(3) Component(3,1)
NodeNumber(11) FileNumber(file12.cpp) Framework(3) Component(3,2)
NodeNumber(12) FileNumber(file12.cpp) Framework(3) Component(3,2)

VIOLATION(Predicate_13): FileInclusionLOOP has been detected:
12 l-> 7 l-> 8 l-> 11 l-> 12
DETAILS:
NodeNumber(12) FileNumber(file12.cpp) Framework(3) Component(3,2)
NodeNumber(7) FileNumber(file31.ann) Framework(3) Component(3,1)
NodeNumber(8) FileNumber(file11.cpp) Framework(3) Component(3,1)
NodeNumber(11) FileNumber(file12.cpp) Framework(3) Component(3,2)
NodeNumber(12) FileNumber(file12.cpp) Framework(3) Component(3,2)

VIOLATION(Predicate_13): FileInclusionLOOP has been detected:
12 l-> 7 l-> 8 l-> 11 l-> 12
DETAILS:
NodeNumber(12) FileNumber(file12.cpp) Framework(3) Component(3,2)
NodeNumber(7) FileNumber(file31.ann) Framework(3) Component(3,1)
NodeNumber(11) FileNumber(file12.cpp) Framework(3) Component(3,2)
NodeNumber(12) FileNumber(file12.cpp) Framework(3) Component(3,2)

VIOLATION(Predicate_13): FileInclusionLOOP has been detected:
12 l-> 8 l-> 9 l-> 10 l-> 5 l-> 7 l-> 11 l-> 12
DETAILS:
NodeNumber(12) FileNumber(file12.cpp) Framework(3) Component(3,2)
NodeNumber(8) FileNumber(file31.ann) Framework(3) Component(3,1)
NodeNumber(9) FileNumber(file31.ann) Framework(3) Component(3,1)
NodeNumber(10) FileNumber(file12.cpp) Framework(3) Component(3,2)
NodeNumber(5) FileNumber(file21.cpp) Framework(2) Component(2,1)
NodeNumber(7) FileNumber(file31.ann) Framework(3) Component(3,1)
NodeNumber(11) FileNumber(file12.cpp) Framework(3) Component(3,2)
NodeNumber(12) FileNumber(file12.cpp) Framework(3) Component(3,2)

VIOLATION(Predicate_13): FileInclusionLOOP has been detected:
12 l-> 8 l-> 10 l-> 5 l-> 7 l-> 11 l-> 12
DETAILS:
NodeNumber(12) FileNumber(file12.cpp) Framework(3) Component(3,2)
NodeNumber(8) FileNumber(file31.ann) Framework(3) Component(3,1)
NodeNumber(10) FileNumber(file12.cpp) Framework(3) Component(3,2)
NodeNumber(5) FileNumber(file21.cpp) Framework(2) Component(2,1)
NodeNumber(7) FileNumber(file31.ann) Framework(3) Component(3,1)
NodeNumber(11) FileNumber(file12.cpp) Framework(3) Component(3,2)
NodeNumber(12) FileNumber(file12.cpp) Framework(3) Component(3,2)

VIOLATION(Predicate_13): FileInclusionLOOP has been detected:
12 l-> 8 l-> 11 l-> 12
DETAILS:
NodeNumber(12) FileNumber(file12.cpp) Framework(3) Component(3,2)
NodeNumber(8) FileNumber(file31.ann) Framework(3) Component(3,1)
NodeNumber(10) FileNumber(file12.cpp) Framework(3) Component(3,2)
NodeNumber(5) FileNumber(file21.cpp) Framework(2) Component(2,1)
NodeNumber(11) FileNumber(file12.cpp) Framework(3) Component(3,2)
NodeNumber(12) FileNumber(file12.cpp) Framework(3) Component(3,2)

Checking for Predicate:4_12 is complete — STATUS: false

204
Checking for Predicate 4.13: GoodUseOf_top_files = predicates 4.18, 4.19, and 4.20
Checking for predicate 4.17: NoFileMayIncludeA_top_file
VIOLATION(Predicate 4.18): Top File file11.top from Framework(1) is included by file11.inm from Framework(1)
VIOLATION(Predicate 4.18): Top File file11.top from Framework(1) is included by file11.inm from Framework(1)
VIOLATION(Predicate 4.18): Top File file21.top from Framework(2) is included by file21.inm from Framework(3)
VIOLATION(Predicate 4.18): Top File file31.top from Framework(3) is included by file21.top from Framework(2)
VIOLATION(Predicate 4.18): Top File file31.top from Framework(3) is included by file21.top from Framework(2)
VIOLATION(Predicate 4.18): Top File file31.top from Framework(3) is included by file21.top from Framework(2)
VIOLATION(Predicate 4.18): Top File file31.top from Framework(3) is included by file21.top from Framework(2)
VIOLATION(Predicate 4.18): Top File file12.top from Framework(3) is included by file31.inm from Framework(3)
VIOLATION(Predicate 4.18): Top File file12.top from Framework(3) is included by file31.inm from Framework(3)
VIOLATION(Predicate 4.18): Top File file12.top from Framework(3) is included by file31.inm from Framework(3)
Checking for predicate 4.17 is complete — STATUS: false

Checking for predicate 4.18: FromSame_Fr_A_top_fileMayIncludeOnlyIts_intermediate_file
VIOLATION(Predicate 4.18): Top File file11.top from Framework(1) includes its Internal File
VIOLATION(Predicate 4.18): Top File file21.top from Framework(2) includes File file31.top from Framework(3)
VIOLATION(Predicate 4.18): Top File file21.top from Framework(2) includes File file11.inm from Framework(1)
VIOLATION(Predicate 4.18): Top File file12.top from Framework(3) includes File file21.inm from Framework(2)
Checking for predicate 4.18 is complete — STATUS: false

Checking for predicate 4.19: A_top_fileMayInclude_shared_filesFromPermittedSharedPools
VIOLATION(Predicate 4.19): Top File file31.top from Framework(3) includes a shared file from a non-accessible SharedPool(1)
Checking for predicate 4.19 is complete — STATUS: false

Checking for Predicate 4.13 is complete — STATUS: false

Checking for Predicate 4.14: GoodUseOf_intermediate_files = predicates 4.21 and 4.22
Checking for predicate 4.20: An_intermediate_fileMayIncludeOnlyIts_intermediate_fileAndMayIncludeOther_intermediate_filesFromSame_Fr
VIOLATION(Predicate 4.20): Intermediate File file11.inm from Framework(1) includes its top file file11.top
VIOLATION(Predicate 4.20): Intermediate File file21.inm from Framework(2) includes file31.inm from Framework(3)
VIOLATION(Predicate 4.20): Intermediate File file21.inm from Framework(2) includes file31.top from Framework(3)
ERROR(Predicate 4.20): Intermediate File file21.inm includes FileNode(30) that is not found in any Framework or SharedPool
VIOLATION(Predicate 4.20): Intermediate File file12.inm from Framework(3) includes its top file file12.top
Checking for predicate 4.20 is complete — STATUS: false

Checking for Predicate 4.14 is complete — STATUS: false

Checking for Predicate 4.15: GoodUseOf_intermediate_files = predicates 4.23 and 4.24
Appendix A3: A Prototype For CO-Toolkits’ File Structures: Example#2 – An Artificial Example

Checking for predicate 4.21: An _internal_fileMayNotIncludeAnyFile

VIOLATION(Predicate 4.21): InternalFile file11.lst includes FileNode 1
VIOLATION(Predicate 4.21): InternalFile file21.lst includes FileNode 8
VIOLATION(Predicate 4.21): InternalFile file31.lst includes FileNode 10
VIOLATION(Predicate 4.21): InternalFile file12.lst includes FileNode 7
VIOLATION(Predicate 4.21): InternalFile file12.lst includes FileNode 4
VIOLATION(Predicate 4.21): InternalFile file12.lst includes FileNode 8

Checking for predicate 4.21 is complete — STATUS: false

Checking for predicate 4.22: NoFileMayIncludeAn _internal_fileExcept _ intermediate_file

VIOLATION(Predicate 4.22): Internal File file11.lst is included by file11.top from Framework(1) — Component(1,1)
VIOLATION(Predicate 4.22): Internal File file31.lst is included by file12.top from Framework(3) — Component(3,2)

Checking for predicate 4.22 is complete — STATUS: false

Checking for Predicate 4.15 is complete — STATUS: false

Checking for Predicate 4.11 is complete — STATUS: false

Checking for Predicate 4.12: InclusionLoopsMayNotExist

Checking for Predicate 4.12 is complete — STATUS: false

Checking for Predicate 4.8 is complete — STATUS: false

Checking for GoodFileStructure_predicate4.5 is complete — STATUS: false

206
Appendix A4

A Prototype For CO-Toolkits' File Structures: Example#3 – ACE Sample File Structure

A4.1 Overview

This Appendix presents the third example that has been used to demonstrate our prototype for CO-Toolkits' file structures. This example has been based on a sample file structure from ACE. Section A4.3 presents the rules violations reported by this prototype for this example. For convenience, we have replaced the .cpp, .h, and .i filename extensions used in ACE by .top, .inm, and .incl.

A4.2 Example#3: ACE Sample File Structure

```
FILE STRUCTURE: As Extracted From The Current Example
Framework(1)
Component(1,1)
  FileNode(1): IO_Cmd_Msg.top includes FileNodes:
  FileNode(2): IO_Cmd_Msg.inm includes FileNodes:
  FileNode(3): IO_Cmd_Msg.inl includes FileNodes:
Component(1,2)
  FileNode(4): Map_Manager.top includes FileNodes: 56 110 155 5 6
  FileNode(5): Map_Manager.inm includes FileNodes: 116 6 4
  FileNode(6): Map_Manager.inl includes FileNodes:
Component(1,3)
  FileNode(7): Message_Block.top includes FileNodes: 8 9
  FileNode(8): Message_Block.inm includes FileNodes: 116 110 9
  FileNode(9): Message_Block.inl includes FileNodes:
Component(1,4)
  FileNode(10): Message_Queue.top includes FileNodes: 11 12
  FileNode(11): Message_Queue.inm includes FileNodes: 8 2 83 12 10
  FileNode(12): Message_Queue.inl includes FileNodes:
Component(1,5)
  FileNode(13): Module.top includes FileNodes: 14 23 15
  FileNode(14): Module.inm includes FileNodes: 116 26 15 13
  FileNode(15): Module.inl includes FileNodes:
Component(1,6)
  FileNode(16): Multiplexor.top includes FileNodes: 17 18
```
Appendix A4: A Prototype For CO-Toolkits' File Structures: Example#3 – ACE Sample File Structure

Component(1,7)
- FileNode(17): Multiplexer.iml Includes FileNodes: 14 5 18
- FileNode(18): Multiplexer.inl Includes FileNodes:

Component(1,8)
- FileNode(19): Stream.top Includes FileNodes: 14 20 23 21
- FileNode(20): Stream.iml Includes FileNodes: 17 116 2 8 143 14 21 19

Component(1,9)
- FileNode(21): Stream.top Includes FileNodes: 17 18
- FileNode(22): Stream.inl Includes FileNodes: 26 24 22

Component(1,9)
- FileNode(23): Task.top Includes FileNodes: 26 14 155 27
- FileNode(24): Task.iml Includes FileNodes: 159 68 27

Framework(2)
Component(2,1)
- FileNode(25): CORBA_Handler.top Includes FileNodes: 29 30
- FileNode(26): CORBA_Handler.inl Includes FileNodes: 155 157 158 30

Component(2,2)
- FileNode(27): CORBA_Ref.top Includes FileNodes: 32 33
- FileNode(28): CORBA_Ref.inl Includes FileNodes: 33 31

Framework(3)
Component(3,1)
- FileNode(29): Set.top Includes FileNodes: 35 36
- FileNode(30): Set.iml Includes FileNodes: 116 36 34

Component(3,2)
- FileNode(31): Stack.top Includes FileNodes: 38 39
- FileNode(32): Stack.iml Includes FileNodes: 116 39 37

Component(3,3)
- FileNode(33): SString.top Includes FileNodes: 110 155 41 42
- FileNode(34): SString.iml Includes FileNodes: 116 42

Framework(4)
Component(4,1)
- FileNode(35): Activation_Queue.top Includes FileNodes: 44
- FileNode(36): Activation_Queue.inl Includes FileNodes: 56 11 50

Component(4,2)
- FileNode(37): Future.top Includes FileNodes: 47
- FileNode(38): Future.iml Includes FileNodes: 56 46

Component(4,3)
- FileNode(39): Method_Object.top Includes FileNodes: 50
- FileNode(40): Method_Object.inl Includes FileNodes: 119

Component(4,4)
- FileNode(41): Process_Manager.top Includes FileNodes: 53 54
- FileNode(42): Process_Manager.inl Includes FileNodes: 56 54

Component(4,5)
- FileNode(43): Synch.top Includes FileNodes: 65 56 143 57
- FileNode(44): Synch.iml Includes FileNodes: 116 149 57 62

Component(4,6)
- FileNode(45): Synch_Options.top Includes FileNodes: 59 116
- FileNode(46): Synch_Options.inl Includes FileNodes:

Component(4,7)
- FileNode(47): Synch_Top Includes FileNodes: 65 143 62 63
- FileNode(48): Synch_Tim Includes FileNodes: 116 56 63 61

Component(4,8)
- FileNode(49): Synch_Tim Includes FileNodes: 65

Component(4,9)
- FileNode(50): Thread_Manager.top Includes FileNodes: 68 69
- FileNode(51): Thread_Manager.inl Includes FileNodes: 65 56 69

Component(4,10)
- FileNode(52): Token.top Includes FileNodes: 65 143 71 72

208
A4.3 Example#3: ACE Sample Output Report

Checking for GoodFileStructure Predicate 4.5: GoodFileStructure = predicates 4.6, 4.7, and 4.8
Checking for Predicate 4.6: OnlyFourFileCategories
Checking for Predicate 4.6 is complete — STATUS: true

Checking for Predicate 4.7: GoodDistinctionOfFileCategories: Predicates 4.9, and 4.10
Checking for Predicate 4.9: FileSetsOfFileCategoriesShouldBeDistinct
Checking for Predicate 4.9 is complete — STATUS: true

Checking for Predicate 4.10: ComponentMainFilesShouldBeDistinct
Checking for Predicate 4.10 is complete — STATUS: true

Checking for Predicate 4.7 is complete — STATUS: true
Checking for Predicate 4.8: GoodFileInclusionRelation = predicates 4.11, 4.12, and 4.13
Checking for Predicate 4.11: GoodUseOfFiles = predicates 4.14, 4.15, 4.16, and 4.17
Checking for Predicate 4.12: inclusion loops MayNotExist
FILE STRUCTURE: As Extracted From The Current Example

VIOLATION(Predicate 4.13): FileInclusionLOOP has been detected:
4 $\rightarrow$ 5 $\rightarrow$ 4
DETAILS:
NodeNumber(4) FileName(Map_Manager.top) Framework(1) Component(1),2
NodeNumber(5) FileName(Map_Manager.imm) Framework(1) Component(1),2
NodeNumber(4) FileName(Map_Manager.top) Framework(1) Component(1),2

VIOLATION(Predicate 4.13): FileInclusionLOOP has been detected:
5 $\rightarrow$ 4 $\rightarrow$ 5
DETAILS:
NodeNumber(5) FileName(Map_Manager.imm) Framework(1) Component(1),2
NodeNumber(4) FileName(Map_Manager.top) Framework(1) Component(1),2
NodeNumber(5) FileName(Map_Manager.imm) Framework(1) Component(1),2

VIOLATION(Predicate 4.13): FileInclusionLOOP has been detected:
10 $\rightarrow$ 11 $\rightarrow$ 10
DETAILS:
NodeNumber(10) FileName(Message_Queue.top) Framework(1) Component(1),4
NodeNumber(11) FileName(Message_Queue.imm) Framework(1) Component(1),4
NodeNumber(10) FileName(Message_Queue.top) Framework(1) Component(1),4

VIOLATION(Predicate 4.13): FileInclusionLOOP has been detected:
11 $\rightarrow$ 101 $\rightarrow$ 11
DETAILS:
NodeNumber(11) FileName(Message_Queue.imm) Framework(1) Component(1),4
NodeNumber(10) FileName(Message_Queue.top) Framework(1) Component(1),4
NodeNumber(11) FileName(Message_Queue.imm) Framework(1) Component(1),4

VIOLATION(Predicate 4.13): FileInclusionLOOP has been detected:
13 $\rightarrow$ 14 $\rightarrow$ 13
DETAILS:
NodeNumber(13) FileName(Module.top) Framework(1) Component(1),5
NodeNumber(14) FileName(Module.imm) Framework(1) Component(1),5
NodeNumber(13) FileName(Module.top) Framework(1) Component(1),5

VIOLATION(Predicate 4.13): FileInclusionLOOP has been detected:
13 $\rightarrow$ 23 $\rightarrow$ 22 $\rightarrow$ 17 $\rightarrow$ 14 $\rightarrow$ 13
DETAILS:
NodeNumber(13) FileName(Module.top) Framework(1) Component(1),5
NodeNumber(23) FileName(Stream_Modules.imm) Framework(1) Component(1),8
NodeNumber(22) FileName(Stream_Modules.top) Framework(1) Component(1),8
NodeNumber(17) FileName(Multiplexor.imm) Framework(1) Component(1),6
NodeNumber(13) FileName(Module.top) Framework(1) Component(1),5

VIOLATION(Predicate 4.13): FileInclusionLOOP has been detected:
14 $\rightarrow$ 13 $\rightarrow$ 14
DETAILS:
NodeNumber(14) FileName(Module.imm) Framework(1) Component(1),5
NodeNumber(13) FileName(Module.top) Framework(1) Component(1),5
NodeNumber(14) FileName(Module.imm) Framework(1) Component(1),5

VIOLATION(Predicate 4.13): FileInclusionLOOP has been detected:
14 $\rightarrow$ 13 $\rightarrow$ 23 $\rightarrow$ 22 $\rightarrow$ 17 $\rightarrow$ 14
DETAILS:
NodeNumber(14) FileName(Module.imm) Framework(1) Component(1),5
NodeNumber(13) FileName(Module.top) Framework(1) Component(1),5
NodeNumber(23) FileName(Stream_Modules.imm) Framework(1) Component(1),8
NodeNumber(22) FileName(Stream_Modules.top) Framework(1) Component(1),8
NodeNumber(17) FileName(Multiplexor.imm) Framework(1) Component(1),6
NodeNumber(14) FileName(Module.imm) Framework(1) Component(1),5

VIOLATION(Predicate 4.13): FileInclusionLOOP has been detected:
17 $\rightarrow$ 14 $\rightarrow$ 13 $\rightarrow$ 23 $\rightarrow$ 22 $\rightarrow$ 17
DETAILS:
NodeNumber(17) FileName(Multiplexor.imm) Framework(1) Component(1),6
NodeNumber(14) FileName(Module.imm) Framework(1) Component(1),5
NodeNumber(13) FileName(Module.top) Framework(1) Component(1),5
NodeNumber(23) FileName(Stream_Modules.imm) Framework(1) Component(1),8
NodeNumber(22) FileName(Stream_Modules.top) Framework(1) Component(1),8
NodeNumber(17) FileName(Multiplexor.imm) Framework(1) Component(1),6

VIOLATION(Predicate 4.13): FileInclusionLOOP has been detected:
19 $\rightarrow$ 20 $\rightarrow$ 19
DETAILS:
NodeNumber(19) FileName(Stream.top) Framework(1) Component(1),7
NodeNumber(20) FileName(Stream.imm) Framework(1) Component(1),7
NodeNumber(19) FileName(Stream.top) Framework(1) Component(1),7
Appendix A4: A Prototype For CO-Toolkits' File Structures: Example#3 – ACE Sample File Structure

VIOLATION(Predicate4_13): FileInclusionLOOP has been detected:
20 → 19 → 20
DETAILS:
- NodeNumber(20) FileName(Stream.inm) Framework(1) Component(1.7)
- NodeNumber(19) FileName(Stream.top) Framework(1) Component(1.7)
- NodeNumber(20) FileName(Stream.inm) Framework(1) Component(1.7)

VIOLATION(Predicate4_13): FileInclusionLOOP has been detected:
22 → 17 → 14 → 13 → 23 → 22
DETAILS:
- NodeNumber(22) FileName(Stream_Modules.top) Framework(1) Component(1.8)
- NodeNumber(17) FileName(Multiplexor.inm) Framework(1) Component(1.6)
- NodeNumber(14) FileName(Module.inm) Framework(1) Component(1.5)
- NodeNumber(13) FileName(Module.top) Framework(1) Component(1.5)
- NodeNumber(23) FileName(Stream_Modules.inm) Framework(1) Component(1.8)
- NodeNumber(22) FileName(Stream_Modules.top) Framework(1) Component(1.8)

VIOLATION(Predicate4_13): FileInclusionLOOP has been detected:
23 → 22 → 17 → 14 → 13 → 23
DETAILS:
- NodeNumber(22) FileName(Stream_Modules.inm) Framework(1) Component(1.8)
- NodeNumber(22) FileName(Stream_Modules.top) Framework(1) Component(1.8)
- NodeNumber(17) FileName(Multiplexor.inm) Framework(1) Component(1.6)
- NodeNumber(14) FileName(Module.inm) Framework(1) Component(1.5)
- NodeNumber(13) FileName(Module.top) Framework(1) Component(1.5)
- NodeNumber(23) FileName(Stream_Modules.inm) Framework(1) Component(1.8)

VIOLATION(Predicate4_13): FileInclusionLOOP has been detected:
31 → 32 → 31
DETAILS:
- NodeNumber(31) FileName(CORBA_Ref.top) Framework(2) Component(2.2)
- NodeNumber(32) FileName(CORBA_Ref.inm) Framework(2) Component(2.2)
- NodeNumber(31) FileName(CORBA_Ref.top) Framework(2) Component(2.2)

VIOLATION(Predicate4_13): FileInclusionLOOP has been detected:
32 → 31 → 32
DETAILS:
- NodeNumber(32) FileName(CORBA_Ref.inm) Framework(2) Component(2.2)
- NodeNumber(31) FileName(CORBA_Ref.top) Framework(2) Component(2.2)
- NodeNumber(32) FileName(CORBA_Ref.inm) Framework(2) Component(2.2)

VIOLATION(Predicate4_13): FileInclusionLOOP has been detected:
34 → 35 → 34
DETAILS:
- NodeNumber(34) FileName(Set.top) Framework(3) Component(3.1)
- NodeNumber(35) FileName(Set.inm) Framework(3) Component(3.1)
- NodeNumber(34) FileName(Set.top) Framework(3) Component(3.1)

VIOLATION(Predicate4_13): FileInclusionLOOP has been detected:
35 → 34 → 35
DETAILS:
- NodeNumber(35) FileName(Set.inm) Framework(3) Component(3.1)
- NodeNumber(34) FileName(Set.top) Framework(3) Component(3.1)
- NodeNumber(35) FileName(Set.inm) Framework(3) Component(3.1)

VIOLATION(Predicate4_13): FileInclusionLOOP has been detected:
37 → 38 → 37
DETAILS:
- NodeNumber(37) FileName(Stack.top) Framework(3) Component(3.2)
- NodeNumber(38) FileName(Stack.inm) Framework(3) Component(3.2)
- NodeNumber(37) FileName(Stack.top) Framework(3) Component(3.2)

VIOLATION(Predicate4_13): FileInclusionLOOP has been detected:
38 → 37 → 38
DETAILS:
- NodeNumber(38) FileName(Stack.inm) Framework(3) Component(3.2)
- NodeNumber(37) FileName(Stack.top) Framework(3) Component(3.2)
- NodeNumber(38) FileName(Stack.inm) Framework(3) Component(3.2)

VIOLATION(Predicate4_13): FileInclusionLOOP has been detected:
46 → 47 → 46
DETAILS:
- NodeNumber(46) FileName(Future.top) Framework(4) Component(4.2)
- NodeNumber(47) FileName(Future.inm) Framework(4) Component(4.2)
- NodeNumber(46) FileName(Future.top) Framework(4) Component(4.2)

VIOLATION(Predicate4_13): FileInclusionLOOP has been detected:
47 → 46 → 47
DETAILS:
- NodeNumber(47) FileName(Future.inm) Framework(4) Component(4.2)
- NodeNumber(46) FileName(Future.top) Framework(4) Component(4.2)
- NodeNumber(47) FileName(Future.inm) Framework(4) Component(4.2)

VIOLATION(Predicate4_13): FileInclusionLOOP has been detected:
56 → 62 → 56
DETAILS:
NodeNumber(56)  FileName(Synch.iam)  Framework(4)  Component(4.5)
NodeNumber(62)  FileName(Synch_Timp)  Framework(4)  Component(4.7)
NodeNumber(56)  FileName(Synch.iam)  Framework(4)  Component(4.5)

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:
56 l-> 62 l-> 61 l-> 143 l-> 56
DETAILS:
NodeNumber(56)  FileName(Synch.iam)  Framework(4)  Component(4.5)
NodeNumber(52)  FileName(Synch_Timp)  Framework(4)  Component(4.7)
NodeNumber(61)  FileName(Synch_Timp)  Framework(4)  Component(4.7)
NodeNumber(143)  FileName(Time_Value.iam)  Framework(11)  Component(11.8)
NodeNumber(56)  FileName(Synch.iam)  Framework(4)  Component(4.5)

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:
61 l-> 62 l-> 61
DETAILS:
NodeNumber(61)  FileName(Synch_Timp)  Framework(4)  Component(4.7)
NodeNumber(62)  FileName(Synch_Timp)  Framework(4)  Component(4.7)
NodeNumber(61)  FileName(Synch_Timp)  Framework(4)  Component(4.7)

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:
61 l-> 143 l-> 56 l-> 62 l-> 61
DETAILS:
NodeNumber(61)  FileName(Synch_Timp)  Framework(4)  Component(4.7)
NodeNumber(143)  FileName(Time_Value.iam)  Framework(11)  Component(11.8)
NodeNumber(56)  FileName(Synch.iam)  Framework(4)  Component(4.5)
NodeNumber(62)  FileName(Synch_Timp)  Framework(4)  Component(4.7)
NodeNumber(61)  FileName(Synch_Timp)  Framework(4)  Component(4.7)

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:
62 l-> 56 l-> 62
DETAILS:
NodeNumber(62)  FileName(Synch_Timp)  Framework(4)  Component(4.7)
NodeNumber(56)  FileName(Synch.iam)  Framework(4)  Component(4.5)
NodeNumber(62)  FileName(Synch_Timp)  Framework(4)  Component(4.7)

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:
62 l-> 61 l-> 62
DETAILS:
NodeNumber(62)  FileName(Synch_Timp)  Framework(4)  Component(4.7)
NodeNumber(61)  FileName(Synch_Timp)  Framework(4)  Component(4.7)
NodeNumber(62)  FileName(Synch_Timp)  Framework(4)  Component(4.7)

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:
62 l-> 61 l-> 143 l-> 56 l-> 62
DETAILS:
NodeNumber(62)  FileName(Synch_Timp)  Framework(4)  Component(4.7)
NodeNumber(61)  FileName(Synch_Timp)  Framework(4)  Component(4.7)
NodeNumber(143)  FileName(Time_Value.iam)  Framework(11)  Component(11.8)
NodeNumber(56)  FileName(Synch.iam)  Framework(4)  Component(4.5)
NodeNumber(62)  FileName(Synch_Timp)  Framework(4)  Component(4.7)

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:
73 l-> 74 l-> 73
DETAILS:
NodeNumber(73)  FileName(Acceptor.top)  Framework(5)  Component(5.1)
NodeNumber(74)  FileName(Acceptor.iam)  Framework(5)  Component(5.1)
NodeNumber(73)  FileName(Acceptor.top)  Framework(5)  Component(5.1)

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:
74 l-> 73 l-> 74
DETAILS:
NodeNumber(74)  FileName(Acceptor.iam)  Framework(5)  Component(5.1)
NodeNumber(73)  FileName(Acceptor.top)  Framework(5)  Component(5.1)
NodeNumber(74)  FileName(Acceptor.iam)  Framework(5)  Component(5.1)

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:
76 l-> 77 l-> 76
DETAILS:
NodeNumber(76)  FileName(Connector.top)  Framework(5)  Component(5.2)
NodeNumber(77)  FileName(Connector.iam)  Framework(5)  Component(5.2)
NodeNumber(76)  FileName(Connector.top)  Framework(5)  Component(5.2)

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:
77 l-> 76 l-> 77
DETAILS:
NodeNumber(77)  FileName(Connector.iam)  Framework(5)  Component(5.2)
NodeNumber(76)  FileName(Connector.top)  Framework(5)  Component(5.2)
NodeNumber(77)  FileName(Connector.iam)  Framework(5)  Component(5.2)

VIOLATION(Predicate_13): FileInclusion_LOOP has been detected:
79 l-> 80 l-> 79
DETAILS:
NodeNumber(79)  FileName(Dynamic_Service.top)  Framework(5)  Component(5.3)
NodeNumber(80)  FileName(Dynamic_Service.iam)  Framework(5)  Component(5.3)
NodeNumber(79)  FileName(Dynamic_Service.top)  Framework(5)  Component(5.3)
Appendix A4: A Prototype For CO-Toolkits' File Structures: Example3 – ACE Sample File Structure

VIOLATION(Predicate_4, 13): FileInclusion.Loop has been detected:
80 l-> 79 l-> 80
DETAILS:
NodeNumber(80) FileName(Dynamic_Service.inm) Framework(5) Component(5,3)
NodeNumber(79) FileName(Dynamic_Service.top) Framework(5) Component(5,3)
NodeNumber(80) FileName(Dynamic_Service.inm) Framework(5) Component(5,3)

VIOLATION(Predicate_4, 13): FileInclusion.Loop has been detected:
85 l-> 86 l-> 85
DETAILS:
NodeNumber(85) FileName(Svc_Handler.top) Framework(3) Component(5,5)
NodeNumber(86) FileName(Svc_Handler.inm) Framework(5) Component(5,5)
NodeNumber(85) FileName(Svc_Handler.top) Framework(5) Component(5,5)

VIOLATION(Predicate_4, 13): FileInclusion.Loop has been detected:
86 l-> 85 l-> 86
DETAILS:
NodeNumber(86) FileName(Svc_Handler.inm) Framework(5) Component(5,5)
NodeNumber(85) FileName(Svc_Handler.top) Framework(5) Component(5,5)
NodeNumber(86) FileName(Svc_Handler.inm) Framework(5) Component(5,5)

VIOLATION(Predicate_4, 13): FileInclusion.Loop has been detected:
122 l-> 123 l-> 122
DETAILS:
NodeNumber(122) FileName(Event_Handler.inm) Framework(11) Component(11,1)
NodeNumber(123) FileName(Event_Handler.top) Framework(11) Component(11,1)
NodeNumber(122) FileName(Event_Handler.inm) Framework(11) Component(11,1)

VIOLATION(Predicate_4, 13): FileInclusion.Loop has been detected:
123 l-> 122 l-> 123
DETAILS:
NodeNumber(123) FileName(Event_Handler.inm) Framework(11) Component(11,1)
NodeNumber(122) FileName(Event_Handler.top) Framework(11) Component(11,1)
NodeNumber(123) FileName(Event_Handler.inm) Framework(11) Component(11,1)

VIOLATION(Predicate_4, 13): FileInclusion.Loop has been detected:
143 l-> 56 l-> 62 l-> 61 l-> 143
DETAILS:
NodeNumber(143) FileName(Time_Value.inm) Framework(11) Component(11,8)
NodeNumber(56) FileName(Synch.inm) Framework(4) Component(4,3)
NodeNumber(62) FileName(Synch_T.inm) Framework(4) Component(4,7)
NodeNumber(61) FileName(Synch_T.top) Framework(4) Component(4,7)
NodeNumber(143) FileName(Time_Value.inm) Framework(11) Component(11,8)

Checking for Predicate 4, 12 is complete — STATUS: false

Checking for Predicate 4, 13: GoodUseOf_top_Files = predicates 4, 18, 4, 19, and 4, 20
Checking for predicate 4, 17: NoFileMayIncludeA_top_File
VIOLATION(Predicate_4, 18): Top File Map_Manager.top from Framework(1) is included by Map_Manager.inm from Framework(1)
VIOLATION(Predicate_4, 18): Top File Message_Queue.top from Framework(1) is included by Message_Queue.inm from Framework(1)
VIOLATION(Predicate_4, 18): Top File Module.top from Framework(1) is included by Module.inm from Framework(1)
VIOLATION(Predicate_4, 18): Top File Stream.top from Framework(1) is included by Stream.inm from Framework(1)
VIOLATION(Predicate_4, 18): Top File stream_Modules.top from Framework(1) is included by Stream_Modules.inm from Framework(1)
VIOLATION(Predicate_4, 18): Top File CORBA_Ref.top from Framework(2) is included by CORBA_Ref.inm from Framework(2)
VIOLATION(Predicate_4, 18): Top File Set.top from Framework(3) is included by Set.inm from Framework(3)
VIOLATION(Predicate_4, 18): Top File Stack.top from Framework(3) is included by Stack.inm from Framework(3)
VIOLATION(Predicate_4, 18): Top File Future.top from Framework(4) is included by Future.inm from Framework(4)
VIOLATION(Predicate_4, 18): Top File Synch_T.top from Framework(4) is included by Synch_T.inm from Framework(4)
VIOLATION(Predicate_4, 18): Top File Acceptor.top from Framework(5) is included by Acceptor.inm from Framework(5)
VIOLATION(Predicate_4, 18): Top File Connector.top from Framework(5) is included by Connector.inm from Framework(5)
VIOLATION(Predicate_4, 18): Top File Dynamic_Service.top from Framework(5) is included by Dynamic_Service.inm from Framework(5)
VIOLATION(Predicate_4, 18): Top File Svc_Handler.top from Framework(5) is included by Svc_Handler.inm from Framework(5)
Checking for predicate 4, 17 is complete — STATUS: false

Checking for predicate 4, 18: FromSame_Fr_A_top_FileMayIncludeOnlyIts_intermediate_File
VIOLATION(Predicate_4, 18): Top File Map_Manager.top from Framework(1) includes File Synch.inm from Framework(4)
VIOLATION(Predicate_4, 18): Top File Map_Manager.top from Framework(1) includes File MailBox.inm from Framework(8)
VIOLATION(Predicate_4, 18): Top File Message_Block.top from Framework(1) includes its Internal File
VIOLATION(Predicate_4, 18): Top File Message_Queue.top from Framework(1) includes its Internal File
VIOLATION(Predicate_4, 18): Top File Module.top from Framework(1) includes its Internal File
VIOLATION(Predicate_4, 18): Top File Multiplexor.top from Framework(1) includes its Internal File
VIOLATION(Predicate_4, 18): Top File Stream.top from Framework(1) includes its Internal File
VIOLATION(Predicate_4, 18): Top File Task.top from Framework(1) includes its Internal File
VIOLATION(Predicate_4, 18): Top File CORBA_Handler.top from Framework(2) includes its Internal File
VIOLATION(Predicate_4, 18): Top File CORBA_Ref.top from Framework(2) includes its Internal File
VIOLATION(Predicate_4, 18): Top File Set.top from Framework(3) includes its Internal File
VIOLATION(Predicate_4, 18): Top File Stack.top from Framework(3) includes its Internal File
VIOLATION(Predicate_4, 18): Top File Svc_Handler.top from Framework(3) includes File MailBox.inm from Framework(8)

214
Appendix A4: A Prototype For CO-Toolkits’ File Structures: Example #3 – ACE Sample File Structure
Appendix A4: A Prototype For CO-Toolkits' File Structures: Example#3 – ACE Sample File Structure

VIOLATION(Predicate 4_22): Internal File Thread_Manager.ist is included by Thread_Manager.top from Framework(4) — Component(4.9)
VIOLATION(Predicate 4_22): Internal File Token.ist is included by Token.top from Framework(4) — Component(4.10)
VIOLATION(Predicate 4_22): Internal File Svc_Handler.ist is included by Svc_Handler.top from Framework(5) — Component(5.5)
VIOLATION(Predicate 4_22): Internal File ACE.ist is included by Log_Msg.top from Framework(7) — Component(7.1)
VIOLATION(Predicate 4_22): Internal File ACE.ist is included by Time_Value.iam from Framework(11) — Component(11.8)
VIOLATION(Predicate 4_22): Internal File Event_Handler.ist is included by Event_Handler.top from Framework(11) — Component(11.1)
VIOLATION(Predicate 4_22): Internal File Event_Handler.ist is included by Event_Handler.ist from Framework(11) — Component(11.2)
VIOLATION(Predicate 4_22): Internal File Handle_Set.ist is included by Handle_Set.top from Framework(11) — Component(11.3)
VIOLATION(Predicate 4_22): Internal File Reactor_Ext.ist is included by Reactor_Ext.top from Framework(11) — Component(11.4)
VIOLATION(Predicate 4_22): Internal File Signal.ist is included by Signal.top from Framework(11) — Component(11.7)
VIOLATION(Predicate 4_22): Internal File Time_Value.ist is included by Time_Value.top from Framework(11) — Component(11.8)
VIOLATION(Predicate 4_22): Internal File Timer_Queue.ist is included by Timer_Queue.top from Framework(11) — Component(11.9)
Checking for predicate 4_22 is complete — STATUS: false
Checking for Predicate 4_15 is complete — STATUS: false

Checking for Predicate 4_12: InclusionLoopsMayNotExist
VIOLATION(Predicate 4_13): FileInclusionLOOP has been detected:
  4 → 5 → 4
DETAILS:
  NodeNumber(4) FileNumber(Map_Manager.top) Framework(1) Component(1.2)
  NodeNumber(5) FileNumber(Map_Manager.ist) Framework(1) Component(1.2)
  NodeNumber(6) FileNumber(Map_Manager.top) Framework(1) Component(1.2)
VIOLATION(Predicate 4_13): FileInclusionLOOP has been detected:
  5 → 4 → 5
DETAILS:
  NodeNumber(5) FileNumber(Map_Manager.ist) Framework(1) Component(1.2)
  NodeNumber(4) FileNumber(Map_Manager.top) Framework(1) Component(1.2)
  NodeNumber(5) FileNumber(Map_Manager.ist) Framework(1) Component(1.2)
VIOLATION(Predicate 4_13): FileInclusionLOOP has been detected:
  10 → 11 → 10
DETAILS:
  NodeNumber(10) FileNumber(Message_Queue.top) Framework(1) Component(1.4)
  NodeNumber(11) FileNumber(Message_Queue.ist) Framework(1) Component(1.4)
  NodeNumber(10) FileNumber(Message_Queue.top) Framework(1) Component(1.4)
VIOLATION(Predicate 4_13): FileInclusionLOOP has been detected:
  11 → 10 → 11
DETAILS:
  NodeNumber(11) FileNumber(Message_Queue.ist) Framework(1) Component(1.4)
  NodeNumber(10) FileNumber(Message_Queue.top) Framework(1) Component(1.4)
  NodeNumber(11) FileNumber(Message_Queue.ist) Framework(1) Component(1.4)
VIOLATION(Predicate 4_13): FileInclusionLOOP has been detected:
  13 → 14 → 13
DETAILS:
  NodeNumber(13) FileNumber(Module.top) Framework(1) Component(1.5)
  NodeNumber(14) FileNumber(Module.ist) Framework(1) Component(1.5)
  NodeNumber(13) FileNumber(Module.top) Framework(1) Component(1.5)
VIOLATION(Predicate 4_13): FileInclusionLOOP has been detected:
  13 → 23 → 22 → 17 → 14 → 13
DETAILS:
  NodeNumber(13) FileNumber(Module.top) Framework(1) Component(1.5)
  NodeNumber(23) FileNumber(Stream_Modules.ist) Framework(1) Component(1.8)
  NodeNumber(22) FileNumber(Stream_Modules.top) Framework(1) Component(1.8)
  NodeNumber(17) FileNumber(Multiplexor.ist) Framework(1) Component(1.6)
  NodeNumber(14) FileNumber(Module.ist) Framework(1) Component(1.5)
  NodeNumber(13) FileNumber(Module.top) Framework(1) Component(1.5)
VIOLATION(Predicate 4_13): FileInclusionLOOP has been detected:
  14 → 13 → 14
DETAILS:
  NodeNumber(14) FileNumber(Module.ist) Framework(1) Component(1.5)
  NodeNumber(13) FileNumber(Module.top) Framework(1) Component(1.5)
  NodeNumber(14) FileNumber(Module.ist) Framework(1) Component(1.5)
VIOLATION(Predicate 4_13): FileInclusionLOOP has been detected:
  14 → 13 → 23 → 22 → 17 → 14
DETAILS:
  NodeNumber(14) FileNumber(Module.ist) Framework(1) Component(1.5)
  NodeNumber(13) FileNumber(Module.top) Framework(1) Component(1.5)
  NodeNumber(23) FileNumber(Stream_Modules.ist) Framework(1) Component(1.8)
  NodeNumber(22) FileNumber(Stream_Modules.top) Framework(1) Component(1.8)
  NodeNumber(17) FileNumber(Multiplexor.ist) Framework(1) Component(1.6)
  NodeNumber(14) FileNumber(Module.ist) Framework(1) Component(1.5)
Appendix A4: A Prototype For CO-Toolkits' File Structures: Example#3 – ACE Sample File Structure

VIOLATION(Predicate4_13): FileInclusion_LOOP has been detected:
17 → 14 → 13 → 23 → 22 → 17
DETAILS:
  NodeNumber(17) fileName(Multiplexor.asm) Framework(1) Component(1.6)
  NodeNumber(14) fileName(Module.im) Framework(1) Component(1.3)
  NodeNumber(13) fileName(Module.top) Framework(1) Component(1.5)
  NodeNumber(23) fileName(Stream_Modules.asm) Framework(1) Component(1.8)
  NodeNumber(22) fileName(Stream_Modules.top) Framework(1) Component(1.8)
  NodeNumber(17) fileName(Multiplexor.asm) Framework(1) Component(1.6)

VIOLATION(Predicate4_13): FileInclusion_LOOP has been detected:
19 → 20 → 19
DETAILS:
  NodeNumber(19) fileName(Stream.top) Framework(1) Component(1.7)
  NodeNumber(20) fileName(Stream.im) Framework(1) Component(1.7)
  NodeNumber(19) fileName(Stream.top) Framework(1) Component(1.7)

VIOLATION(Predicate4_13): FileInclusion_LOOP has been detected:
20 → 19 → 20
DETAILS:
  NodeNumber(20) fileName(Stream.im) Framework(1) Component(1.7)
  NodeNumber(19) fileName(Stream.top) Framework(1) Component(1.7)
  NodeNumber(20) fileName(Stream.im) Framework(1) Component(1.7)

VIOLATION(Predicate4_13): FileInclusion_LOOP has been detected:
22 → 17 → 14 → 13 → 23 → 22
DETAILS:
  NodeNumber(22) fileName(Stream_Modules.top) Framework(1) Component(1.8)
  NodeNumber(17) fileName(Multiplexor.asm) Framework(1) Component(1.6)
  NodeNumber(14) fileName(Module.im) Framework(1) Component(1.5)
  NodeNumber(13) fileName(Module.top) Framework(1) Component(1.5)
  NodeNumber(23) fileName(Stream_Modules.asm) Framework(1) Component(1.8)
  NodeNumber(22) fileName(Stream_Modules.top) Framework(1) Component(1.8)

VIOLATION(Predicate4_13): FileInclusion_LOOP has been detected:
23 → 22 → 17 → 14 → 13 → 23
DETAILS:
  NodeNumber(23) fileName(Stream_Modules.asm) Framework(1) Component(1.8)
  NodeNumber(22) fileName(Stream_Modules.top) Framework(1) Component(1.8)
  NodeNumber(17) fileName(Multiplexor.asm) Framework(1) Component(1.6)
  NodeNumber(14) fileName(Module.im) Framework(1) Component(1.5)
  NodeNumber(13) fileName(Module.top) Framework(1) Component(1.5)
  NodeNumber(23) fileName(Stream_Modules.asm) Framework(1) Component(1.8)

VIOLATION(Predicate4_13): FileInclusion_LOOP has been detected:
31 → 32 → 31
DETAILS:
  NodeNumber(31) fileName(CORBA_Ref.top) Framework(2) Component(2.2)
  NodeNumber(32) fileName(CORBA_Ref.im) Framework(2) Component(2.2)
  NodeNumber(31) fileName(CORBA_Ref.top) Framework(2) Component(2.2)

VIOLATION(Predicate4_13): FileInclusion_LOOP has been detected:
32 → 31 → 32
DETAILS:
  NodeNumber(32) fileName(CORBA_Ref.im) Framework(2) Component(2.2)
  NodeNumber(31) fileName(CORBA_Ref.top) Framework(2) Component(2.2)
  NodeNumber(32) fileName(CORBA_Ref.im) Framework(2) Component(2.2)

VIOLATION(Predicate4_13): FileInclusion_LOOP has been detected:
34 → 35 → 34
DETAILS:
  NodeNumber(34) fileName(Set.top) Framework(3) Component(3.1)
  NodeNumber(35) fileName(Set.im) Framework(3) Component(3.1)
  NodeNumber(34) fileName(Set.top) Framework(3) Component(3.1)

VIOLATION(Predicate4_13): FileInclusion_LOOP has been detected:
35 → 34 → 35
DETAILS:
  NodeNumber(35) fileName(Set.im) Framework(3) Component(3.1)
  NodeNumber(34) fileName(Set.top) Framework(3) Component(3.1)
  NodeNumber(35) fileName(Set.im) Framework(3) Component(3.1)

VIOLATION(Predicate4_13): FileInclusion_LOOP has been detected:
37 → 38 → 37
DETAILS:
  NodeNumber(37) fileName(Stack.top) Framework(3) Component(3.2)
  NodeNumber(38) fileName(Stack.im) Framework(3) Component(3.2)
  NodeNumber(37) fileName(Stack.top) Framework(3) Component(3.2)

VIOLATION(Predicate4_13): FileInclusion_LOOP has been detected:
38 → 37 → 38
DETAILS:
  NodeNumber(38) fileName(Stack.im) Framework(3) Component(3.2)
  NodeNumber(37) fileName(Stack.top) Framework(3) Component(3.2)
  NodeNumber(38) fileName(Stack.im) Framework(3) Component(3.2)
Appendix A4: A Prototype For CO-Toolkits' File Structures: Example#3 – ACE Sample File Structure

VIOLATION(Predicate 4_13): FileInclusionLOOP has been detected:
46 l–> 47 l–> 46
DETAILS:
NodeNumber(46) FileName(Fuse.top) Framework(4) Component(4,2)
NodeNumber(47) FileName(Fuse.iim) Framework(4) Component(4,2)
NodeNumber(46) FileName(Fuse.top) Framework(4) Component(4,2)

VIOLATION(Predicate 4_13): FileInclusionLOOP has been detected:
47 l–> 46 l–> 47
DETAILS:
NodeNumber(47) FileName(Fuse.iim) Framework(4) Component(4,2)
NodeNumber(46) FileName(Fuse.top) Framework(4) Component(4,2)
NodeNumber(47) FileName(Fuse.iim) Framework(4) Component(4,2)

VIOLATION(Predicate 4_13): FileInclusionLOOP has been detected:
56 l–> 62 l–> 56
DETAILS:
NodeNumber(56) FileName(Synch.iim) Framework(4) Component(4,5)
NodeNumber(62) FileName(Synch_Timem) Framework(4) Component(4,7)
NodeNumber(56) FileName(Synch.iim) Framework(4) Component(4,5)

VIOLATION(Predicate 4_13): FileInclusionLOOP has been detected:
56 l–> 62 l–> 143 l–> 56 l–> 56
DETAILS:
NodeNumber(56) FileName(Synch.iim) Framework(4) Component(4,5)
NodeNumber(62) FileName(Synch_Timem) Framework(4) Component(4,7)
NodeNumber(61) FileName(Synch_Timem) Framework(4) Component(4,7)
NodeNumber(143) FileName(Time_Value.iim) Framework(11) Component(11,8)
NodeNumber(56) FileName(Synch.iim) Framework(4) Component(4,5)

VIOLATION(Predicate 4_13): FileInclusionLOOP has been detected:
61 l–> 62 l–> 61
DETAILS:
NodeNumber(61) FileName(Synch_Timem) Framework(4) Component(4,7)
NodeNumber(62) FileName(Synch_Timem) Framework(4) Component(4,7)
NodeNumber(61) FileName(Synch_Timem) Framework(4) Component(4,7)

VIOLATION(Predicate 4_13): FileInclusionLOOP has been detected:
61 l–> 143 l–> 56 l–> 62 l–> 61
DETAILS:
NodeNumber(61) FileName(Synch_Timem) Framework(4) Component(4,7)
NodeNumber(143) FileName(Time_Value.iim) Framework(11) Component(11,8)
NodeNumber(56) FileName(Synch.iim) Framework(4) Component(4,5)
NodeNumber(62) FileName(Synch_Timem) Framework(4) Component(4,7)
NodeNumber(61) FileName(Synch_Timem) Framework(4) Component(4,7)

VIOLATION(Predicate 4_13): FileInclusionLOOP has been detected:
62 l–> 56 l–> 62
DETAILS:
NodeNumber(62) FileName(Synch_Timem) Framework(4) Component(4,7)
NodeNumber(56) FileName(Synch.iim) Framework(4) Component(4,5)
NodeNumber(62) FileName(Synch_Timem) Framework(4) Component(4,7)

VIOLATION(Predicate 4_13): FileInclusionLOOP has been detected:
62 l–> 61 l–> 62
DETAILS:
NodeNumber(62) FileName(Synch_Timem) Framework(4) Component(4,7)
NodeNumber(61) FileName(Synch_Timem) Framework(4) Component(4,7)
NodeNumber(62) FileName(Synch_Timem) Framework(4) Component(4,7)

VIOLATION(Predicate 4_13): FileInclusionLOOP has been detected:
DETAILS:
NodeNumber(62) FileName(Synch_Timem) Framework(4) Component(4,7)
NodeNumber(61) FileName(Synch_Timem) Framework(4) Component(4,7)
NodeNumber(143) FileName(Time_Value.iim) Framework(11) Component(11,8)
NodeNumber(56) FileName(Synch.iim) Framework(4) Component(4,5)
NodeNumber(62) FileName(Synch_Timem) Framework(4) Component(4,7)

VIOLATION(Predicate 4_13): FileInclusionLOOP has been detected:
73 l–> 74 l–> 73
DETAILS:
NodeNumber(73) FileName(Concatexx.top) Framework(5) Component(5,1)
NodeNumber(74) FileName(Concatexx.iim) Framework(5) Component(5,1)
NodeNumber(73) FileName(Concatexx.top) Framework(5) Component(5,1)

VIOLATION(Predicate 4_13): FileInclusionLOOP has been detected:
74 l–> 73 l–> 74
DETAILS:
NodeNumber(74) FileName(Concatexx.iim) Framework(5) Component(5,1)
NodeNumber(73) FileName(Concatexx.top) Framework(5) Component(5,1)
NodeNumber(74) FileName(Concatexx.iim) Framework(5) Component(5,1)

VIOLATION(Predicate 4_13): FileInclusionLOOP has been detected:
761 l–> 77 l–> 76
DETAILS:
Appendix A4: A Prototype For CO-Toolkits' File Structures: Example#3 – ACE Sample File Structure

NodeNumber(76) FileName(Connector.top) Framework(5) Component(5.2)
NodeNumber(76) FileName(Connector.iml) Framework(5) Component(5.2)
VIOLATION(Predicate_4.13): FileInclusionLOOP has been detected:
77 l-> 76 l-> 77
DETAILS:
NodeNumber(77) FileName(Connector.iml) Framework(5) Component(5.2)
NodeNumber(76) FileName(Connector.top) Framework(5) Component(5.2)
NodeNumber(77) FileName(Connector.iml) Framework(5) Component(5.2)
VIOLATION(Predicate_4.13): FileInclusionLOOP has been detected:
79 l-> 80 l-> 79
DETAILS:
NodeNumber(79) FileName(Dynamic_Service.top) Framework(5) Component(5.3)
NodeNumber(80) FileName(Dynamic_Service.iml) Framework(5) Component(5.3)
NodeNumber(79) FileName(Dynamic_Service.top) Framework(5) Component(5.3)
VIOLATION(Predicate_4.13): FileInclusionLOOP has been detected:
80 l-> 79 l-> 80
DETAILS:
NodeNumber(80) FileName(Dynamic_Service.iml) Framework(5) Component(5.3)
NodeNumber(79) FileName(Dynamic_Service.top) Framework(5) Component(5.3)
NodeNumber(80) FileName(Dynamic_Service.iml) Framework(5) Component(5.3)
VIOLATION(Predicate_4.13): FileInclusionLOOP has been detected:
85 l-> 86 l-> 85
DETAILS:
NodeNumber(86) FileName(Svc_Handler.top) Framework(5) Component(5.5)
NodeNumber(86) FileName(Svc_Handler.iml) Framework(5) Component(5.5)
NodeNumber(86) FileName(Svc_Handler.top) Framework(5) Component(5.5)
VIOLATION(Predicate_4.13): FileInclusionLOOP has been detected:
86 l-> 85 l-> 86
DETAILS:
NodeNumber(86) FileName(Svc_Handler.iml) Framework(5) Component(5.5)
NodeNumber(85) FileName(Svc_Handler.top) Framework(5) Component(5.5)
NodeNumber(86) FileName(Svc_Handler.iml) Framework(5) Component(5.5)
VIOLATION(Predicate_4.13): FileInclusionLOOP has been detected:
122 l-> 123 l-> 122
DETAILS:
NodeNumber(122) FileName(Event_Handler.iml) Framework(11) Component(11.1)
NodeNumber(122) FileName(Event_Handler.iml) Framework(11) Component(11.1)
NodeNumber(122) FileName(Event_Handler.iml) Framework(11) Component(11.1)
VIOLATION(Predicate_4.13): FileInclusionLOOP has been detected:
123 l-> 122 l-> 123
DETAILS:
NodeNumber(122) FileName(Event_Handler.iml) Framework(11) Component(11.1)
NodeNumber(122) FileName(Event_Handler.iml) Framework(11) Component(11.1)
NodeNumber(122) FileName(Event_Handler.iml) Framework(11) Component(11.1)
VIOLATION(Predicate_4.13): FileInclusionLOOP has been detected:
143 l-> 56 l-> 62 l-> 61 l-> 143
DETAILS:
NodeNumber(143) FileName(Time_Value.iml) Framework(11) Component(11.8)
NodeNumber(56) FileName(Synch.iml) Framework(4) Component(4.5)
NodeNumber(52) FileName(Synch_Tim.iml) Framework(4) Component(4.7)
NodeNumber(61) FileName(Synch_Tim.top) Framework(4) Component(4.7)
NodeNumber(143) FileName(Time_Value.iml) Framework(11) Component(11.8)
Checking for Predicate 4.12 is complete —— STATUS: false

Checking for Predicate 4.8 is complete —— STATUS: false

Checking for GoodFileStructure Predicate 4.5 is complete —— STATUS: false

220
APPENDIX B

A PROTOTYPE FOR CO–TOOLKITS’ DESIGN AND IMPLEMENTATION STRUCTURES

(1) Appendix B1: A Prototype For CO–Toolkits’ Design and Implementation
   Structures: An Overview ———————————— P223–330

(2) Appendix B2: A Prototype For CO–Toolkits’ Design and Implementation
   Structures: Example#1 – An Artificial Example ———— P331–336

(3) Appendix B3: A Prototype For CO–Toolkits’ Design and Implementation
   Structures: Example#2 – An Artificial Example ———— P337–342

(4) Appendix B4: A Prototype For CO–Toolkits’ Design and Implementation
   Structures: Example#3 – ACE Sample Inheritance Structure ———— P343–366
Appendix B1

A Prototype For CO–Toolkits’ Design and Implementation Structures: An Overview

B1.1. Overview

Similar to the previous prototype (c.f. Appendix A), Java has been used to build this prototype based on the view for CO–Toolkits’ design and implementation structures (c.f. chapter 4). The prototype’s data structure represents the view structure of section 5.2.2. Each predicate from the view’s characteristics section (c.f. section 5.2.3) is implemented as method that returns a boolean value (i.e. TRUE or FALSE). For convenience, we named each of these methods by its formula number as in chapter 5.

Similar to the previous prototype, we kept this prototype simple. All of its inputs are built–in. Currently, we have three built–in examples on design and implementation, one of them is on a sample ACE inheritance structure.

To run this prototype:

1. The twenty two files listed in the next section should be in a directory named V2_DesignAndImplementation_R3. In Java, path and file names are case sensitive.

2. Run the command

   javac -deprecation V2_DesignAndImplementation_R3/*.java
(3) Run the command

```
java V2_DesignAndImplementation_R3.CheckDesignAndImplementation N
```

where N is the argument. It represents the example number of choice (i.e. 1 for example 1, 2 for example 2, and 3 for the ACE example).

The prototype operation goes through the following steps:

1. It assigns values for the various constants such as $F$ (i.e. number of frameworks), $S$ (i.e. number of shared pools), $C[]$ (i.e. numbers of components in each framework), $c[][]$ (numbers of classes in each component), $MP[][][]$ (i.e. number of public methods in each class), $MR[][][]$ (i.e. number of protected methods in each class), $MV[][][]$ (i.e. number of private methods in each class), $OP[][][]$ (i.e. number of public objects in each class), and etc.

2. It allocates its internal data structure based on the view structure in section 5.2.2.

3. It updates its internal data structures with the classes names, methods names, objects names, and inheritance relations.

4. It checks its internal data structure by sequentially invoking its methods that represent the various predicate (i.e. rules) in the view characteristic section (c.f. section 5.2.3). Each of these methods reports any violation to its embedded rule.

### B1.2. Source Code Description

This section describes this prototype's source files:

#### B1.2.1. CheckDesignAndImplementation.java

This file contains the implementation of all predicates that represent the characteristics of the Design and Implementation's View. These predicates are named by their formula numbers as in chapter 5. For example, predicate5–40 is the implementation of characteristic (5–40). Next is the source code listing for this file:
package V2_DesignAndImplementation_R3;

/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#2 For CO_Toolkits Design&Implementation — Chapter 5 of My Ph.D. Thesis
 * File: CheckDesignAndImplementation.java
 */

public class CheckDesignAndImplementation_R3 {
    private static String output;
    public CheckDesignAndImplementation_R3() {
        super();
    }
    private static boolean Associates_Bi_ClassLevel(int fr_id1, int co_id1, int cl_id1, int fr_id2, int co_id2, int cl_id2) {
        boolean returnV1 = One_to_One_UniDirectionalAssociation_ClassLevel(fr_id1, co_id1, cl_id1, fr_id2, co_id2, cl_id2);
        boolean returnV2 = One_to_One_BiDirectionalAssociation_ClassLevel(fr_id1, co_id1, cl_id1, fr_id2, co_id2, cl_id2);
        boolean returnV3 = One_to_M_UniDirectionalAssociation_ClassLevel(fr_id1, co_id1, cl_id1, fr_id2, co_id2, cl_id2);
        boolean returnV4 = One_to_M_BiDirectionalAssociation_ClassLevel(fr_id1, co_id1, cl_id1, fr_id2, co_id2, cl_id2);
        boolean returnV5 = M_to_M_BiDirectionalAssociation_ClassLevel(fr_id1, co_id1, cl_id1, fr_id2, co_id2, cl_id2);
        boolean returnV6 = (returnV1 || returnV2) || (returnV3 || returnV4) || returnV5;
        return returnV6;
    }
    private static boolean Associates_Bi_SharedClassLevel(int shp_id1, int shc_id1, int shp_id2, int shc_id2) {
        boolean returnV1 = One_to_One_UniDirectionalAssociation_SharedClassLevel(shp_id1, shc_id1, shp_id2, shc_id2);
        boolean returnV2 = One_to_One_BiDirectionalAssociation_SharedClassLevel(shp_id1, shc_id1, shp_id2, shc_id2);
        boolean returnV3 = One_to_M_UniDirectionalAssociation_SharedClassLevel(shp_id1, shc_id1, shp_id2, shc_id2);
        boolean returnV4 = One_to_M_BiDirectionalAssociation_SharedClassLevel(shp_id1, shc_id1, shp_id2, shc_id2);
        boolean returnV5 = M_to_M_BiDirectionalAssociation_SharedClassLevel(shp_id1, shc_id1, shp_id2, shc_id2);
        boolean returnV6 = (returnV1 || returnV2) || (returnV3 || returnV4) || returnV5;
        return returnV6;
    }
    private static boolean Associates_ClassLevel(int fr_id1, int co_id1, int cl_id1, int fr_id2, int co_id2, int cl_id2) {
        boolean returnV1 = One_to_One_UniDirectionalAssociation_ClassLevel(fr_id1, co_id1, cl_id1, fr_id2, co_id2, cl_id2);
        boolean returnV2 = One_to_One_BiDirectionalAssociation_ClassLevel(fr_id1, co_id1, cl_id1, fr_id2, co_id2, cl_id2);
        boolean returnV3 = One_to_M_UniDirectionalAssociation_ClassLevel(fr_id1, co_id1, cl_id1, fr_id2, co_id2, cl_id2);
        boolean returnV4 = One_to_M_BiDirectionalAssociation_ClassLevel(fr_id1, co_id1, cl_id1, fr_id2, co_id2, cl_id2);
        boolean returnV5 = M_to_M_BiDirectionalAssociation_ClassLevel(fr_id1, co_id1, cl_id1, fr_id2, co_id2, cl_id2);
        boolean returnV6 = returnV1 || returnV2 || returnV3 || returnV4 || returnV5;
    }
}
return return Value;
}
private static boolean Associates_ComponentLevel(int fr_id1, int co_id1, int fr_id2, int co_id2) {
    boolean return Value1 = One_to_One_UniDirectionalAssociation_ComponentLevel(fr_id1, co_id1, fr_id2, co_id2);
    boolean return Value2 = One_to_One_BiDirectionalAssociation_ComponentLevel(fr_id1, co_id1, fr_id2, co_id2);
    boolean return Value3 = One_to_M_UniDirectionalAssociation_ComponentLevel(fr_id1, co_id1, fr_id2, co_id2);
    boolean return Value4 = One_to_M_BiDirectionalAssociation_ComponentLevel(fr_id1, co_id1, fr_id2, co_id2);
    boolean return Value5 = M_to_M_BiDirectionalAssociation_ComponentLevel(fr_id1, co_id1, fr_id2, co_id2);
    boolean return Value = return Value1 || return Value2 || return Value3 || return Value4 || return Value5;
    return return Value;
}
private static boolean Associates_FrameworkLevel(int fr_id1, int fr_id2) {
    boolean return Value1 = One_to_One_UniDirectionalAssociation_FrameworkLevel(fr_id1, fr_id2);
    boolean return Value2 = One_to_One_BiDirectionalAssociation_FrameworkLevel(fr_id1, fr_id2);
    boolean return Value3 = One_to_M_UniDirectionalAssociation_FrameworkLevel(fr_id1, fr_id2);
    boolean return Value4 = One_to_M_BiDirectionalAssociation_FrameworkLevel(fr_id1, fr_id2);
    boolean return Value5 = M_to_M_BiDirectionalAssociation_FrameworkLevel(fr_id1, fr_id2);
    boolean return Value = return Value1 || return Value2 || return Value3 || return Value4 || return Value5;
    return return Value;
}
private static boolean Associates_SharedClassLevel(int shp_id1, int shc_id1, int shp_id2, int shc_id2) {
    boolean return Value1 = One_to_One_UniDirectionalAssociation_SharedClassLevel(shp_id1, shc_id1, shp_id2, shc_id2);
    boolean return Value2 = One_to_One_BiDirectionalAssociation_SharedClassLevel(shp_id1, shc_id1, shp_id2, shc_id2);
    boolean return Value3 = One_to_M_UniDirectionalAssociation_SharedClassLevel(shp_id1, shc_id1, shp_id2, shc_id2);
    boolean return Value4 = One_to_M_BiDirectionalAssociation_SharedClassLevel(shp_id1, shc_id1, shp_id2, shc_id2);
    boolean return Value5 = M_to_M_BiDirectionalAssociation_SharedClassLevel(shp_id1, shc_id1, shp_id2, shc_id2);
    boolean return Value = return Value1 || return Value2 || return Value3 || return Value4 || return Value5;
    return return Value;
}
private static boolean Associates_SharedPoolLevel(int shp_id1, int shp_id2) {
    boolean return Value1 = One_to_One_UniDirectionalAssociation_SharedPoolLevel(shp_id1, shp_id2);
    boolean return Value2 = One_to_One_BiDirectionalAssociation_SharedPoolLevel(shp_id1, shp_id2);
    boolean return Value3 = One_to_M_UniDirectionalAssociation_SharedPoolLevel(shp_id1, shp_id2);
    boolean return Value4 = One_to_M_BiDirectionalAssociation_SharedPoolLevel(shp_id1, shp_id2);
    boolean return Value5 = M_to_M_BiDirectionalAssociation_SharedPoolLevel(shp_id1, shp_id2);
    boolean return Value = return Value1 || return Value2 || return Value3 || return Value4 || return Value5;
    return return Value;
}
private static boolean Associates_Uni_ClassLevel(int fr_id1, int co_id1, int cl_id1, int fr_id2, int co_id2, int cl_id2) {
    boolean return Value1 = One_to_One_UniDirectionalAssociation_Uni_ClassLevel(fr_id1, co_id1, cl_id1, fr_id2, co_id2, cl_id2);
    boolean return Value2 = One_to_One_BiDirectionalAssociation_Uni_ClassLevel(fr_id1, co_id1, cl_id1, fr_id2, co_id2, cl_id2);
    boolean return Value3 = One_to_M_UniDirectionalAssociation_Uni_ClassLevel(fr_id1, co_id1, cl_id1, fr_id2, co_id2, cl_id2);
    boolean return Value4 = One_to_M_BiDirectionalAssociation_Uni_ClassLevel(fr_id1, co_id1, cl_id1, fr_id2, co_id2, cl_id2);
    boolean return Value = return Value1 || return Value2 || return Value3 || return Value4;
    return return Value;
}
Appendix B1: A Prototype For CO-Toolkits’ Design and Implementation Structures: An Overview

```java
fr_id2, co_id2, cl_id2);
boolean returnValuen5=M_to_M_BiDirectionalAssociation_ClassLevel(fr_id1, co_id1, cl_id1,
fr_id2, co_id2, cl_id2);
boolean returnValue=(((returnValue1)&&(returnValue2)&&(returnValue3)&&(returnValue4)&&(returnValue5));
return returnValue;
}
private static boolean Associates_Uni_SharedClassLevel(int shp_id1, int shc_id1,int shp_id2, int shc_id2) {
    boolean returnValue1=One_to_One UniDirectionalAssociation_SharedClassLevel(shp_id1,shc_id1,shp_id2,shc_id2);
    boolean returnValue2=One_to_One_BiDirectionalAssociation_SharedClassLevel(shp_id1,shc_id1,shp_id2,shc_id2);
    boolean returnValue3=One_to_M UniDirectionalAssociation_SharedClassLevel(shp_id1,shc_id1,shp_id2,shc_id2);
    boolean returnValue4=One_to_M_BiDirectionalAssociation_SharedClassLevel(shp_id1,shc_id1,shp_id2,shc_id2);
    boolean returnValue5=M_to_M_BiDirectionalAssociation_SharedClassLevel(shp_id1,shc_id1,shp_id2,shc_id2);
    boolean returnValue= ((returnValue1)&&(returnValue2)&&(returnValue3)&&(returnValue4)&&(returnValue5));
    return returnValue;
}
private static void displayDesignAndImplementationStructure() {} 
private static void eXample_10 (DesignAndImplementationStructure.initializeFor_Example_10(); 
private static void eXample_20 (DesignAndImplementationStructure.initializeFor_Example_20();
private static void eXample_3 ACE() (DesignAndImplementationStructure.initializeFor_Example_3 ACE();
private static boolean inherits ClassLevel(int fr_id1, int co_id1,int cl_id1,
                        int fr_id2, int co_id2, int cl_id2){
    boolean returnValue=false;
    if((DesignAndImplementationStructure.Frames[fr_id1].Component[co_id1].Class[cl_id1]!=null) &&
    (DesignAndImplementationStructure.Frames[fr_id2].Component[co_id2].Class[cl_id2]!=null)){
        int ii=1;
        int cnt=0;
        while ((DesignAndImplementationStructure.Frames[fr_id1].Component[co_id1].Class[cl_id1].PublicInheritedClasses[ii]!=null){
            int inh_fr_id=
            int inh_co_id=
            int inh_cl_id=
            DesignAndImplementationStructure.Frames[fr_id1].Component[co_id1].Class[cl_id1].PublicInheritedClasses[ii].ClassNameNodeNumber;
            if ((inh_fr_id==fr_id2)&&(inh_co_id==co_id2)&&(inh_cl_id==cl_id2)){
                returnValue=true;
                break;
            }
            ii=ii+1;
        }
    }
    return returnValue;
```
    int inh_co_id =
    int inh_cl_id =
    if ((inh_fr_id==fr_id2) && (inh_co_id==co_id2) && (inh_cl_id==cl_id2)) {
        returnValue=true;
        reportViolation("Class("+fr_id1+","+co_id1+","+cl_id1+") inherits in protected from
Class("+fr_id2+","+co_id2+","+cl_id2+");
        break;
    }
    ii=ii+1;
}
}
if ((DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[cl_id1]!=null) &&
(DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[cl_id2]!=null)) {
    int ii=1;
    int cnt=0;
    while (DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[cl_id1].PrivateInheritedClasses[ii]!=null) {
        int inh_fr_id =
        int inh_co_id =
        int inh_cl_id =
        if ((inh_fr_id==fr_id2) && (inh_co_id==co_id2) && (inh_cl_id==cl_id2)) {
            returnValue=true;
            reportViolation("Class("+fr_id1+","+co_id1+","+cl_id1+") inherits in private from
Class("+fr_id2+","+co_id2+","+cl_id2+");
            break;
        }
        ii=ii+1;
    }
    return returnValue;
} /*****
 * @return boolean
 */
private static boolean Inherits_SharedClassLevel(int shp_id1, int shc_id1, int shp_id2, int shc_id2) {
    boolean returnValue=false;
    if ((DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1]!=null) &&
(DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2]!=null)) {
        int ii=1;
        int cnt=0;
        while (DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].PublicInheritedSharedClasses[ii]!=null) {
            int inh_shp_id =
            int inh_shc_id =
DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].PublicInheritedSharedClasses[ii].ClassName=Number;
    if ((inh_shp_id==shp_id2) &(inh_shc_id==shc_id2)) {
        returnValue=true;
        break;
    }
    ii=ii+1;

}

if((DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1]!=null) &&
(DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2]!=null))
    int ii=1;
    int cnt=0;
    while ((DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].ProtectedInheritedSharedClasses[ii]!=null)) {
        int inh_shc_id=DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].ProtectedInheritedSharedClasses[ii].ClassName=Number;
        if ((inh_shp_id==shp_id2) &(inh_shc_id==shc_id2)) {
            returnValue=true;
            reportViolation("SharedClass("+shp_id1+","+shc_id1+)" inherits in protected from Shared-Class("+shp_id2+","+shc_id2+)");
            break;
        }
        ii=ii+1;
    }

    if((DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1]!=null) &&
(DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2]!=null)) {
    int ii=1;
    int cnt=0;
    while ((DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].PrivateInheritedSharedClasses[ii]!=null)) {
        int inh_shc_id=DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].PrivateInheritedSharedClasses[ii].ClassName=Number;
        if ((inh_shp_id==shp_id2) &(inh_shc_id==shc_id2)) {
            returnValue=true;
            reportViolation("SharedClass("+shp_id1+","+shc_id1+)" inherits in private from Shared-Class("+shp_id2+","+shc_id2+)");
            break;
        }
        ii=ii+1;
    }
    return returnValue;
}
public static void initializeDesignAndImplementationStructure() {
    if (Globals.EXAMPLE_NOM==1){
        eXample_10();
    }else{
        if (Globals.EXAMPLE_NOM==2){
            eXample_20();
        }else{
            if (Globals.EXAMPLE_NOM==3){
                eXample_3_ACE();
            }
        }
    }
}
private static boolean M_to_M_BiDirectionalAssociation_ClassLevel(
    int fr_id1, int co_id1, int cl_id1,
    int fr_id2, int co_id2, int cl_id2)
{
    boolean return Value = true;
    if (DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[cl_id1] != null &&
    {
        int ii = 1;
        int cnt = 0;
        String className = DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[cl_id2].ID.ClassName;
        while (!DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[cl_id1].ProtectedSection.Object[ii].name.equals(className))
        {
            objType = DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[cl_id1].ProtectedSection.Object[ii].type;
            if (objType.equals(className))
            {
                cnt = cnt + 1;
            }
            ii = ii + 1;
        }
        while (!DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[cl_id1].PrivateSection.Object[ii].name.equals(className))
        {
            if (objType.equals(className))
            {
                cnt = cnt + 1;
            }
            ii = ii + 1;
        }
        if (cnt > 1)
        {
            ii = 1;
            cnt = 0;
            className = DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[cl_id1].ID.ClassName;
            while (!DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[cl_id2].PublicSection.Object[ii].name.equals(className))
            {
                ii = ii + 1;
            }
        }
        else
        {
            System.out.println("Wrong Example Number. It has to be 1, 2, or 3");
            return false;
        }
    }
    return true;
}

// need to determine what parameters to pass
private static boolean M_to_M_BiDirectionalAssociation_ClassLevel(
    int fr_id1, int co_id1, int cl_id1,
    int fr_id2, int co_id2, int cl_id2)
{
if (objType==className){
cnt2=cnt2+1;
}
ii=ii+1;
}
i=1;
while (DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[cl_id2].ProtectedSection.Object[ii]!=null){
  objType=DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[cl_id2].ProtectedSection.ObjectType[ii];
  if (objType==className){
cnt2=cnt2+1;
  }
  ii=ii+1;
  }
i=1;
while (DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[cl_id2].PrivateSection.Object[ii]!=null){
  objType=DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[cl_id2].PrivateSection.ObjectType[ii];
  if (objType==className){
cnt2=cnt2+1;
  }
  ii=ii+1;
  }
  if (cnt2<2){returnValue=false;}
}
else{returnValue=false; }
else{returnValue=false; }
return returnValue;
}/**
 * @return boolean
 */
// need to determine what parameters to pass
private static boolean M_to_M_BiDirectionalAssociation_ComponentLevel(
  int fr_id1, int co_id1,
  int fr_id2, int co_id2){ boolean returnValue=true;

int ii=1;
while (DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[ii]!=null){
  int cl_id1=DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[ii].ID.ClassNodeNumber;
  int jj=1;
  while (DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[jj]!=null){
    int cl_id2=DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[jj].ID.ClassNodeNumber;
    returnValue=M_to_M_BiDirectionalAssociation_ComponentLevel(fr_id1,co_id1,cl_id1,fr_id2,co_id2,cl_id2);
    if (!returnValue){break;}
    jj=jj+1;
  }
  if (!returnValue){break;}
  ii=ii+1;
}
return returnValue;
}/**
* @return boolean
* /
// need to determine what parameters to pass
private static boolean M_to_M_BiDirectionalAssociation_FrameworkLevel(int fr_id1,int fr_id2){
    boolean returnValue=true;
    int ii=1;
    while (DesignAndImplementationStructure.Frameworks[fr_id1].Component[ii]!=null){
        int co_id1=DesignAndImplementationStructure.Frameworks[fr_id1].Component[ii].ID.ComponentNodeNumber;
        int jj=1;
        while (DesignAndImplementationStructure.Frameworks[fr_id2].Component[jj]!=null){
            int co_id2=DesignAndImplementationStructure.Frameworks[fr_id2].Component[jj].ID.ComponentNodeNumber;
            returnValue=M_to_M_BiDirectionalAssociation_ComponentLevel(fr_id1,co_id1,fr_id2,co_id2);
            if (!returnValue){break;}
            jj=jj+1;
        }
        if (!returnValue){break;}
        ii=ii+1;
    }
    return returnValue;
}
private static boolean M_to_M_BiDirectionalAssociation_SharedClassLevel{
    int shp_id1, int shc_id1,
    int shp_id2, int shc_id2{
    boolean returnValue=true;
    if((DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1]!=null) &&
    (DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2]!=null)){
        int ii=1;
        int cnt=0;
        String className=DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].ClassName;
        while (DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].PublicSection.Object[ii]!=null){
            String objType=DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].PublicSection.Object[iii];
            if (objType==className){
                cnt=cnt+1;
            }
            ii=ii+1;
        }
        ii=1;
        while (DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].ProtectedSection.Object[ii]!=null){
            String objType=DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].ProtectedSection.Object[ii];
            if (objType==className){
                cnt=cnt+1;
            }
            ii=ii+1;
        }
        ii=1;
        while (DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].PrivateSection.Object[ii]!=null){
            String objType=DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].PrivateSection.Object[ii];
            if (objType==className){
                cnt=cnt+1;
            }
        }
    }
Appendix B1: A Prototype For CO-Toolkits’ Design and Implementation Structures: An Overview

```java
iii = iii + 1;
}
if (cnt1 > 1) {
  iii = 1;
  int cnt2 = 0;
  className = DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].ClassName;
  while (DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].PublicSection.Object[object][ii] != null) {
    String objType = DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].PublicSection.ObjectType[ii];
    if (objType == className) {
      cnt2 = cnt2 + 1;
    }
    iii = iii + 1;
  }
  i = 1;
  while (DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].ProtectedSection.Object[ii] != null) {
    String objType = DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].ProtectedSection.ObjectType[ii];
    if (objType == className) {
      cnt2 = cnt2 + 1;
    }
    iii = iii + 1;
  }
  ii = 1;
  while (DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].PrivateSection.Object[ii] != null) {
    String objType = DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].PrivateSection.ObjectType[ii];
    if (objType == className) {
      cnt2 = cnt2 + 1;
    }
    iii = iii + 1;
  }
  if (cnt2 < 2) { return Value = false; }
  else { return Value = false; }
}
else { return Value = false; }
return return Value;
/**
 * @return boolean
 */
// need to determine what parameters to pass
private static boolean M_to_M_BiDirectionalAssociation_SharedPoolLevel(int shp_id1, int shp_id2) {
  boolean returnValue = true;
  int iii = 1;
  while (DesignAndImplementationStructure.SharedPools[shp_id1].Class[iii] != null) {
    int shc_id1 = DesignAndImplementationStructure.SharedPools[shp_id1].Class[iii].ID.ParentSharedPoolNodeNumber;
    int jj = 1;
    while (DesignAndImplementationStructure.SharedPools[shp_id2].Class[jj] != null) {
      int shc_id2 = DesignAndImplementationStructure.SharedPools[shp_id2].Class[jj].ID.ClassNodeNumber;
      return Value = M_to_M_BiDirectionalAssociation_SharedClassLevel(shp_id1, shc_id1, shp_id2, shc_id2);
      if ((return Value) { break; })
      jj = jj + 1;
    }
  }
  return return Value;
}
```
if (!returnValue) {break;}
    ii=ii+1;
}
return returnValue;
}

public static void main(String args[]) {
    Globals.EXAMPLE_NAME= Integer.parseInt(args[0]);
    initializeDesignAndImplementationStructure();
    predicate5_18_GoodDesignAndImplementation();
}

private static boolean One_to_M_BiDirectionalAssociation_ClassLevel(
    int fr_id1, int co_id1, int cl_id1,
    int fr_id2, int co_id2, int cl_id2){
    boolean returnValue=true;
    if((DesignAndImplementationStructureFrameworks[fr_id1].Component[co_id1].Class[cl_id1]!=null) & &
    (DesignAndImplementationStructureFrameworks[fr_id2].Component[co_id2].Class[cl_id2]!=null)){
        int ii=1;
        int cnt=0;
        String className=DesignAndImplementationStructureFrameworks[fr_id2].Component[
            co_id2].Class[cl_id2].ID.ClassName;
        while (DesignAndImplementationStructureFrameworks[fr_id1].Component[co_id1].Class[cl_id1].P
            ublicSection.Object[ii]!=null){
            String
            objType=DesignAndImplementationStructureFrameworks[fr_id1].Component[
            co_id1].Class[cl_id1].PublicSection.ObjectType[ii];
            if (objType==className){
                cnt=cnt+1;
            }
            ii=ii+1;
        }
        ii=1;
        while (DesignAndImplementationStructureFrameworks[fr_id1].Component[co_id1].Class[cl_id1].P
            rotectedSection.Object[ii]!=null){
            String
            objType=DesignAndImplementationStructureFrameworks[fr_id1].Component[
            co_id1].Class[cl_id1].ProtectedSection.ObjectType[ii];
            if (objType==className){
                cnt=cnt+1;
            }
            ii=ii+1;
        }
        ii=1;
        while (DesignAndImplementationStructureFrameworks[fr_id1].Component[co_id1].Class[cl_id1].P
            rivateSection.Object[ii]!=null){
            String
            objType=DesignAndImplementationStructureFrameworks[fr_id1].Component[
            co_id1].Class[cl_id1].PrivateSection.ObjectType[ii];
            if (objType==className){
                cnt=cnt+1;
            }
            ii=ii+1;
        }
    }
    if (cnt>1){
        ii=1;
        int cnt2=0;
        className=DesignAndImplementationStructureFrameworks[fr_id1].Component[
            co_id1].Class[cl_id1].ID.ClassName;
        while (DesignAndImplementationStructureFrameworks[fr_id2].Component[co_id2].Class[cl_id2].P
            ublicSection.Object[ii]!=null){
            String
            objType=DesignAndImplementationStructureFrameworks[fr_id2].Component[
            co_id2].Class[cl_id2].PublicSection.ObjectType[ii];
            if (objType==className){

        
}
}
cnt2=cnt2+1;
if (cnt2>1) {returnValue=false; break; }
}
ii=ii+1;
}
ii=1;
while (DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[cl_id2].ProtectedSection.Object[ii]!=null) {
  String objType=DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[cl_id2].ProtectedSection.Object[ii].ObjectType;
  if (objType==className) {
    cnt2=cnt2+1;
    if (cnt2>1) {returnValue=false; break; }
  }
  ii=ii+1;
}
ii=1;
while (DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[cl_id2].PrivateSection.Object[ii]!=null) {
  String objType=DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[cl_id2].PrivateSection.Object[ii].ObjectType;
  if (objType==className) {
    cnt2=cnt2+1;
    if (cnt2>1) {returnValue=false; break; }
  }
  ii=ii+1;
}
else {returnValue=false; }
else {returnValue=false; }
return returnValue;
}
private static boolean One_to_M_BiDirectionalAssociation_ComponentLevel(
  int fr_id1, int co_id1,
  int fr_id2, int co_id2)
  boolean returnValue=true;
  int ii=1;
  while (DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[ii]!=null) {
    int cl_id1=DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[ii].ID.ClassNodeNumber;
    int jj=1;
    while (DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[jj]!=null) {
      int cl_id2=DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[jj].ID.ClassNodeNumber;
      returnValue=One_to_M_BiDirectionalAssociation_ClassLevel(fr_id1,co_id1,cl_id1,fr_id2,co_id2,cl_id2);
      if (!returnValue) {break; }
      jj=jj+1;
    }
    if (!returnValue) {break; }
    ii=ii+1;
  }
  return returnValue;
}
private static boolean One_to_M_BiDirectionalAssociation_FrameworkLevel(int fr_id1,int fr_id2) {
  boolean returnValue=true;
  int ii=1;

235
Appendix B1: A Prototype For CO-Toolkits' Design and Implementation Structures: An Overview

while (DesignAndImplementationStructure.Frameworks[fr_id1].Component[ii]!=null){
    int co_id1=DesignAndImplementationStructure.Frameworks[fr_id1].Component[ii].ID.ComponentNodeNumber;
    int jj=1;
    while (DesignAndImplementationStructure.Frameworks[fr_id2].Component[jj]!=null){
        int co_id2=DesignAndImplementationStructure.Frameworks[fr_id2].Component[jj].ID.ComponentNodeNumber;
        returnValue=One_to_M_BiDirectionalAssociation_ComponentLevel(fr_id1,co_id1,fr_id2,co_id2);
        if ((returnValue){ break;}
        jj=jj+1;
    }
    if ((returnValue){ break;}
    ii=ii+1;
}
return returnValue;

private static boolean One_to_M_BiDirectionalAssociation_SharedClassLevel(
    int shp_id1, int shc_id1,
    int shp_id2, int shc_id2){
    boolean returnValue=true;
    if ((DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1]!=null) &&
        (DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2]!=null)){
        int ii=1;
        int cnt=0;
        String className=DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].ClassName;
        while ((DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].PublicSection.Object[ii]!=null)){
            String objType=DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].PublicSection.ObjectType[ii];
            if (objType==className){
                cnt=cnt+1;
            }
            ii=ii+1;
        }
        ii=1;
        while ((DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].ProtectedSection.Object[ii]!=null)){
            String objType=DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].ProtectedSection.ObjectType[ii];
            if (objType==className){
                cnt=cnt+1;
            }
            ii=ii+1;
        }
        ii=1;
        while ((DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].PrivateSection.Object[ii]!=null)){
            String objType=DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].PrivateSection.ObjectType[ii];
            if (objType==className){
                cnt=cnt+1;
            }
            ii=ii+1;
        }
    if (cnt>1){
        ii=1;
        int cnt2=0;
        className=DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].ClassName;
        return false;
    }else{return true;}
while (DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].PublicSection.Object[ii]!=null){
    String objType=DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].PublicSection.Object[ii];
    if (objType==className){
        cnt2=cnt2+1;
        if (cnt2>1){returnValue=false; break; }
    }
    ii=ii+1;
}

ii=1;
while (DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].ProtectedSection.Object[ii]!=null){
    String objType=DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].ProtectedSection.Object[ii];
    if (objType==className){
        cnt2=cnt2+1;
        if (cnt2>1){returnValue=false; break; }
    }
    ii=ii+1;
}

ii=1;
while (DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].PrivateSection.Object[ii]!=null){
    String objType=DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].PrivateSection.Object[ii];
    if (objType==className){
        cnt2=cnt2+1;
        if (cnt2>1){returnValue=false; break; }
    }
    ii=ii+1;
}
else{returnValue=false; }
else{returnValue=false; }
return returnValue;
}

private static boolean One_to_M_BiDirectionalAssociation_SharedPoolLevel(int shp_id1, int shp_id2){
    boolean returnValue=true;
    int ii=1;
    while (DesignAndImplementationStructure.SharedPools[shp_id1].Class[ii]!=null){
        int shc_id1=DesignAndImplementationStructure.SharedPools[shp_id1].Class[ii].ID.ParentSharedPoolNodeNumber;
        int jj=1;
        while (DesignAndImplementationStructure.SharedPools[shp_id2].Class[jj]!=null){
            int shc_id2=DesignAndImplementationStructure.SharedPools[shp_id2].Class[jj].ID.ClassNodeNumber;
            returnValue=One_to_M_BiDirectionalAssociation_SharedClassLevel(shp_id1,shc_id1,shp_id2,shc_id2);
            if (!returnValue){break; }
            jj=jj+1;
        }
        if (!returnValue){break; }
        ii=ii+1;
    }
    return returnValue;
}
private static boolean One_to_M_UniDirectionalAssociation_ClassLevel(
    int fr_id1, int co_id1, int cl_id1,
    int fr_id2, int co_id2, int cl_id2){
    boolean returnValue=true;
    if((DesignAndImplementationStructure.Framesworks[fr_id1].Component[co_id1].Class[cl_id1]!=null) &&
       (DesignAndImplementationStructure.Framesworks[fr_id2].Component[co_id2].Class[cl_id2]!=null)){
        int ii=1;
        int cnt=0;
        String className=DesignAndImplementationStructure.Framesworks[fr_id2].Component[co_id2].Class[cl_id2].ID.ClassName;
        while  (DesignAndImplementationStructure.Framesworks[fr_id1].Component[co_id1].Class[cl_id1].PublicSection.Object[ii]!=null){
            String objType=DesignAndImplementationStructure.Framesworks[fr_id1].Component[co_id1].Class[cl_id1].PublicSection.ObjectType[ii];
            if (objType==className){
                cnt=cnt+1;
            }
            ii=ii+1;
        }
        ii=1;
        while  (DesignAndImplementationStructure.Framesworks[fr_id1].Component[co_id1].Class[cl_id1].ProtectedSection.Object[ii]!=null){
            String objType=DesignAndImplementationStructure.Framesworks[fr_id1].Component[co_id1].Class[cl_id1].ProtectedSection.ObjectType[ii];
            if (objType==className){
                cnt=cnt+1;
            }
            ii=ii+1;
        }
        ii=1;
        while  (DesignAndImplementationStructure.Framesworks[fr_id1].Component[co_id1].Class[cl_id1].PrivateSection.Object[ii]!=null){
            String objType=DesignAndImplementationStructure.Framesworks[fr_id1].Component[co_id1].Class[cl_id1].PrivateSection.ObjectType[ii];
            if (objType==className){
                cnt=cnt+1;
            }
            ii=ii+1;
        }
        if (cnt>1){
            ii=1;
            int cnt2=0;
            className=DesignAndImplementationStructure.Framesworks[fr_id1].Component[co_id1].Class[cl_id1].ID.ClassName;
            while  (DesignAndImplementationStructure.Framesworks[fr_id2].Component[co_id2].Class[cl_id2].PublicSection.Object[ii]!=null){
                String objType=DesignAndImplementationStructure.Framesworks[fr_id2].Component[co_id2].Class[cl_id2].PublicSection.ObjectType[ii];
                if (objType==className){
                    cnt2=cnt2+1;
                    if (cnt2>=1){returnValue=false; break; }
                }
                ii=ii+1;
            }
        }
        if (returnValue=true) {
            returnValue=false; break; }
    }
    return returnValue;
}
null[co_id2].Class[cl_id2].ProtectedSection.ObjectType[ii];
        if (objType==className) {
            cnt2=cnt2+1;
            if (cnt2>=1){returnValue=false; break;}
        }
        ii=ii+1;
    }
    ii=1;
    while (DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[cl_id2].PrivateSection.Object[ii]!=null) {
        String objType=DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[cl_id2].PrivateSection.ObjectType[ii];
        if (objType==className) {
            cnt2=cnt2+1;
            if (cnt2>=1){returnValue=false; break;}
        }
        ii=ii+1;
    }
    else {returnValue=false; }
    else {returnValue=false; }
    return returnValue;
}
private static boolean One_to_M_UniDirectionalAssociation_ComponentLevel(
    int fr_id1, int co_id1,
    int fr_id2, int co_id2){
    boolean returnValue=true;
    int i=1;
    while (DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[i]!=null) {
        int cl_id1=DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[i].ID.ClassNodeNumber;
        int j=1;
        while (DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[j]!=null) {
            int cl_id2=DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[j].ID.ClassNodeNumber;
            returnValue=One_to_M_UniDirectionalAssociation_ClassLevel(fr_id1,co_id1,cl_id1,fr_id2,co_id2,cl_id2);
            if (!returnValue) {break;}
            j=j+1;
        }
        if (!returnValue) {break;}
        ii=ii+1;
    }
    return returnValue;
}
private static boolean One_to_M_UniDirectionalAssociation_FrameworkLevel(int fr_id1,int fr_id2){
    boolean returnValue=true;
    int ii=1;
    while (DesignAndImplementationStructure.Frameworks[fr_id1].Component[ii]!=null) {
        int co_id1=DesignAndImplementationStructure.Frameworks[fr_id1].Component[ii].ID.GetComponentNodeNumber;
        int jj=1;
        while (DesignAndImplementationStructure.Frameworks[fr_id2].Component[jj]!=null) {
            int co_id2=DesignAndImplementationStructure.Frameworks[fr_id2].Component[jj].ID.GetComponentNodeNumber;
            returnValue=One_to_M_UniDirectionalAssociation_ComponentLevel(fr_id1,co_id1,fr_id2,co_id2);
            if (!returnValue) {break;}
        }
    }
    return returnValue;
}
private static boolean One_to_M_UniDirectionalAssociation_SharedClassLevel(
    int shp_id1, int shc_id1,
    int shp_id2, int shc_id2) {

    boolean returnValue = true;
    if ((DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1] != null) &&
        (DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2] != null)) {
        int ii = 1;
        int cnt = 0;
        String className = DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].ClassName;
        while (DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].ProtectedSection.Object[ii] != null) {
            String objectType = DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].ProtectedSection.ObjectType[ii];
            if (objectType == className) {
                cnt += 1;
            }
            ii += 1;
        }
        ii = 1;
        while (DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].ProtectedSection.Object[ii] != null) {
            String objectType = DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].ProtectedSection.ObjectType[ii];
            if (objectType == className) {
                cnt += 1;
            }
            ii += 1;
        }
        ii = 1;
        while (DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].PrivateSection.Object[ii] != null) {
            String objectType = DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].PrivateSection.ObjectType[ii];
            if (objectType == className) {
                cnt += 1;
            }
            ii += 1;
        }
        if (cnt > 1) {
            ii = 1;
            cnt = 2;
            className = DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].ClassName;
            while (DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].PublicSection.Object[ii] != null) {
                String objectType = DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].PublicSection.ObjectType[ii];
                if (objectType == className) {
                    cnt2 = cnt2 + 1;
                    if (cnt2 >= 1) { returnValue = false; break; }
                }
            }
        }
        ii += 1;
    }
}

240
Appendix B1: A Prototype For CO-Toolkits’ Design and Implementation Structures: An Overview

```java
}

} 
while ( DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].ProtectedSection.Object[ii]!=null) {
    String objectType = DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].ProtectedSection.Object[ii];
    if (objectType==className){
        cnt2=cnt2+1;
        if (cnt2>=1){returnValue=false; break; }
    }
    ii=ii+1;
}
ii=1;
while ( DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].PrivateSection.Object[ii]!=null) {
    String objectType = DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].PrivateSection.Object[ii];
    if (objectType==className){
        cnt2=cnt2+1;
        if (cnt2>=1){returnValue=false; break; }
    }
    ii=ii+1;
}
else {returnValue=false; }
else {returnValue=false; }
return returnValue;
}
private static boolean One_to_M_UniDirectionalAssociation_SharedPoolLevel(int shp_id1,int shp_id2){
    boolean returnValue=true;
    int ii=1;
    while ( DesignAndImplementationStructure.SharedPools[shp_id1].Class[ii]!=null){
        int shc_id1=DesignAndImplementationStructure.SharedPools[shp_id1].Class[ii].ID.PARENTSHAREDPOOLID;
        int jj=1;
        while ( DesignAndImplementationStructure.SharedPools[shp_id2].Class[jj]!=null){
            int shc_id2=DesignAndImplementationStructure.SharedPools[shp_id2].Class[jj].ID.CLASSNODENUMBER;
            returnValue=One_to_M_UniDirectionalAssociation_SharedClassLevel(shp_id1,shc_id1,shp_id2,shc_id2);
            if (!returnValue) {break;}
            jj=jj+1;
        }
    }
    if (!returnValue) {break;}
    ii=ii+1;
}
return returnValue;
}
private static boolean One_to_One_BiDirectionalAssociation_ClassLevel(
    int fr_id1, int co_id1, int cl_id1,
    int fr_id2, int co_id2, int cl_id2){
    boolean returnValue=true;
    if((DesignAndImplementationStructure.Frames[fr_id1].Component[co_id1].Class[cl_id1]!=null) &&
    (DesignAndImplementationStructure.Frames[fr_id2].Component[co_id2].Class[cl_id2]!=null)){
        int ii=1;
        int cnt=0;
        String className=DesignAndImplementationStructure.Frames[fr_id2].Component[co_id2].Class[cl_id2].ID.ClassName;
```
while (DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[cl_id1].PublicSection.Object[ii]!=null)
{
    String objType=DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[cl_id1].PublicSection.Object[ii];
    if (objType==className){
        cnt=cnt+1;
        if (cnt>1){returnValue=false; break; }
    }
    ii=ii+1;
}
ii=1;
while ((cnt<=1) && (DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[cl_id1].ProtectedSection.Object[ii]!=null)){
    String objType=DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[cl_id1].ProtectedSection.Object[ii];
    if (objType==className){
        cnt=cnt+1;
        if (cnt>1){returnValue=false; break; }
    }
    ii=ii+1;
}
ii=1;
while ((cnt<=1) && (DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[cl_id1].PrivateSection.Object[ii]!=null)){
    String objType=DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[cl_id1].PrivateSection.Object[ii];
    if (objType==className){
        cnt=cnt+1;
        if (cnt>1){returnValue=false; break; }
    }
    ii=ii+1;
}
if (cnt==1){
    ii=1;
    int cnt2=0;
    className=DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[cl_id1].ID.ClassName;
    while (DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[cl_id2].PublicSection.Object[ii]!=null){
        String objType=DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[cl_id2].PublicSection.Object[ii];
        if (objType==className){
            cnt2=cnt2+1;
            if (cnt2>1){returnValue=false; break; }
        }
        ii=ii+1;
    }
    ii=1;
    while ((cnt2<=1) && (DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[cl_id2].ProtectedSection.Object[ii]!=null)){
        String objType=DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[cl_id2].ProtectedSection.Object[ii];
        if (objType==className){
            cnt2=cnt2+1;
            if (cnt2>1){returnValue=false; break; }
        }
        ii=ii+1;
    }
}
ii=1;
while (cnt2<=1) & & (DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[cl_id2].PrivateSection.Object[ii]!=null)
    {
        String objType=DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[cl_id2].PrivateSection.ObjectType[ii];
        if (objType==className)
        {
            cnt2=cnt2+1;
            if (cnt2>1){returnValue=false; break; }
        }
        ii=ii+1;
    }
else {returnValue=false;}
else{returnValue=false;
return returnValue;
}

private static boolean One_to_One_BiDirectionalAssociation_ComponentLevel(
    int fr_id1, int co_id1,
    int fr_id2, int co_id2)
    {
        boolean returnValue=true;
        int ii=1;
        while (DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[ii]!=null){
            int cl_id1=DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[ii].ID.ClassNodeNumber;
            int jj=1;
            while (DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[jj]!=null){
                int cl_id2=DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[jj].ID.ClassNodeNumber;
                returnValue=One_to_One_BiDirectionalAssociation_ComponentLevel(fr_id1,co_id1,cl_id1,fr_id2,co_id2,cl_id2);
                if (!returnValue){break; }
                jj=jj+1;
            }
        }
        if (!returnValue){break; }
        ii=ii+1;
    }
    return returnValue;
}

private static boolean One_to_One_BiDirectionalAssociation_FrameworkLevel(int fr_id1,int fr_id2)
    {
        boolean returnValue=true;
        int ii=1;
        while (DesignAndImplementationStructure.Frameworks[fr_id1].Component[ii]!=null){
            int co_id1=DesignAndImplementationStructure.Frameworks[fr_id1].Component[ii].ID.ComponentNodeNumber;
            int jj=1;
            while (DesignAndImplementationStructure.Frameworks[fr_id2].Component[jj]!=null){
                int co_id2=DesignAndImplementationStructure.Frameworks[fr_id2].Component[jj].ID.ComponentNodeNumber;
                returnValue=One_to_One_BiDirectionalAssociation_ComponentLevel(fr_id1,co_id1,fr_id2,co_id2);
                if (!returnValue){break; }
                jj=jj+1;
            }
        }
        if (!returnValue){break; }
        ii=ii+1;
    }
return returnValue;
}

private static boolean One_to_One_BiDirectionalAssociation_SharedCClassLevel(
    int shp_id1, int shc_id1,
    int shp_id2, int shc_id2)
{
    boolean returnValue=true;
    if(((DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1]!=null) &&
        (DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2]!=null))){
        int ii=1;
        int cnt=0;
        String className=DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].ClassName;
        while (DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].PublicSection.Object[ii]!=null){
            String objType=DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].PublicSection.ObjectType[ii];
            if (objType==className){
                cnt=cnt+1;
                if (cnt>1){returnValue=false; break; }
            }
            ii=ii+1;
        }
        ii=1;
        while (DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].ProtectedSection.Object[ii]!=null){
            String objType=DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].ProtectedSection.ObjectType[ii];
            if (objType==className){
                cnt=cnt+1;
                if (cnt>1){returnValue=false; break; }
            }
            ii=ii+1;
        }
        ii=1;
        while (DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].PrivateSection.Object[ii]!=null){
            String objType=DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].PrivateSection.ObjectType[ii];
            if (objType==className){
                cnt=cnt+1;
                if (cnt>1){returnValue=false; break; }
            }
            ii=ii+1;
        }
        if (cnt==1){
            ii=1;
            int cnt2=0;
            className=DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].ClassName;
            while (DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].PublicSection.Object[ii]!=null){
                String objType=DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].PublicSection.ObjectType[ii];
                if (objType==className){
                    cnt2=cnt2+1;
                    if (cnt2>1){returnValue=false; break; }
                }
                ii=ii+1;
            }
            ii=1;
        }
    }
}
while (DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].ProtectedSection.Object[ii]!=null){
    String objType=DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].ProtectedSection.Object[ii];
    if (objType==className){
        cnt2=cnt2+1;
        if (cnt2>1){returnValue=false; break; }
    }
    ii=ii+1;
}
ii=1;
while (DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].PrivateSection.Object[ii]!=null){
    String objType=DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].PrivateSection.Object[ii];
    if (objType==className){
        cnt2=cnt2+1;
        if (cnt2>1){returnValue=false; break; }
    }
    ii=ii+1;
}
else{returnValue=false; }
}
else{returnValue=false; }
return returnValue;
}
private static boolean One_to_One_BiDirectionalAssociation_SharedPoolLevel(int shp_id1,int shp_id2){
    boolean returnValue=true;
    int ii=1;
    while (DesignAndImplementationStructure.SharedPools[shp_id1].Class[ii]!=null){
        int shc_id1=DesignAndImplementationStructure.SharedPools[shp_id1].Class[ii].ID.ParentSharedPoolNodeNumber;
        int jj=1;
        while(DesignAndImplementationStructure.SharedPools[shp_id2].Class[jj]!=null){
            int shc_id2=DesignAndImplementationStructure.SharedPools[shp_id2].Class[jj].ID.ClassNodeNumber;
            returnValue=One_to_One_BiDirectionalAssociation_SharedClassLevel(shp_id1,shc_id1,shp_id2,shc_id2);
            if (!returnValue){break; }
            jj=jj+1;
        }
        if (!returnValue){break; }
        ii=ii+1;
    }
    return returnValue;
}
private static boolean One_to_One_UniDirectionalAssociation_ClassLevel(
    int fr_id1, int co_id1, int cl_id1,
    int fr_id2, int co_id2, int cl_id2){
    boolean returnValue=true;
    if((DesignAndImplementationStructureFrameworks[fr_id1].Component[co_id1].Class[cl_id1]!=null) &
        (DesignAndImplementationStructureFrameworks[fr_id2].Component[co_id2].Class[cl_id2]!=null)){
        int ii=1;
        int cnt=0;
        String className=DesignAndImplementationStructureFrameworks[fr_id2].Component[co_id2].Class[cl_id2].ID.ClassName;
        while (DesignAndImplementationStructureFrameworks[fr_id1].Component[co_id1].Class[cl_id1].PublicSection.Object[ii]!=null)){

245
String objType=DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[cl_id1].PublicSection.ObjectType[ii];
if (objType==className){
cnt=cnt+1;
if (cnt>1){returnValue=false; break; }
}
ii=ii+1;
}
i=1;
while (DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[cl_id1].ProtectedSection.Object[ii]!=null){
String objType=DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[cl_id1].ProtectedSection.ObjectType[ii];
if (objType==className){
cnt=cnt+1;
if (cnt>1){returnValue=false; break; }
}
ii=ii+1;
}
i=1;
while (DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[cl_id1].PrivateSection.Object[ii]!=null){
String objType=DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[cl_id1].PrivateSection.ObjectType[ii];
if (objType==className){
cnt=cnt+1;
if (cnt>1){returnValue=false; break; }
}
ii=ii+1;
}
if (cnt==1){
i=1;
int cnt2=0;
className=DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[cl_id1].ID ClassName;
while (DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[cl_id2].PublicSection.Object[ii]!=null){
String objType=DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[cl_id2].PublicSection.ObjectType[ii];
if (objType==className){
cnt2=cnt2+1;
if (cnt2==1){returnValue=false; break; }
}
ii=ii+1;
}
}
}
while ((cnt2<=1) && (DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[cl_id2].ProtectedSection.Object[ii]!=null)){
String objType=DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[cl_id2].ProtectedSection.ObjectType[ii];
if (objType==className){
cnt2=cnt2+1;
if (cnt2==1){returnValue=false; break; }
}
ii=ii+1;
}
ii=1;
while ((cnt2<=1) && (DesignAndImplementationStructure.Frameworks[fr_id2].Comp-
nent[co_id2].Class[cl_id2].PrivateSection.Object[ii]!=null)) {
  String objType=DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[cl_id2].PrivateSection.Object[ii];
  if (objType==cl�Name)
    cnt2++;
    if (cnt2>=1){returnValue=false; break;}
  } ii=ii+1;
}
else{returnValue=false;}
}
else{returnValue=false; return returnValue;
}
private static boolean One_to_One_UniDirectionalAssociation_ComponentLevel(
  fr_id1, co_id1,
  fr_id2, co_id2){
  boolean returnValue=true;
  int ii=1;
  while (DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[ii]!=null) {
    int cl_id1=DesignAndImplementationStructure.Frameworks[fr_id1].Component[co_id1].Class[ii].ID.ClassNodeNumber;
    int jj=1;
    while (DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[jj]!=null) {
      int cl_id2=DesignAndImplementationStructure.Frameworks[fr_id2].Component[co_id2].Class[jj].ID.ClassNodeNumber;
      returnValue=One_to_One_UniDirectionalAssociation_ClassLevel(fr_id1,co_id1,fr_id2,co_id2,cl_id2);
      if (!returnValue){break;}
      jj=jj+1;
    }
    if (!returnValue){break;}
    ii=ii+1;
  }
  return returnValue;
}
private static boolean One_to_One_UniDirectionalAssociation_FrameworkLevel(int fr_id1,int fr_id2){
  boolean returnValue=true;
  int ii=1;
  while (DesignAndImplementationStructure.Frameworks[fr_id1].Component[ii]!=null) {
    int co_id1=DesignAndImplementationStructure.Frameworks[fr_id1].Component[ii].ID.ComponentNodeNumber;
    int jj=1;
    while (DesignAndImplementationStructure.Frameworks[fr_id2].Component[jj]!=null) {
      int co_id2=DesignAndImplementationStructure.Frameworks[fr_id2].Component[jj].ID.ComponentNodeNumber;
      returnValue=One_to_One_UniDirectionalAssociation_ComponentLevel(fr_id1,co_id1,fr_id2,co_id2);
      if (!returnValue){break;}
      jj=jj+1;
    }
    if (!returnValue){break;}
    ii=ii+1;
  }
  return returnValue;
}
private static boolean One_to_One UniDirectionalAssociation_FromSharedPoolsToFrameworks(int fr_id, int co_id, int cl_id, int shp_id, int shc_id) {
    boolean return_value = true;
    String classname = DesignAndImplementationStructure.Frameworks[fr_id].Component[co_id].Class[cl_id].ID.ClassName;
    int ii = 1;
    while (DesignAndImplementationStructure.SharedPools[shp_id].Class[shc_id].PublicSection.Object[ii] != null) {
        if (DesignAndImplementationStructure.SharedPools[shp_id].Class[shc_id].PublicSection.Object.Type[ii] == className) {
            returnValue = false;
            reportViolation("SharedClass(\"+shp_id+\", \"+shc_id+\") contains objects from Class(\"+fr_id+\", \"+co_id+\", \"+cl_id+\")");
            break;
        }
        ii = ii + 1;
    }
    return returnValue;
}

private static boolean One_to_One UniDirectionalAssociation_SharedClassLevel(
    int shp_id1, int shc_id1,
    int shp_id2, int shc_id2) {
    boolean return_value = true;
    if (DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1] != null) &&
        (DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2] != null)) {
        int ii = 1;
        int cnt = 0;
        String className = DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].ClassName;
        while (DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].PublicSection.Object[ii] != null) {
            String objType = DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].PublicSection.ObjectType[ii];
            if (objType == className) {
                cnt = cnt + 1;
                if (cnt > 1) {returnValue = false; break;}
            }
            ii = ii + 1;
        }
        ii = 1;
        while (DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].ProtectedSection.Object[ii] != null) {
            String objType = DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].ProtectedSection.ObjectType[ii];
            if (objType == className) {
                cnt = cnt + 1;
                if (cnt > 1) {returnValue = false; break;}
            }
            ii = ii + 1;
        }
        ii = 1;
        while (DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].PrivateSection.Object[ii] != null) {
            String objType = DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].PrivateSection.ObjectType[ii];
            if (objType == className) {
                cnt = cnt + 1;
                if (cnt > 1) {returnValue = false; break;}
            }
        }
    }
}
ii=ii+1;
}
if (cat==1)  
  
int cnt2=0;

className=DesignAndImplementationStructure.SharedPools[shp_id1].Class[shc_id1].ClassName;
while (DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].PublicSection.Object[ii]!=null)  
  
String objType=DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].PublicSection.ObjectType[ii];
    if (objType==className){
      cnt2=cnt2+1;
      if (cnt2>=1){returnValue=false; break; }
    }
  ii=ii+1;
}
ii=1;
while (DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].ProtectedSection.Object[ii]!=null)  
  
String objType=DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].ProtectedSection.ObjectType[ii];
    if (objType==className){
      cnt2=cnt2+1;
      if (cnt2>=1){returnValue=false; break; }
    }
  ii=ii+1;
}
ii=1;
while (DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].PrivateSection.Object[ii]!=null)  
  
String objType=DesignAndImplementationStructure.SharedPools[shp_id2].Class[shc_id2].PrivateSection.ObjectType[ii];
    if (objType==className){
      cnt2=cnt2+1;
      if (cnt2>=1){returnValue=false; break; }
    }
  ii=ii+1;
}
else{returnValue=false; }
}
else{returnValue=false; }
return returnValue;
}/*
* @return boolean
*/
// need to determine what parameters to pass
private static boolean One_to_One_UnidirectionalAssociation_SharedPoolLevel(int shp_id1,int shp_id2) {
    boolean returnValue=true;
    int ii=1;
    while (DesignAndImplementationStructure.SharedPools[shp_id1].Class[ii]!=null) {
      int shc_id1=DesignAndImplementationStructure.SharedPools[shp_id1].Class[ii].ID_ParentSharedPoolNodeNumber;
      int jj=1;
      while (DesignAndImplementationStructure.SharedPools[shp_id1].Class[ii]!=null) {  
        int shc_id2=DesignAndImplementationStructure.SharedPools[shp_id2].Class[jj].ID_ClassNodeNumber;
        returnValue=One_to_One_UnidirectionalAssociation_SharedClassLevel;
      }
vel(shp_id1,shc_id1,shp_id2,shc_id2);
    if (!returnValue) {break;}
    jj=jj+1;
    }
    if (!returnValue) {break;}
    ii=ii+1;
    }
    return returnValue;
}
/**
 * @return boolean
 */
protected static boolean predicate5_18_GoodDesignAndImplementation( ) { 
    reportViolation("Checking for Main predicate5_18: GoodDesignAndImplementation");
    boolean returnValue1=predicate5_19();
    boolean returnValue=returnValue1;
    reportViolation("Checking for Main predicate5_18: GoodDesignAndImplementation is complete. STATUS: "+returnValue);
    return (returnValue);
}
/**
 * @return boolean
 */
protected static boolean predicate5_19(){
    reportViolation("Checking for Predicate5_19: GoodUseOfRelationsAmongComponents");
    boolean returnValue1=predicate5_20();
    boolean returnValue2=predicate5_21();
    boolean returnValue3=predicate5_22();
    boolean returnValue=returnValue1&&returnValue2&&returnValue3;
    reportViolation("Checking for Predicate5_19 is complete. STATUS: "+returnValue);
    return returnValue;
}
/**
 * @return boolean
 */
protected static boolean predicate5_20(){
    reportViolation("Checking for predicate5_20: SafeInheritance");
    boolean returnValue1=predicate5_23();
    boolean returnValue2=predicate5_24();
    boolean returnValue3=predicate5_25();
    boolean returnValue4=predicate5_26();
    boolean returnValue=returnValue1&&returnValue2&&returnValue3&&returnValue4;
    reportViolation("Checking for predicate5_20 is complete. STATUS: "+returnValue);
    return (returnValue);
}
/**
 * @return boolean
 */
protected static boolean predicate5_21(){
    reportViolation("Checking for predicate5_21: GoodAssociation");
    boolean returnValue=true;
    if (Globals.EXAMPLE_NOM !=3) {
        boolean returnValue1=predicate5_27();
        boolean returnValue2=predicate5_28();
        returnValue=returnValue1&&returnValue2;
    }else{
        returnValue=true;
        // reportViolation("This Predicate has been disabled only for the ACE example 3");

250
```java
} reportViolation("Checking for predicate5_21 is complete. STATUS: "+returnValue); return returnValue;
}
/**
 * @return boolean
 */
protected static boolean predicate5_22(){
    reportViolation("Checking for predicate5_22: GoodUseOfOtherRelations");
    boolean returnValue=true;
    //This predicate is for future expansion purposes.
    reportViolation("Checking for predicate5_22 is complete. STATUS: "+returnValue);
    return (returnValue);
}
/**
 * @return boolean
 */
protected static boolean predicate5_23(){
    output="Checking for predicate 5_23: InheritanceMayNotExistAcrossFrameworks";
    boolean returnValue=true;
    for (int i=1; i<=DesignAndImplementationStructure.F;i++){
        for (int j=1;j<=DesignAndImplementationStructure.C[i][j][i];j++){
            for (int k=1; k<=DesignAndImplementationStructure.c[i][j]; k++){
                int ii=1;
                int fr_id, co_id, cl_id;
                ClassID inhClass=DesignAndImplementationStructure.Frames[i].Component[j].Class[k].PublicInheritedClasses[ii];
                while (inhClass!=null){
                    cl_id=DesignAndImplementationStructure.Frames[i].Component[j].Class[k].PublicInheritedClasses[ii].ClassNodeId;
                    if (fr_id !=i){
                        output=output+"nVIOLATION(Predicate5_23): Class("+ii++","+j++","+k++) inherits in public from Class("+fr_id++","+co_id++","+cl_id++) from another framework";
                        returnValue=false;
                    }
                }
                ii=ii+1;
                inhClass=DesignAndImplementationStructure.Frames[i].Component[j].Class[k].PublicInheritedClasses[ii];
            }
            ii=1;
            inhClass=DesignAndImplementationStructure.Frames[i].Component[j].Class[k].ProtectedInheritedClasses[ii];
            while (inhClass!=null){
                cl_id=DesignAndImplementationStructure.Frames[i].Component[j].Class[k].ProtectedInheritedClasses[ii].ClassNodeId;
                if (fr_id !=i){
                    output=output+"nVIOLATION(Predicate5_23): Class("+ii++","+j++","+k++) inherits in protected from Class("+fr_id++","+co_id++","+cl_id++) from another framework";
                    returnValue=false;
                }
            }
        }
    }
}
`
ii=ii+1;
   inhClass=DesignAndImplementationStructure.Frameworks[i].Component[j].Class[k].ProtectedInheritedClasses[ii];
   }
   ii=1;
   inhClass=DesignAndImplementationStructure.Frameworks[i].Component[j].Class[k].PrivateInheritedClasses[ii];
   while (inhClass!=null) {
      if (fr_id !=i) {
         output=output+"\nVIOLATION(Predicate5_23): Class(\"+i+\",\"+j+\",\"+k+\") inherits in private from Class(\"+fr_id+\",\"+co_id+\",\"+cl_id+\") from another framework";
         returnValue=false;
      }
      ii=ii+1;
      inhClass=DesignAndImplementationStructure.Frameworks[i].Component[j].Class[k].PrivateInheritedClasses[ii];
   }
   }
   output=\n"\nChecking for predicate5_23 is complete. STATUS: "+returnValue;
reportViolation(output);
return returnValue;
}
/**
 * @return boolean
 */
protected static boolean predicate5_24() {
   reportViolation("Checking for predicate5_24: SafeInheritance_AmongComponents_WithinTheSameFramework");
   boolean returnValue1=predicate5_30();
   boolean returnValue2=predicate5_31();
   boolean returnValue=returnValue1&&returnValue2;
   reportViolation("Checking for predicate5_24 is complete. STATUS: "+returnValue);
   return returnValue;
}
/**
 * @return boolean
 */
protected static boolean predicate5_25() {
   reportViolation("Checking for Predicate5_25: SafeInheritance_AmongClasses_WithinTheSameComponent");
   boolean returnValue1=predicate5_32();
   boolean returnValue2=predicate5_33();
   boolean returnValue=returnValue1&&returnValue2;
   reportViolation("Checking for Predicate5_25 is complete. STATUS: "+returnValue);
   return returnValue;
}
/**
 * @return boolean
 */
protected static boolean predicate5_26() {

reportViolation("Checking for predicate 5_26: SafeInheritance_AmongClasses_WithInTheSharedPools");
boolean returnValue1=predicate5_34();
boolean returnValue2=predicate5_35();
boolean returnValue=returnValue1&&!returnValue2;
reportViolation("Checking for predicate 5_26 is complete. STATUS: "+returnValue);
return returnValue;
}}
/**
 * @return boolean
 */

protected static boolean predicate5_27()
{
    reportViolation("Checking for Predicate 5_27: GoodAssociationOnTheComponentLevel");
    boolean returnValue1=predicate5_36();
    boolean returnValue2=predicate5_37();
    boolean returnValue=returnValue1&&!returnValue2;
    reportViolation("Checking for Predicate 5_27 is complete. STATUS: "+returnValue);
    return returnValue;
}
/**
 * @return boolean
 */

protected static boolean predicate5_28()
{
    reportViolation("Checking for Predicate 5_28: GoodAssociationOnInternalClassesLevel");
    boolean returnValue=predicate5_38();
    reportViolation("Checking for Predicate 5_28 is complete. STATUS: "+returnValue);
    return returnValue;
}
/**
 * @return boolean
 */

protected static boolean predicate5_300()
{
    boolean return_value=true;
    int i=0;
    int fr_id, co_id, cl_id;
    for (int i=1; i<=DesignAndImplementationStructure.F;i++)
    {
        for (int j=1; j<=DesignAndImplementationStructure.C[i]; j++)
        {
            for (int k=1; k<=DesignAndImplementationStructure.C[i][j]; k++)
            {
                int ii=1;
                ClassID inhClass=DesignAndImplementationStructure.Frames[i].Component[j].Class[k].PublicInheritedClasses[ii];
                while (inhClass!=null)
                {
                    cl_id=DesignAndImplementationStructure.Frames[i].Component[j].Class[k].PublicInheritedClasses[ii].ClassName;
                    if (fr_id!=i)
                    {
                        output=output+"inVIOLATION(Predicate5_30): Class("+i+","+j+","+k+"") inherits from Class("+fr_id+","+co_id+","+cl_id+"") from different framework";
                        returnValue=false;
                    }
                    else
                    {
                        if (co_id!=i)
                        {
                            {
cnt=cnt+1;
}
}
if (cnt>1){
    output=output+"\nVIOLATION(Predicate5_30): Component("+i+","+j+") inherits from more than one other components from the same framework";
    returnValue=false;
}
i=i+1;
inhClass=DesignAndImplementationStructure.Frameworks[i].Component[j].Class[k].PublicInheritedClasses[i];
}
i=1;
inhClass=DesignAndImplementationStructure.Frameworks[i].Component[j].Class[k].ProtectedInheritedClasses[i];
while (inhClass!=null){
    cl_id=DesignAndImplementationStructure.Frameworks[i].Component[j].Class[k].ProtectedInheritedClasses[i].ClassNodeNumber;
    if (fr_id==i){
        output=output+"\nVIOLATION(Predicate5_30): Class("+i+","+j+","+k+") inherits from Class("+fr_id+","+co_id+","+cl_id+") from different framework";
        returnValue=false;
    } else{
        if (co_id!=j){
            if (DesignAndImplementationStructure.Frameworks[i].Component[j].Class[k].ProtectedInheritedClasses[i].ParentComponent.ComponentNodeNumber != co_id){
                cnt=cnt+1;
            }
        }
    }
}
if (cnt>1){
    output=output+"\nVIOLATION(Predicate5_30): Component("+i+","+j+") inherits from more than one other components from the same framework";
    returnValue=false;
    break;
}
i=i+1;
inhClass=DesignAndImplementationStructure.Frameworks[i].Component[j].Class[k].ProtectedInheritedClasses[i];
}
i=1;
inhClass=DesignAndImplementationStructure.Frameworks[i].Component[j].Class[k].PrivateInheritedClasses[i];
while (inhClass!=null){
    cl_id=DesignAndImplementationStructure_Frameworks[i].Component[j].Class[k].PrivateInheritedClasses[i].ClassNodeNumber;
    if (fr_id==i){
        output=output+"\nVIOLATION(Predicate5_30): Class("+i+","+j+","+k+") inherits from Class("+fr_id+","+co_id+","+cl_id+") from different framework";
        returnValue=false;
    } else{
        if (co_id!=j){
            if (DesignAndImplementationStructure_Frameworks[i].Component[j].Class[k].PrivateInheritedClasses[i].ParentComponent.ComponentNodeNumber != co_id){
                cnt=cnt+1;
            }
        }
    }
}
if (cnt>1){
    output=output+"\nVIOLATION(Predicate5_30): Component("+i+","+j+") inherits from more than one other components from the same framework";
    returnValue=false;
    break;
}
}
Appendix B1: A Prototype For CO-Toolkits' Design and Implementation Structures: An Overview

```java
cl_id=DesignAndImplementationStructure.Frameworks[i].Component[j].Class[k].PrivateInheritedClasses[ii].ClassName;
if (fr_id!=i) {
    output=output+"\nVIOLATION(Predicate5_30): Class("+i+","+j+","+k") inherits from
Class("+fr_id+","+co_id+","+cl_id") from different framework."
    returnValue=false;
}
else {
    if (co_id!=j) {
            cnt=cnt+1;
        }
    }
    if (cnt>1) {
        output=output+"\nVIOLATION(Predicate5_30): Component("+i+","+j") inherits from more than
one other components from the same framework."
        returnValue=false;
        break;
    }
    ii=ii+1;
    inhClass=DesignAndImplementationStructure.Frameworks[i].Component[j].Class[k].PrivateInheritedClasses[ii];
}
}
}

output=output+"\nChecking for predicate5_30 is complete. STATUS: "+returnValue;
reportViolation(output);
return returnValue;

/**
 * @return boolean
 */
protected static boolean predicate5_31() {
    output="Checking for OnlyPublicInheritanceMayExistAcrossComponentsWithInTheSameFramework."
    boolean returnValue=true;
    int fr_id, co_id, cl_id;
    for (int i=1; i<=DesignAndImplementationStructure.Fr;i++) {
        for (int j=1; j<=DesignAndImplementationStructure.Cl[i]; j++) {
            for (int k=1; k<=DesignAndImplementationStructure.C[i][j]; k++) {
                int ii=1;
                ClassID inhClass=DesignAndImplementationStructure.Frameworks[i].Component[j].Class[k].ProtectedInheritedClasses[ii];
                while (inhClass!=null) {
                    cl_id=DesignAndImplementationStructure.Frameworks[i].Component[j].Class[k].ProtectedInheritedClasses[ii].ClassName;
                    if (fr_id!=i) {
                        output=output+"\nVIOLATION(Predicate5_31): Class("+i+","+j+","+k") inherits from
Class("+fr_id+","+co_id+","+cl_id") from different framework."
                    }
                }
            }
        }
    }
    return returnValue;
```
return Value=false;
}
else{
    if (co_id!=j){
        if (DesignAndImplementationStructure.Frameworks[i].Component[j].Class[k].ProtectedInheritedClasses[ii].ParentComponent.ComponentNodeNumber !=co_id){
            output=output+"nVIOLATION(Predicate5_31): Component("+i++","+j++") inherits in protected from Component("+i++","+co_id++");
            return Value=false;
        }
    }
    ii=ii+1;
    inhClass=DesignAndImplementationStructure.Frameworks[i].Component[j].Class[k].ProtectedInheritedClasses[ii];
}
ii=1;
inhClass=DesignAndImplementationStructure.Frameworks[i].Component[j].Class[k].PrivateInheritedClasses[ii];
    while (inhClass!=null){
        if (fr_id==i){
            output=output+"nVIOLATION(Predicate5_31): Class("+i++","+j++","+k++) inherits from Class("+fr_id++","+co_id++","+cl_id++") from different framework";
            return Value=false;
        }
    }
    if (co_id!=j){
            output=output+"nVIOLATION(Predicate5_31): Component("+i++","+j++") inherits in private from Component("+i++","+co_id++");
            return Value=false;
        }
    }
    ii=ii+1;
    inhClass=DesignAndImplementationStructure.Frameworks[i].Component[j].Class[k].PrivateInheritedClasses[ii];
}
}
output=output+"nChecking for predicate5_31 is complete. STATUS: "+return Value;
reportViolation(output);
return return Value;
}/**
* @return boolean

256
protected static boolean predicate5_32()
    reportViolation("unChecking NoMultipleInheritanceMayExistAmongClassesWithinTheSameComponent");
    boolean returnValue=true;
    int fr_id, co_id, cl_id;
    for (int i=1; i<=DesignAndImplementationStructure.F;i++){
        for (int j=1; j<=DesignAndImplementationStructure.C[i]; j++){
            int cnt=0;
            for (int k=1; k<=DesignAndImplementationStructure.c[i][j]; k++){
                int ii=1;
                ClassID inhClass=DesignAndImplementationStructure.Frameworks[i].Component[j].Class[k].PublicInheritedClasses[ii];
                    while (inhClass!=null){
                        cl_id=DesignAndImplementationStructure.Frameworks[i].Component[j].Class[k].PublicInheritedClasses[ii].ClassNodeNumber;
                        if (fr_id==i){
                            reportViolation("nVIOLATION(Predicate5_32): Class("+fr_id++","+co_id++","+cl_id++") inherits from Class("+fr_id++","+co_id++","+cl_id++") from different framework");
                            cnt=0;
                            returnValue=false;
                            break;
                        }
                        else{
                            if (co_id==j) cnt=cnt+1;
                            if (cnt>1){
                                reportViolation("nVIOLATION(Predicate5_32: Class("+i++","+j++","+k++") has multiple inheritance within the same component");
                                cnt=0;
                                returnValue=false;
                                break;
                            }
                        }
                        ii=ii+1;
                        inhClass=DesignAndImplementationStructure.Frameworks[i].Component[j].Class[k].PublicInheritedClasses[ii];
                    }
            ii=1;
            inhClass=DesignAndImplementationStructure.Frameworks[i].Component[j].Class[k].ProtectedInheritedClasses[ii];
            while (inhClass!=null){
                cl_id=DesignAndImplementationStructure.Frameworks[i].Component[j].Class[k].ProtectedInheritedClasses[ii].ClassNodeNumber;
                if (fr_id==i){
                    reportViolation("nVIOLATION(Predicate5_32): Class("+fr_id++","+co_id++","+cl_id++") inherits from Class("+fr_id++","+co_id++","+cl_id++") from different framework");
                    cnt=0;
                    returnValue=false;
                }
                else{
if (co_id==j) {
    cnt=cnt+1;
}
}
if (cnt>1) {
    reportViolation("nVIOLATION(Predicate5_32: Class("+i+","+j+","+k+)") has multiple inheritance from other classes within the same component");
    cnt=0;
    returnValue=false;
    break;
}
ii=ii+1;
inheritedClass=DesignAndImplementationStructure.Frameworks[i].Component[j].Class[k].ProtectedInheritedClasses[ii];
}
ii=1;
inheritedClass=DesignAndImplementationStructure.Frameworks[i].Component[j].Class[k].PrivateInheritedClasses[ii];
while (inheritedClass!=null) {
    cl_id=DesignAndImplementationStructure.Frameworks[i].Component[j].Class[k].PrivateInheritedClasses[ii].ClassName;
    if (fr_id==i) {
        reportViolation("nVIOLATION(Predicate5_32): Class("+i+","+j+","+k+") inherits from Class("+fr_id+","+co_id+","+cl_id+") from different framework");
        cnt=0;
        returnValue=false;
    }
    else {
        if (co_id==j) {
            cnt=cnt+1;
        }
    }
    if (cnt>1) {
        reportViolation("nVIOLATION(Predicate5_32: Class("+i+","+j+","+k+)") has multiple inheritance from other classes within the same component");
        cnt=0;
        returnValue=false;
        break;
    }
    ii=ii+1;
inheritedClass=DesignAndImplementationStructure.Frameworks[i].Component[j].Class[k].PrivateInheritedClasses[ii];
}
}
reportViolation("nChecking for predicate5_32 is complete. STATUS: ++returnValue");
return returnValue;
/**
 * @return boolean
 */
protected static boolean predicate5_33() {
    reportViolation("Checking for Predicate5_33: InheritanceMaybe_Public/Protected/or/Private_WithinTheSameComponent");
}

258
// This predicate is guaranteed by the Structure
boolean return_Value=true;
reportViolation("Checking for Predicate 5_33 is complete. STATUS: "+return_Value);
return return_Value;
}
/**
* @return boolean
*/
protected static boolean predicate5_34(){
output="Checking for predicate 5_34: InheritanceMayNotExistAcrossSharedPools;"
boolean return_Value=true;
for (int i=1; i<=DesignAndImplementationStructure.S;i++)
for (int j=1; j<=DesignAndImplementationStructure.SC[i];j++)
int ii=1;
SharedClassID inhClass=DesignAndImplementationStructure.SharedPools[i].Class[j].PublicInheritedSharedClasses[ii];
while (inhClass!=null){
    int shc_id=DesignAndImplementationStructure.SharedPools[i].Class[j].PublicInheritedSharedClasses[ij].ClassNameNumber;
    if (shp_id!=i){
        output=output+"nVIOLATION(Predicate 5_34): SharedClass("+i+","+j+"") inherits from SharedClass("+shp_id+","+shc_id+") from another shared pool";
        return_Value=false;
    }
    ii=ii+1;
    inhClass=DesignAndImplementationStructure.SharedPools[i].Class[j].PublicInheritedSharedClasses[ii];
}
ii=1;
inhClass=DesignAndImplementationStructure.SharedPools[i].Class[j].ProtectedInheritedSharedClasses[ii];
while (inhClass!=null){
    int shc_id=DesignAndImplementationStructure.SharedPools[i].Class[j].ProtectedInheritedSharedClasses[ii].ClassNameNumber;
    if (shp_id!=i){
        output=output+"nVIOLATION(Predicate 5_34): SharedClass("+i+","+j+"") inherits from SharedClass("+shp_id+","+shc_id+") from another shared pool";
        return_Value=false;
    }
    ii=ii+1;
    inhClass=DesignAndImplementationStructure.SharedPools[i].Class[j].ProtectedInheritedSharedClasses[ii];
}
ii=1;
inhClass=DesignAndImplementationStructure.SharedPools[i].Class[j].PrivateInheritedSharedClasses[ii];
while (inhClass!=null){
    int shc_id=DesignAndImplementationStructure.SharedPools[i].Class[j].PrivateInheritedSharedClasses[ii].ClassNameNumber;
    if (shp_id!=i){
        output=output+"nVIOLATION(Predicate 5_34): SharedClass("+i+","+j+"") inherits from SharedClass("+shp_id+","+shc_id+") from another shared pool";
    }
}}
Appendix B1: A Prototype For CO-Toolkits’ Design and Implementation Structures: An Overview

...
if ((DesignAndImplementationStructure.SharedPools[i].Class[j].ProtectedInheritedSharedClasses[j].ParentSharedPoolNodeNumber!=i) {
    output=output+"\nVIOLATION(predicate5_35): SharedClass("+i+","+j+") inherits from SharedClass("+shp_id+","+shc_id+") from another shared pool";
    returnValue=false;
} else {
    cnt=cnt+1;
    shc_id=DesignAndImplementationStructure.SharedPools[i].Class[j].ProtectedInheritedSharedClasses[j].ClassNodeNumber;
    if (cnt>1) {
        output=output+"\nVIOLATION(Predicate 5_35): SharedClass("+i+","+j+") has multiple inheritance from other shared classes within the same shared pool";
        break;
    }
}
ii=ii+1;
inhClass=DesignAndImplementationStructure.SharedPools[i].Class[j].ProtectedInheritedSharedClasses[ii];
}
ii=1;
cnt=0;
shc_id=0;
inhClass=DesignAndImplementationStructure.SharedPools[i].Class[j].PrivateInheritedSharedClasses[ii];
while (inhClass!=null) {
        output=output+"\nVIOLATION(predicate5_35): SharedClass("+i+","+j+") inherits from SharedClass("+shp_id+","+shc_id+") from another shared pool";
        returnValue=false;
    } else {
        cnt=cnt+1;
        if (cnt>1) {
            output=output+"\nVIOLATION(Predicate 5_35): SharedClass("+i+","+j+") has multiple inheritance from other shared classes within the same shared pool";
            break;
        }
    }
    ii=ii+1;
    inhClass=DesignAndImplementationStructure.SharedPools[i].Class[j].PrivateInheritedSharedClasses[ii];
}
output=output+"\nChecking for predicate 5_35 is complete. STATUS: "+returnValue;
reportViolation(output);
return returnValue;
}/**
 * @return boolean
 */
protected static boolean predicate5_36(){
    boolean returnValue=true;
    reportViolation("Checking for predicate5_36:

261
ComponentsMayHaveOnlyUniDirectionalAssociationWithClassesFromTheSharedPools);  
// This requires cross checking the classes of components and classes in the shared pools. Left for further expansion
reportViolation("Checking for predicate5_36 is complete. STATUS: "+returnValue);
return returnValue;
}
/**
 * @return boolean
 */
protected static boolean predicate5_37(){
reportViolation("Checking for predicate5_37: AssociationMayNotExistAcrossFrameworks");
boolean returnValue=true;
for (int i=1; i<=DesignAndImplementationStructure.F;i++){
   for (int ii=1; ii<=DesignAndImplementationStructure.F;ii++){
      if (ii==i){ii=ii+1;}
      if (ii>DesignAndImplementationStructure.F){break;}
      if (Associates_FrameworkLevel(i,ii)){
         returnValue=false;
         reportViolation("VIOLATION(Predicate5_37): Framework("+i+") associates with Framework("+ii+"));
      }
   }
   reportViolation("Checking for predicate5_37 is complete. STATUS: "+returnValue);
   return returnValue;
}
/**
 * @return boolean
 */
protected static boolean predicate5_38(){
reportViolation("Checking for predicate5_38: AssociationAndInheritanceMayNotExistBetweenTwoClassesAtTheSameTime");
boolean returnValue=true;
for (int i=1; i<=DesignAndImplementationStructure.F;i++){
   for (int j=1; j<=DesignAndImplementationStructure.C[i];j++){
      for (int k=1; k<=DesignAndImplementationStructure.c[i][j];k++){
         for (int ii=1; ii<=DesignAndImplementationStructure.F;ii++){
            if (ii==i){ii=ii+1;}
            if (ii>DesignAndImplementationStructure.F){break;}
            for (int jj=1; jj<=DesignAndImplementationStructure.C[ii];jj++){
               if (jj==j){jj=jj+1;}
               if (jj>DesignAndImplementationStructure.C[ii]){break;}
               for (int kk=1; kk<=DesignAndImplementationStructure.c[ii][jj];kk++){
                  if (kk==k){kk=kk+1;}
                  if (kk>DesignAndImplementationStructure.c[ii][jj]){break;}
                  if (Inherits_ClassLevel(i,j,k,ii,jj,kk) && Associates_ClassLevel(i,j,k,ii,jj,kk)){
                     returnValue=false;
                     break;
                  }
               }
            }
         }
      }
   }
   reportViolation("Checking for predicate5_38 is complete. STATUS: "+returnValue);
   return returnValue;
}
private static void reportOutput(String output){
262
System.out.println("n"+output+"n");
return;
}
private static void reportViolation(String output){
System.out.println("n"+output+"n");
return;
}

B1.2.2. Globals.java

This file contains the globals variables used for the implementation of this view’s structure. Next is the source code listing for this file:

```java
package V2_DesignAndImplementation_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#2 For CO_Toolkits Design&Implementation — Chapter 5 of My Ph.D. Thesis
 * File: Globals.java
 */
public class Globals {
    public static int EXAMPLE_NOM;
    public static int MAX_FR=16; // maximum number of frameworks -1
    public static int MAX_CO=46; // maximum number of components -1 per framework
    public static int MAX_CL=25; // maximum number of classes per component
    public static int MAX_PUB_INH_CL=6; // maximum number of publicly inherited classes by a one class
    public static int MAX_PRO_INH_CL=6; // maximum number of protectly inherited classes by a one class;
    it should be 0
    public static int MAX_PRV_INH_CL=6; // maximum number of privately inherited classes by a one class;
    it should be 0
    public static int MAX_PUB_OBJ=6; // maximum number of objects per public section of a class; it should be 0
    public static int MAX_PUB_MTD=6; // maximum number of methods per public section of a class
    public static int MAX_PUB_INH_MTD=6; // maximum number of methods inherited by a class;
    public static int MAX_PRO_OBJ=6; // maximum number of objects per public section of a class
    public static int MAX_PRO_MTD=6; // maximum number of methods per public section of a class
    public static int MAX_PRO_INH_MTD=6; // maximum number of methods inherited by a class; it should be 0
    public static int MAX_PRV_OBJ=6; // maximum number of objects per public section of a class
    public static int MAX_PRV_MTD=6; // maximum number of methods per public section of a class
    public static int MAX_PRV_INH_MTD=6; // maximum number of methods inherited by a class; it should be 0
    public static int MAX_IN_ARGS=4; // maximum number of input arguments per method
    public static int MAX_OUT_ARGS=4; // maximum number of output arguments per method
    public static int MAX_SP=10; // maximum number of shared pools
    public static int MAX_SH_CL=10; // maximum number of shared classes per shared pool
    public static int MAX_SH_ENT=10; // maximum number of shared entities per shared pool
    public static int MAX_SH_PUB_INH_CL=6; // maximum number of publicly inherited classes by a one class
    public static int MAX_SH_PRO_INH_CL=6; // maximum number of protectly inherited classes by a one
```

263
B1.2.3. MethodARGS.java

This file contains the code that used to represent a method arguments. Next is the source code listing for this file:

```java
package V2_DesignAndImplementation_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#2 For CO_Toolkits Design & Implementation — Chapter 5 of My Ph.D. Thesis
 * File: MethodARGS.java
 */
public class MethodARGS {
    String IN_ArgName[] = new String[Globals.MAX_IN_ARGS];
    String IN_ArgType[] = new String[Globals.MAX_IN_ARGS];
    String OUT_ArgName[] = new String[Globals.MAX_OUT_ARGS];
    String OUT_ArgType[] = new String[Globals.MAX_OUT_ARGS];
}
```

B1.2.4. MethodID.java

This file contains the code used to represent a method ID. Next is the source code listing for this file:

```java
package V2_DesignAndImplementation_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#2 For CO_Toolkits Design & Implementation — Chapter 5 of My Ph.D. Thesis
 */
```
* File: MethodID.java
*/
public class MethodID {
    ClassID ParentClass = new ClassID();
    int MethodNodeNumber;
}

B1.2.5. MethodNode.java

This file contains the code used to represent a method node in the structure. Next is the source code listing for this file:

package V2_DesignAndImplementation_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool #2 For CO_Toolkits Design&Implementation — Chapter 5 of My Ph.D. Thesis
 * File: MethodNode.java
 */
public class MethodNode {
    MethodID ID=new MethodID();
    MethodARGS ARGS=new MethodARGS();
    String ReturnedValue=-null;
    String MethodName=-null;public MethodNode() {super();}
}

B1.2.6. PublicSectionNode.java

This file contains the code used to represent a public section node in the structure. Next is the source code listing for this file:

package V2_DesignAndImplementation_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool #2 For CO_Toolkits Design&Implementation — Chapter 5 of My Ph.D. Thesis
 * File: PublicSectionNode.java
 */
public class PublicSectionNode {
    String Object[]=new String[Globals.MAX_PUB_OBJ];
    String ObjectType[]=new String[Globals.MAX_PUB_OBJ];
    MethodNode Method[]=new MethodNode[Globals.MAX_PUB_MT];public PublicSectionNode() {super();}
}
B1.2.7. ProtectedSectionNode.java

This file contains the code used to represent a protected section node in the structure.

Next is the source code listing for this file:

```java
package V2_DesignAndImplementation_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#2 For CO_Toolkits Design&Implementation — Chapter 5 of My Ph.D. Thesis
 * File: ProtectedSectionNode.java
 */
public class ProtectedSectionNode {
    String Object[] = new String[Globals.MAX_PRO_OBJ];
    String ObjectType[] = new String[Globals.MAX_PRO_OBJ];
    MethodNode Method[] = new MethodNode[Globals.MAX_PRO_MTD];
    public ProtectedSectionNode() {
        super();
    }
}
```

B1.2.8. PrivateSectionNode.java

This file contains the code that used to represent a private section node. Next is the source code listing for this file:

```java
package V2_DesignAndImplementation_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#2 For CO_Toolkits Design&Implementation — Chapter 5 of My Ph.D. Thesis
 * File: PrivateSectionNode.java
 */
public class PrivateSectionNode {
    String Object[] = new String[Globals.MAX_PRV_OBJ];
    String ObjectType[] = new String[Globals.MAX_PRV_OBJ];
    MethodNode Method[] = new MethodNode[Globals.MAX_PRV_MTD];
    public PrivateSectionNode() {
        super();
    }
}
```
B1.2.9. ClassID.java

This file contains the code that used to represent a class ID. Next is the source code listing for this file:

```java
package V2_DesignAndImplementation_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#2 For CO_Toolkits Design&Implementation — Chapter 5 of My Ph.D. Thesis
 * File: ClassID.java
 */
public class ClassID {
    ComponentID ParentComponent = new ComponentID();
    int ClassName;
}
```

B1.2.10. ClassNode.java

This file contains the code that used to represent a class node. Next is the source code listing for this file:

```java
package V2_DesignAndImplementation_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#2 For CO_Toolkits Design&Implementation — Chapter 5 of My Ph.D. Thesis
 * File: ClassNode.java
 */
public class ClassNode {
    ClassID ID = new ClassID();
    ClassID PublicInheritedClasses[] = new ClassID(Globals.MAX/pub_Inh_CL);
    ClassID ProtectedInheritedClasses[] = new ClassID(Globals.MAX.PRO_INH_CL);
    ClassID PrivateInheritedClasses[] = new ClassID(Globals.MAX.PRIV_INH_CL);
    PublicSectionNode PublicSection =new PublicSectionNode();
    ProtectedSectionNode ProtectedSection =new ProtectedSectionNode();
    PrivateSectionNode PrivateSection =new PrivateSectionNode();
    SharedClassID PublicInheritedSharedClasses[] = new SharedClassID(Globals.MAX_SH/PUB_INH_CL);
    SharedClassID ProtectedInheritedSharedClasses[] = new SharedClassID(Globals.MAX_SH.PRO_INH_CL);
    SharedClassID PrivateInheritedSharedClasses[] = new SharedClassID(Globals.MAX_SH_PRV_INH_CL);
    public ClassNode() {
        super();
    }
}
```
B1.2.11. ComponentID.java

This file contains the code that used to represent a component ID arguments. Next is the source code listing for this file:

```java
package V2_DesignAndImplementation_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#2 For CO_Toolkits Design&Implementation — Chapter 5 of My Ph.D. Thesis
 * File: ComponentID.java
 */
public class ComponentID {
    int ParentFrameworkNodeNumber;
    int ComponentNodeNumber;
}
```

B1.2.12. ComponentNode.java

This file contains the code used to represent a component node in the structure. Next is the source code listing for this file:

```java
package V2_DesignAndImplementation_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#2 For CO_Toolkits Design&Implementation — Chapter 5 of My Ph.D. Thesis
 * File: ComponentNode.java
 */
public class ComponentNode {  
    ComponentID ID = new ComponentID();
    String MainSection=null;
    ClassName Class[]=new ClassName[Globals.MAX_CL];
    public ComponentNode() {
        super();
    }
}
```

B1.2.13. FrameworkNode.java

This file contains the code that used to represent a framework node. Next is the source code listing for this file:

```java
package V2_DesignAndImplementation_R3;
/**
 */
```
**Appendix B1: A Prototype For CO-Toolkits’ Design and Implementation Structures: An Overview**

* Author: Mohammad Radaideh  
* Software Engineering Research Group  
* Department of Electrical & Computer Engineering  
* McMaster University  
* Hamilton, Ontario  
* January 1999  
* Proto-type Tool View#2 For CO_Toolkits Design&Implementation — Chapter 5 of My Ph.D. Thesis  
* File: FrameworkNode.java

```java
public class FrameworkNode {
    int FrameworkNodeNumber;
    ComponentNode Component[] = new ComponentNode[Globals.MAX_CO];
    public FrameworkNode() {
        super();
    }
}
```

**B1.2.14. SharedMethodID.java**

This file contains the code used to represent a a shared method ID. Next is the source code listing for this file:

```java
package V2_DesignAndImplementation_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#2 For CO_Toolkits Design&Implementation — Chapter 5 of My Ph.D. Thesis
 * File: SharedMethodID.java
 */
public class SharedMethodID {
    SharedClassID ParentSharedClass = new SharedClassID();
    int MethodNodeNumber;
}
```

**B1.2.15. SharedMethodNode.java**

This file contains the code used to represent a a shared method node. Next is the source code listing for this file:

```java
package V2_DesignAndImplementation_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#2 For CO_Toolkits Design&Implementation — Chapter 5 of My Ph.D. Thesis
 * File: SharedMethodNode.java
```
Appendix B1: A Prototype For CO-Toolkits’ Design and Implementation Structures: An Overview

*/
public class SharedMethodNode {
    SharedMethodID ID=new SharedMethodID();
    MethodARGS ARGS=new MethodARGS();
    String ReturnedValueType=null;
    String MethodName=null; public SharedMethodNode() {super();}
}

B1.2.16. SharedPublicSectionNode.java

This file contains the code used to represent a a shared class public section. Next is the
source code listing for this file:

package V2_DesignAndImplementation_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#2 For CO_Toolkits Design&Implementation — Chapter 5 of My Ph.D. Thesis
 * File: SharedPublicSectionNode.java
 */
public class SharedPublicSectionNode {
    String Object[]=new String[Globals.MAX_PUB_OBJ];
    String ObjectType[]= new String[Globals.MAX_PUB_OBJ];
    SharedMethodNode Method[]=new SharedMethodNode[Globals.MAX_PUB_MTD]; public SharedPub-
    licSectionNode() {super();}
}

B1.2.17. SharedProtectedSectionNode.java

This file contains the code used to represent a a shared class protected section. Next is the
source code listing for this file:

package V2_DesignAndImplementation_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#2 For CO_Toolkits Design&Implementation — Chapter 5 of My Ph.D. Thesis
 * File: SharedProtectedSectionNode.java
 */
public class SharedProtectedSectionNode {
    String Object[]=new String[Globals.MAX_SH_PRO_OBJ];
    String ObjectType[]= new String[Globals.MAX_SH_PRO_OBJ];
    SharedMethodNode Method[]=new SharedMethodNode[Globals.MAX_SH_PRO_MTD]; public SharedPro-
B1.2.18. SharedPrivateSectionNode.java

This file contains the code used to represent a shared class private section. Next is the source code listing for this file:

```java
package V2_DesignAndImplementation_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#2 For CO_Toolkits Design&Implementation — Chapter 5 of My Ph.D. Thesis
 * File: SharedPrivateSectionNode.java
 */
public class SharedPrivateSectionNode {
    String Object[] = new String[Globals.MAX_SH_PRV_OBJ];
    String ObjectType[] = new String[Globals.MAX_SH_PRV_OBJ];
    SharedMethodNode Method[] = new SharedMethodNode[Globals.MAX_SH_PRV_MTD];
    public SharedPrivateSectionNode() {super();}
}
```

B1.2.19. SharedClassID.java

This file contains the code used to represent a shared class ID in the structure. Next is the source code listing for this file:

```java
package V2_DesignAndImplementation_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#2 For CO_Toolkits Design&Implementation — Chapter 5 of My Ph.D. Thesis
 * File: SharedClassID.java
 */
public class SharedClassID {
    int ParentSharedPoolNodeNumber; // parent shared pool
    int ClassNodeNumber;
}
```

B1.2.20. SharedTreeNode.java

This file contains the code used to represent a shared class node in the structure. Next is the source code listing for this file:
package V2_DesignAndImplementation_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#2 For CO_Toolkits Design&Implementation — Chapter 5 of My Ph.D. Thesis
 * File: SharedClassNode.java
 */
public class SharedClassNode {
    SharedClassID ID = new SharedClassID();
    String ClassName;
    ClassID PublicInheritedClasses[] = new ClassID[Globals.MAX_PUB_INH_CL];
    ClassID ProtectedInheritedClasses[] = new ClassID[Globals.MAX_PRO_INH_CL];
    ClassID PrivateInheritedClasses[] = new ClassID[Globals.MAX_PRV_INH_CL];
    SharedClassID PublicInheritedSharedClasses[] = new SharedClassID[Globals.MAX_SH_PUB_INH_CL];
    SharedClassID ProtectedInheritedSharedClasses[] = new SharedClassID[Globals.MAX_SH_PRO_INH_CL];
    SharedClassID PrivateInheritedSharedClasses[] = new SharedClassID[Globals.MAX_SH_PRV_INH_CL];
    SharedPublicSectionNode PublicSection = new SharedPublicSectionNode();
    SharedProtectedSectionNode ProtectedSection = new SharedProtectedSectionNode();
    SharedPrivateSectionNode PrivateSection = new SharedPrivateSectionNode();
    public SharedClassNode() {
        super();
    }
}

B1.2.21. SharedPoolNode.java

This file contains the code that used to represent a shared pool node. Next is the source code listing for this file:

package V2_DesignAndImplementation_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#2 For CO_Toolkits Design&Implementation — Chapter 5 of My Ph.D. Thesis
 * File: SharedPoolNode.java
 */
public class SharedPoolNode {
    int SharedPoolNodeNumber;
    SharedClassNode Class[] = new SharedClassNode[Globals.MAX_SH_CL];
    public SharedPoolNode() {
        super();
    }
}
B1.2.22. DesignAndImplementationStructure.java

This file contains the implementation of the Design and Implementation Structure. Users should notice that the three examples are imbedded in this class. Next is the source code listing for this file:

```java
package V2_DesignAndImplementation_R3;
/**
 * Author: Mohammad Radaideh
 * Software Engineering Research Group
 * Department of Electrical & Computer Engineering
 * McMaster University
 * Hamilton, Ontario
 * January 1999
 * Proto-type Tool View#2 For CO-Toolkits Design&Implementation — Chapter 5 of My Ph.D. Thesis
 * File: DesignAndImplementationStructure.java
 */
public class DesignAndImplementationStructure {
    // F: number of frameworks in the system
    // C[][]: to hold number of components per framework
    // c[][]: to hold numbers of classes per component per framework
    // className[][]: to hold names of classes... only ACE example uses this
    // MP[][][][]: to hold numbers of methods in the public section per class per component per framework
    // MR[][][][]: to hold numbers of methods in the protected section per class per component per framework
    // MV[][][][]: to hold numbers of methods in the private section per class per component per framework
    // MinhP[][][][]: to hold numbers of inherited methods in the public section per class per component per framework
    // MinhR[][][][]: to hold numbers of inherited methods in the protected section per class per component per framework
    // MinhV[][][][]: to hold numbers of inherited methods in the private section per class per component per framework
    // OP[][]: to hold numbers of objects in the public section per class per component per framework
    // OMR[][][][]: to hold numbers of objects in the protected section per class per component per framework
    // OV[][]: to hold numbers of objects in the private section per class per component per framework
    // OinhP[][][][]: to hold numbers of inherited objectss in the public section per class per component per framework
    // OinhR[][][][]: to hold numbers of inherited objectss in the protected section per class per component per framework
    // OinhV[][][][]: to hold numbers of inherited objectss in the private section per class per component per framework
    public static int F; // NumberOfFrameworks;
    public static int C[] = new int[Globals.MAX_FR];
    // to hold number of components per framework
    public static int c[][] = new int[Globals.MAX_FR][Globals.MAX_CO][Globals.MAX_CL];
    public static String className[][] = new String[Globals.MAX_FR][Globals.MAX_CO][Globals.MAX_CL];
    public static int MP[][][][] = new int[Globals.MAX_FR][Globals.MAX_CO][Globals.MAX_CL];
    public static int MR[][][][] = new int[Globals.MAX_FR][Globals.MAX_CO][Globals.MAX_CL];
    public static int MV[][][][] = new int[Globals.MAX_FR][Globals.MAX_CO][Globals.MAX_CL];
    public static int MinhP[][][][] = new int[Globals.MAX_FR][Globals.MAX_CO][Globals.MAX_CL];
    public static int MinhR[][][][] = new int[Globals.MAX_FR][Globals.MAX_CO][Globals.MAX_CL];
    public static int MinhV[][][][] = new int[Globals.MAX_FR][Globals.MAX_CO][Globals.MAX_CL];
    public static int OP[][] = new int[Globals.MAX_FR][Globals.MAX_CO][Globals.MAX_CL];
    public static int OR[][][] = new int[Globals.MAX_FR][Globals.MAX_CO][Globals.MAX_CL];
    public static int OinhP[][][][] = new int[Globals.MAX_FR][Globals.MAX_CO][Globals.MAX_CL];
    public static int OinhR[][][][] = new int[Globals.MAX_FR][Globals.MAX_CO][Globals.MAX_CL];
    public static int OinhV[][][][] = new int[Globals.MAX_FR][Globals.MAX_CO][Globals.MAX_CL];
```
public static int OV[][][] = new int[Globals.MAX_FR][Globals.MAX_CO][Globals.MAX_CL];
// public static int OinhP[][][] = new int[Globals.MAX_FR][Globals.MAX_CO][Globals.MAX_CL];
// public static int OinhR[][][] = new int[Globals.MAX_FR][Globals.MAX_CO][Globals.MAX_CL];
// public static int OinhV[][][] = new int[Globals.MAX_FR][Globals.MAX_CO][Globals.MAX_CL];
public static int S; // NumberOfSharedPools;
public static int SC[] = new int[Globals.MAX_SP];
public static int SMnP[][] = new int[Globals.MAX_SP][Globals.MAX_SH_CLIP];
public static int SMnR[][] = new int[Globals.MAX_SP][Globals.MAX_SH_CLIP];
public static int SMnV[][] = new int[Globals.MAX_SP][Globals.MAX_SH_CLIP];
public static int SMinhP[][] = new int[Globals.MAX_SP][Globals.MAX_SH_CLIP];
public static int SMinhR[][] = new int[Globals.MAX_SP][Globals.MAX_SH_CLIP];
public static int SMinhRF[][] = new int[Globals.MAX_SP][Globals.MAX_SH_CLIP];
public static int SMinhVF[][] = new int[Globals.MAX_SP][Globals.MAX_SH_CLIP];

public static int SOP[][][] = new int[Globals.MAX_SP][Globals.MAX_SH_CLIP];
public static int SOR[][][] = new int[Globals.MAX_SP][Globals.MAX_SH_CLIP];

public static int SOV[][][] = new int[Globals.MAX_SP][Globals.MAX_SH_CLIP];
// public static int SOinhP[][][] = new int[Globals.MAX_SP][Globals.MAX_SH_CLIP];
// public static int SOinhR[][][] = new int[Globals.MAX_SP][Globals.MAX_SH_CLIP];
// public static int SOinhV[][][] = new int[Globals.MAX_SP][Globals.MAX_SH_CLIP];

private static String output;
public static FrameworkNode[] Frameworks = new FrameworkNode[Globals.MAX_FR];
public static SharedPoolNode[] SharedPools = new SharedPoolNode[Globals.MAX_SP];

protected static void display() {
    reportStructure("Design and Implementation Structure: As Extracted From The Current Example");
    displayFrameworks();
    displaySharedPools();
    reportStructure("=" * 80 + "=");
    return;
}

protected static void displayFrameworks() { output = "\n";
    for (int i = 1; i <= F; i++)
        reportStructure("\n Framework(" + i + ")");
    for (int j = 1; j <= C[i]; j++)
        reportStructure("\n\n Component(" + i + ");
    for (int k = 0; k <= C[i][j]; k++)
        String ClassName = Frameworks[i].Component[j].Class[k].ID.ClassName;
        reportStructure("\n\n Class(" + i + ");" + ClassName);

    if (MinhP[i][j][k] > 0)
        reportStructure("\n\n\nPublic Inheritance From Classes:");
    for (int kk = 1; kk <= MinhP[i][j][k]; kk++)
        int cln = Frameworks[i].Component[j].Class[k].PublicInheritedClasses[kk].ClassName;
        int con = Frameworks[i].Component[j].Class[k].PublicInheritedClasses[kk].ParentComponent;
        reportStructure("\n\n Class(" + fmn + ");" + con + ");" + Frameworks[fmn].Component[con].Class[cln].ID.ClassName);
}

if (MinhPS[i][j][k] > 0) {
for(int kk=1;kk<=MinhPS[i][j][k];kk++)
    int cln = Frameworks[i].Component[j].Class[k].PublicInheritedSharedClasses[kk].ClassName;
    int con = Frameworks[i].Component[j].Class[k].PublicInheritedSharedClasses[kk].ParentSharedPoolNodeNumber;
    reportStructure("\"SharedClass("+con+","+cln+\")":"+SharedPools[con].Class[cln].ClassName);
} }

// protected inheritance
if (MinhR[i][j][k]>0) {
    reportStructure("\n\n\nProtected Inheritance From Classes:\n");
    for (int kk=1;kk<=MinhR[i][j][k];kk++)
        int cln = Frameworks[i].Component[j].Class[k].ProtectedInheritedClasses[kk].ClassName;
        int con = Frameworks[i].Component[j].Class[k].ProtectedInheritedClasses[kk].ParentComponentNodeId;
    int fnr = Frameworks[i].Component[j].Class[k].ProtectedInheritedClasses[kk].ParentComponentNodeNumber;
    reportStructure("\tClass("+fnr+","+con+","+cln+\")":"+Frameworks[fnr].Component[con].Class[cln].ClassName);
} }

if(MinhRS[i][j][k]>0) {
    for (int kk=1; kk<=MinhRS[i][j][k]; kk++)
        int cln = Frameworks[i].Component[j].Class[k].ProtectedInheritedSharedClasses[kk].ClassName;
        int con = Frameworks[i].Component[j].Class[k].ProtectedInheritedSharedClasses[kk].ParentSharedPoolNodeNumber;
        reportStructure("\tSharedClass("+con+","+cln+\")":"+SharedPools[con].Class[cln].ClassName);
} }

// private inheritance
if (MinhV[i][j][k]>0) {
    reportStructure("\n\n\nPrivate Inheritance From Classes:\n");
    for (int kk=1; kk<=MinhV[i][j][k]; kk++)
        int cln = Frameworks[i].Component[j].Class[k].PrivateInheritedClasses[kk].ClassName;
        int con = Frameworks[i].Component[j].Class[k].PrivateInheritedClasses[kk].ParentComponentNodeId;
    int fnr = Frameworks[i].Component[j].Class[k].PrivateInheritedClasses[kk].ParentComponentNodeNumber;
    reportStructure("\tClass("+fnr+","+con+","+cln+\")":"+Frameworks[fnr].Component[con].Class[cln].ClassName);
} }

if(MinhVS[i][j][k]>0) {
    for (int kk=1; kk<=MinhVS[i][j][k]; kk++)
        int cln = Frameworks[i].Component[j].Class[k].PrivateInheritedSharedClasses[kk].ClassName;
        int con = Frameworks[i].Component[j].Class[k].PrivateInheritedSharedClasses[kk].ParentSharedPoolNodeNumber;
        reportStructure("\tSharedClass("+con+","+cln+\")":"+SharedPools[con].Class[cln].ClassName);
} }

//public section
reportStructure("\n\n\nPublic Section Methods & Objects\n");
for (int l=1; l<=MP[i][j][k]; l++)
    String MethodName=Frameworks[i].Component[j].Class[k].PublicSection.Method[l].MethodName;
    reportStructure("\n\n\nMethodName: "+MethodName);
    //ReturnedValueType
Appendix B1: A Prototype For CO-Toolkits’ Design and Implementation Structures: An Overview

```java
String ReturnedValueType=Frameworks[i].Component[j].Class[k].PublicSection.Method[l].ReturnedValueType;
reportStructure("\n\n\n\t\n\t\tReturnedValueType:\n\t"+ReturnedValueType);
//IN_ARGS
reportStructure("\n\n\n\t\n\t\tIN_ArgNames(Types): ");
int ll=1;
String IN_ARG=Frameworks[i].Component[j].Class[k].PublicSection.Method[l].ARGS.IN_ArgName[l1];
String IN_ARGType=Frameworks[i].Component[j].Class[k].PublicSection.Method[l].ARGS.IN_ArgType[l1];
while (IN_ARG!=null){
    reportStructure("\n\n\n\tIN_ARG="+IN_ARG+"\n\t"+IN_ARGType+"");
    ll=ll+1;
    IN_ARG=Frameworks[i].Component[j].Class[k].PublicSection.Method[l].ARGS.IN_ArgName[l1];
    IN_ARGType=Frameworks[i].Component[j].Class[k].PublicSection.Method[l].ARGS.IN_ArgType[l1];
}
//OUT_ARGS
reportStructure("\n\n\n\t\n\t\tOUT_ArgNames(Types): ");
ll=1;
String OUT_ARG=Frameworks[i].Component[j].Class[k].PublicSection.Method[l].ARGS.OUT_ArgName[l1];
String OUT_ARGType=Frameworks[i].Component[j].Class[k].PublicSection.Method[l].ARGS.OUT_ArgType[l1];
while (OUT_ARG!=null){
    reportStructure("\n\n\n\tOUT_ARG="+OUT_ARG+"\n\t"+OUT_ARGType+"");
    ll=ll+1;
    OUT_ARG=Frameworks[i].Component[j].Class[k].PublicSection.Method[l].ARGS.OUT_ArgName[l1];
    OUT_ARGType=Frameworks[i].Component[j].Class[k].PublicSection.Method[l].ARGS.OUT_ArgType[l1];
}
//objects of in public section
output=output+"\n\n\n\tObjects: ";
ll=1;
String ObjectName=Frameworks[i].Component[j].Class[k].PublicSection.Object[l1];
String ObjectType=Frameworks[i].Component[j].Class[k].PublicSection.ObjectType[l1];
while (ObjectName!=null){
    output=output+"\n\n\t"+ObjectName+"\n\t"+ObjectType+"\t";
    ll=ll+1;
    ObjectName=Frameworks[i].Component[j].Class[k].PublicSection.Object[l1];
    ObjectType=Frameworks[i].Component[j].Class[k].PublicSection.ObjectType[l1];
}
//protected section
reportStructure("\n\n\n\tProtected Section Methods & Objects");
for (int ll=1; l<=MR[i][j][k]; ll++){
    String MethodName=Frameworks[i].Component[j].Class[k].ProtectedSection.Method[l].MethodName;
    reportStructure("\n\n\n\t\n\t\tMethodName: "+MethodName);
    //ReturnedValueType
    String ReturnedValueType=Frameworks[i].Component[j].Class[k].ProtectedSection.Method[l].ReturnedValueType;
    reportStructure("\n\n\n\t\n\t\tReturnedValueType:\n\t"+ReturnedValueType);
    //IN_ARGS
    reportStructure("\n\n\n\t\n\t\tIN_ArgNames(Types): ");
    ll=1;
```
String IN_ARG=Frameworks[i].Component[j].Class[k].ProtectedSection.Method[l].ARGS.IN_ArgName[ll];
String IN_ARGType=Frameworks[i].Component[j].Class[k].ProtectedSection.Method[l].ARGS.IN_ArgType[ll];
while (IN_ARG!=null){
    reportStructure("\t"+IN_ARG++"("+IN_ARGType++")");
    ll=ll+1;
    IN_ARG=Frameworks[i].Component[j].Class[k].ProtectedSection.Method[l].ARGS.IN_ArgName[ll];
    IN_ARGType=Frameworks[i].Component[j].Class[k].ProtectedSection.Method[l].ARGS.IN_ArgType[ll];
}
//OUT_ARGS
reportStructure("\n\\n\\n\\nOUT_ArgNames(Types): ");
ll=1;
String OUT_ARG=Frameworks[i].Component[j].Class[k].ProtectedSection.Method[l].ARGS.OUT_ArgName[ll];
String OUT_ARGType=Frameworks[i].Component[j].Class[k].ProtectedSection.Method[l].ARGS.OUT_ArgType[ll];
while (OUT_ARG!=null){
    reportStructure("\t"+OUT_ARG++"("+OUT_ARGType++")");
    ll=ll+1;
    OUT_ARG=Frameworks[i].Component[j].Class[k].ProtectedSection.Method[l].ARGS.OUT_ArgName[ll];
    OUT_ARGType=Frameworks[i].Component[j].Class[k].ProtectedSection.Method[l].ARGS.OUT_ArgType[ll];
}
//objects of in protected section
reportStructure("\n\\n\\n\\nObjects: ");
ll=1;
ObjectName=Frameworks[i].Component[j].Class[k].ProtectedSection.Object[ll];
ObjectType=Frameworks[i].Component[j].Class[k].ProtectedSection.ObjectType[ll];
while (ObjectName!=null){
    reportStructure("\t"+ObjectName++"("+ObjectType++")");
    ll=ll+1;
    ObjectName=Frameworks[i].Component[j].Class[k].ProtectedSection.Object[ll];
    ObjectType=Frameworks[i].Component[j].Class[k].ProtectedSection.ObjectType[ll];
}
//private section
reportStructure("\n\\n\\nPrivate Section Methods & Objects: ");
for (int l=1; l<=MV[i][j][k]; l++){
    String MethodName=Frameworks[i].Component[j].Class[k].PrivateSection.Method[l].MethodName;
    reportStructure("\n\\nMethodName: "+MethodName);
    //ReturnedValueType
    String ReturnedValueTypeName=Frameworks[i].Component[j].Class[k].PrivateSection.Method[l].ReturnedValueType;
    reportStructure("\n\\nReturnedValueType: "+ReturnedValueType);
    //IN_ARGS
    reportStructure("\n\\nIN_ArgNames(Types): ");
    ll=1;
    String IN_ARG=Frameworks[i].Component[j].Class[k].PrivateSection.Method[l].ARGS.IN_ArgName[ll];
    String IN_ARGType=Frameworks[i].Component[j].Class[k].PrivateSection.Method[l].ARGS.IN_ArgType[ll];
    while (IN_ARG!=null){
        reportStructure("\t"+IN_ARG++"("+IN_ARGType++")");
        ll=ll+1;
    }
IN_ARG = Frameworks[i].Component[j].Class[k].PrivateSection.Method[l].ARGS.IN_Arg_Name[l];
IN_ARG_Type = Frameworks[i].Component[j].Class[k].PrivateSection.Method[l].ARGS.IN_Arg_Type[l];
}

//OUT_ARGS
reportStructure("\n\t\t\tOUT_ArgNames(Types): ");
ll = 1;
String OUT_ARG = Frameworks[i].Component[j].Class[k].PrivateSection.Method[l].ARGS.OUT_Arg_Name[l];
String OUT_ARG_Type = Frameworks[i].Component[j].Class[k].PrivateSection.Method[l].ARGS.OUT_Arg_Type[l];
while (OUT_ARG != null){
    reportStructure("\t" + OUT_ARG + "(" + OUT_ARG_Type + ")");
    ll = ll + 1;
    OUT_ARG = Frameworks[i].Component[j].Class[k].PrivateSection.Method[l].ARGS.OUT_Arg_Name[l];
    OUT_ARG_Type = Frameworks[i].Component[j].Class[k].PrivateSection.Method[l].ARGS.OUT_Arg_Type[l];
}

//objects of in private section
reportStructure("\n\t\tObjects: ");
ll = 1;
ObjectName = Frameworks[i].Component[j].Class[k].PrivateSection.Object[l];
ObjectType = Frameworks[i].Component[j].Class[k].PrivateSection.ObjectType[l];
while (ObjectName != null){
    reportStructure("\t" + ObjectName + "(" + ObjectType + ")");
    ll = ll + 1;
    ObjectName = Frameworks[i].Component[j].Class[k].PrivateSection.Object[l];
    ObjectType = Frameworks[i].Component[j].Class[k].PrivateSection.ObjectType[l];
}

reportStructure("\n\n");

protected static void displaySharedPools() {
    output = "\n";
    for (int i = 1; i <= S[i]; i++) {
        output = output + "\n\nSharedPool(" + i + ")\n";
    }
    for (int j = 1; j <= SC[j]; j++) {
        String ClassName = SharedPools[i].Class[j].ClassName;
        output = output + "\nClass(" + i + "," + j + "):" + ClassName + "\n";
        // public inheritance
        if (SMinhFI[i][j] > 0) {
            output = output + "\nPublic Inheritance From Classes:\t";
            for (int k = 1; k <= SMinhPFI[i][j][k]; k++) {
                int cln = SharedPools[i].Class[j].PublicInheritedSharedClasses[k].ClassName;
                int con = SharedPools[i].Class[j].PublicInheritedSharedClasses[k].ParentSharedPoolNodeNumber;
                output = output + "\nSharedClass(" + con + "," + cln + ")\n";
            }
        }
        output = output + "\n\n";
    }
    output = output + "\n";
    if (SMinhPF[i][j] > 0) {
        for (int k = 1; k <= SMinhPF[i][j][k]; k++) {
            int cln = SharedPools[i].Class[j].PublicInheritedClasses[k].ClassName;
            int con = SharedPools[i].Class[j].PublicInheritedClasses[k].ParentComponent.ComponentNodeNumber;
        }
    }
}
int fn= SharedPools[i].Class[j].PublicInheritedClasses[k].ParentComponent.ParentFrameworkNodeNumber;
    output=output+"\nClass("+frn+","+con+","+cln+"j";}
    }
// protected inheritance
if(SMinhR[i][j][k]>0){
    output=output+"\nProtected Inheritance From Classes:\n";
    for(int k=1;k<=SMinhR[i][j][k];k++){
    int cln= SharedPools[i].Class[j].ProtectedInheritedSharedClasses[k].ClassNameNumber;
    int con= SharedPools[i].Class[j].ProtectedInheritedSharedClasses[k].ParentSharedPoolNodeNumber;
    output=output+"\nSharedClass("+con+","+cln+"j";}
    }

if(SMinhRF[i][j][k]>0){
    for(int k=1;k<=SMinhRF[i][j][k];k++){
    int cln= SharedPools[i].Class[j].ProtectedInheritedClasses[k].ClassNameNumber;
    int con= SharedPools[i].Class[j].ProtectedInheritedClasses[k].ParentComponent.ComponentNodeNumber;
    int fn= SharedPools[i].Class[j].ProtectedInheritedClasses[k].ParentComponent.ParentFrameworkNodeNumber;
    output=output+"\nClass("+frn+","+con+","+cln+"j";}
    }
    output=output+"\n\n";
// private inheritance
if(SMinhV[i][j][k]>0){
    output=output+"\nPrivate Inheritance From Classes:\n";
    for(int k=1;k<=SMinhV[i][j][k];k++){
    int cln= SharedPools[i].Class[j].PrivateInheritedSharedClasses[k].ClassNameNumber;
    int con= SharedPools[i].Class[j].PrivateInheritedSharedClasses[k].ParentSharedPoolNodeNumber;
    output=output+"\nSharedClass("+con+","+cln+"j";}
    }

if(SMinhVF[i][j][k]>0){
    for(int k=1;k<=SMinhVF[i][j][k];k++){
    int cln= SharedPools[i].Class[j].PrivateInheritedClasses[k].ClassNameNumber;
    int con= SharedPools[i].Class[j].PrivateInheritedClasses[k].ParentComponent.ComponentNodeNumber;
    output=output+"\nClass("+frn+","+con+","+cln+"j";}
    }
    output=output+"\n\n";
// public section
output=output+"\nPublic Section Methods & Objects\n";
for(int l=1;l<=SMP[i][j][k];l++){
    String MethodName=SharedPools[i].Class[j].PublicSection.Method[l].MethodName;
    output=output+"\nMethod Name:"+MethodName+"\n";
//ReturnValueType
String ReturnedValueType=SharedPools[i].Class[j].PublicSection.Method[l].ReturnedValueType;
output=output+"\nReturnedValueType:"+ReturnedValueType+"\n";
//IN_ARGS
output=output+"\nIN_ArgNames(Types): ";
int ll=1;
String IN_ARG=SharePools[i].Class[j].PublicSection.Method[l].ARGS.IN_ArgName[l];
String IN_ARGType=SharePools[i].Class[j].PublicSection.Method[l].ARGS.IN_ArgType[l];
while (IN_ARG!=null) {
    output=output+"\t"+IN_ARG+"("+IN_ARGType+");
    l=l+1;
    IN_ARG=SharePools[i].Class[j].PublicSection.Method[l].ARGS.IN_ArgName[l];
    IN_ARGType=SharePools[i].Class[j].PublicSection.Method[l].ARGS.IN_ArgType[l];
}

//OUT_ARGS
output=output+"\n\n\tOUT_ArgNames(Types): ";

String OUT_ARG=SharePools[i].Class[j].PublicSection.Method[l].ARGS.OUT_ArgName[l];
String OUT_ARGType=SharePools[i].Class[j].PublicSection.Method[l].ARGS.OUT_ArgType[l];
while (OUT_ARG!=null) {
    output=output+"\t"+OUT_ARG+"("+OUT_ARGType+");
    l=l+1;
    OUT_ARG=SharePools[i].Class[j].PublicSection.Method[l].ARGS.OUT_ArgName[l];
    OUT_ARGType=SharePools[i].Class[j].PublicSection.Method[l].ARGS.OUT_ArgType[l];
}

output=output+"\n";

//objects of in public section
output=output+"\n\n\tObjects: ";
int l=1;
String ObjectName=SharePools[i].Class[j].PublicSection.Object[l];
String ObjectType=SharePools[i].Class[j].PublicSection.ObjectType[l];
while (ObjectName!=null) {
    output=output+"\t"+ObjectName+"("+ObjectType+");
    l=l+1;
    ObjectName=SharePools[i].Class[j].PublicSection.Object[l];
    ObjectType=SharePools[i].Class[j].PublicSection.ObjectType[l];
}

output=output+"\n";

//protected section
output=output+"\n\nProtected Section Methods & Objects:\n";
for (int l=1; l<SMR[i][j]; l++) {
    String MethodName=SharePools[i].Class[j].ProtectedSection.Method[l].MethodName;
    output=output+"\t\tMethod Name:"+MethodName+"\n";
    //ReturnedValueType
    String ReturnedValueType=SharePools[i].Class[j].ProtectedSection.Method[l].ReturnedValueType;
    output=output+"\t\tReturned ValueType:\n"+ReturnedValueType+"\n";
    //IN_ARGS
    output=output+"\t\tt\tIN_ArgNames(Types): ";
    l=1;
    String IN_ARG=SharePools[i].Class[j].ProtectedSection.Method[l].ARGS.IN_ArgName[l];
    String IN_ARGType=SharePools[i].Class[j].ProtectedSection.Method[l].ARGS.IN_ArgType[l];
    while (IN_ARG!=null) {
        output=output+"\t"+IN_ARG+"("+IN_ARGType+");
        l=l+1;
        IN_ARG=SharePools[i].Class[j].ProtectedSection.Method[l].ARGS.IN_ArgName[l];
        IN_ARGType=SharePools[i].Class[j].ProtectedSection.Method[l].ARGS.IN_ArgType[l];
    }

    //OUT_ARGS
    output=output+"\n\t\tt\tOUT_ArgNames(Types): ";
    l=1;
    String OUT_ARG=SharePools[i].Class[j].ProtectedSection.Method[l].ARGS.OUT_ArgName[l];
    String OUT_ARGType=SharePools[i].Class[j].ProtectedSection.Method[l].ARGS.OUT_ArgType[l];
while (OUT_ARG!=null){
    output=output+"\n"+OUT_ARG+"("+OUT_ARGType+"\n";
    ll=ll+1;
    OUT_ARG=SharedPools[i].Class[j].ProtectedSection.Method[ll].ARGS.OUT_AргName[ll];
    OUT_ARGType=SharedPools[i].Class[j].ProtectedSection.Method[ll].ARGS.OUT_AргType[ll];
}
output=output+"\n\n";

//objects of in protected section
output=output+"\n\n\nObjects: \n";
ll=1;
ObjectName=SharedPools[i].Class[j].ProtectedSection.Object[ll];
ObjectType=SharedPools[i].Class[j].ProtectedSection.ObjectType[ll];
while (ObjectName!=null){
    output=output+"\n"+ObjectName+"("+ObjectType+"\n";
    ll=ll+1;
    ObjectName=SharedPools[i].Class[j].ProtectedSection.Object[ll];
    ObjectType=SharedPools[i].Class[j].ProtectedSection.ObjectType[ll];
}
output=output+"\n\n";

//private section
output=output+"\n\n\nPrivate Section Methods & Objects\n";
for (int l=1; l<=SMV[i][j]; l++){
    String MethodName=SharedPools[i].Class[j].PrivateSection.Method[l].MethodName;
    output=output+"\n\n\nMethodName: "+MethodName+"\n";
    //ReturnedValueType
    String ReturnedValueType=SharedPools[i].Class[j].PrivateSection.Method[l].ReturnedValue
    output=output+"\n\n\nReturnedValue\n"+ReturnedValue+"\n";
    //IN_ARGS
    output=output+"\n\n\nIN_ArgNames(Types): \n";
    ll=1;
    String IN_ARG=SharedPools[i].Class[j].PrivateSection.Method[l].ARGS.IN_AргName[ll];
    String IN_ARGType=SharedPools[i].Class[j].PrivateSection.Method[l].ARGS.IN_AргType[ll];
    while (IN_ARG!=null){
        output=output+"\n"+IN_ARG+"("+IN_ARGType+"\n";
        ll=ll+1;
        IN_ARG=SharedPools[i].Class[j].PrivateSection.Method[l].ARGS.IN_AргName[ll];
        IN_ARGType=SharedPools[i].Class[j].PrivateSection.Method[l].ARGS.IN_AргType[ll];
    }
    //OUT_ARGS
    output=output+"\n\n\nOUT_ArgNames(Types): \n";
    ll=1;
    String OUT_ARG=SharedPools[i].Class[j].PrivateSection.Method[l].ARGS.OUT_AргName[ll];
    String OUT_ARGType=SharedPools[i].Class[j].PrivateSection.Method[l].ARGS.OUT_AргType[ll];
    while (OUT_ARG!=null){
        output=output+"\n"+OUT_ARG+"("+OUT_ARGType+"\n";
        ll=ll+1;
        OUT_ARG=SharedPools[i].Class[j].PrivateSection.Method[l].ARGS.OUT_AргName[ll];
        OUT_ARGType=SharedPools[i].Class[j].PrivateSection.Method[l].ARGS.OUT_AргType[ll];
    }
output=output+"\n\n";

//objects of in private section
output=output+"\n\n\nObjects: \n";
ll=1;
ObjectName=SharedPools[i].Class[j].PrivateSection.Object[ll];
ObjectType=SharedPools[i].Class[j].PrivateSection.ObjectType[ll];
while (ObjectName!=null){
    output=output+""+ObjectName+"("+ObjectType+");
    ll=ll+1;
    ObjectName=SharedPools[i].Class[i].PrivateSection.Object[ll];
    ObjectType=SharedPools[i].Class[i].PrivateSection.ObjectType[ll];
}
output=output+"\n\n";
}
reportStructure(output);
}
protected static void initializeFor_Example_1 () {
    initializeFrameworks_Example_1();
    initializeSharedPools_Example_1();
    display();
    return;
}
protected static void initializeFor_Example_2 () {
    initializeFrameworks_Example_2();
    initializeSharedPools_Example_2();
    display();
    return;
}
protected static void initializeFor_Example_3_ACE () {
    initializeFrameworks_Example_3_ACE();
    initializeSharedPools_Example_3_ACE();
    display();
    return;
}
protected static void initializeFrameworks_Example_10 {
    // ****************************************************
    // STEP#1 read values for:
    // F, C[], c[].
    // MP[], MR[], MV[]
    // OP[], OR[], OV[]
    // ****************************************************
    //number of frameworks
    F=2;
    //number of components per framework
    C[1]=2; C[2]=2;
    //number of classes per component per framework
    c[1][1]=2; c[1][2]=2;
    c[2][1]=2; c[2][2]=2;
    //number of public methods per class per component per framework
    MP[1][1][1]=2; MP[1][1][2]=2;
    MP[1][2][1]=2; MP[1][2][2]=2;
    MP[2][1][1]=2; MP[2][1][2]=2;
    MP[2][2][1]=2; MP[2][2][2]=2;
    //number of protected methods per class per component per framework
    MR[1][1][1]=1; MR[1][1][2]=1;
    MR[1][2][1]=1; MR[1][2][2]=1;
    MR[2][1][1]=1; MR[2][1][2]=1;
    MR[2][2][1]=1; MR[2][2][2]=1;
    //number of private methods per class per component per framework
    MV[1][1][1]=1; MV[1][1][2]=1;
    MV[1][2][1]=1; MV[1][2][2]=1;
    MV[2][1][1]=1; MV[2][1][2]=1;
}
MV[2][2][1]=1;   MV[2][2][2]=1;
//number of public objects per class per component per framework
OP[1][1][1]=0;   OP[1][1][2]=1;
OP[1][2][1]=1;   OP[1][2][2]=0;
OP[2][1][1]=1;   OP[2][1][2]=0;
OP[2][2][1]=0;   OP[2][2][2]=0;
//number of protected methods per class per component per framework
OR[1][1][1]=2;   OR[1][1][2]=2;
OR[1][2][1]=2;   OR[1][2][2]=2;
OR[2][1][1]=2;   OR[2][1][2]=2;
OR[2][2][1]=2;   OR[2][2][2]=2;
//number of private methods per class per component per framework
OV[1][1][1]=4;   OV[1][1][2]=4;
OV[1][2][1]=4;   OV[1][2][2]=4;
OV[2][1][1]=4;   OV[2][1][2]=4;
OV[2][2][1]=4;   OV[2][2][2]=4;
//number of classes inherited in public per class per component per framework
MinhP[1][1][1]=1;  MinhP[1][1][2]=0;
MinhP[1][2][1]=1;  MinhP[1][2][2]=0;
MinhP[2][1][1]=1;  MinhP[2][1][2]=0;
MinhP[2][2][1]=1;  MinhP[2][2][2]=0;
//number of classes inherited in protected per class per component per framework
MinhR[1][1][1]=0;  MinhR[1][1][2]=0;
MinhR[1][2][1]=0;  MinhR[1][2][2]=0;
MinhR[2][1][1]=0;  MinhR[2][1][2]=0;
MinhR[2][2][1]=0;  MinhR[2][2][2]=0;
//number of classes inherited in private per class per component per framework
MinhV[1][1][1]=0;  MinhV[1][1][2]=0;
MinhV[1][2][1]=0;  MinhV[1][2][2]=0;
MinhV[2][1][1]=0;  MinhV[2][1][2]=0;
MinhV[2][2][1]=0;  MinhV[2][2][2]=0;
//number of classes inherited from shared pools in public per class per component per framework
MinhPS[1][1][1]=0;  MinhPS[1][1][2]=0;
MinhPS[1][2][1]=0;  MinhPS[1][2][2]=0;
MinhPS[2][1][1]=0;  MinhPS[2][1][2]=0;
MinhPS[2][2][1]=0;  MinhPS[2][2][2]=0;
//number of classes inherited from shared pools in protected per class per component per framework
MinhRS[1][1][1]=0;  MinhRS[1][1][2]=0;
MinhRS[1][2][1]=0;  MinhRS[1][2][2]=0;
MinhRS[2][1][1]=0;  MinhRS[2][1][2]=0;
MinhRS[2][2][1]=0;  MinhRS[2][2][2]=0;
//number of classes inherited from shared pools in private per class per component per framework
MinhVS[1][1][1]=0;  MinhVS[1][1][2]=0;
MinhVS[1][2][1]=0;  MinhVS[1][2][2]=0;
MinhVS[2][1][1]=0;  MinhVS[2][1][2]=0;
MinhVS[2][2][1]=0;  MinhVS[2][2][2]=0;  //
******************************************************************************

// STEP#2: instantiate for the Frameworks objects and the embedded Component, MainSection,
// Class, PublicSection, ProtectedSection, PrivateSection, Method, etc. etc. objects
// ALSO: Assign ID values for each object
******************************************************************************

// instantiate the framework objects
for (int i=1; i<=F; i++)
{   
    Frameworks[i]=new FrameworkNode();
    Frameworks[i].FrameworkNodeNumber=i;
}

// instantiate the component objects for this new framework object
for (int j=1; j<=C[i]; j++){
Frameworks[i].Component[j]=new ComponentNode();
Frameworks[i].Component[j].ID=ComponentNodeNumber=j;
Frameworks[i].Component[j].ID.ParenFrameworkNodeNumber=i;
Frameworks[i].Component[j].MainSection="main_"++"_"++j;
// instantiate the class objects for this component object
for (int k=1; k<=c[i][j]; k++) {
    Frameworks[i].Component[j].Class[k]=new ClassNode();
    Frameworks[i].Component[j].Class[k].ID=ClassNumber=k;
    Frameworks[i].Component[j].Class[k].ID.ClassName="class"++j+k;
    Frameworks[i].Component[j].Class[k].ID.ParenComponent.ComponentNodeNumber=j;
    Frameworks[i].Component[j].Class[k].ID.ParenComponent.ParenFrameworkNodeNumber=i;
    // instantiate for the public section of this class
    Frameworks[i].Component[j].Class[k].PublicSection=new PublicSectionNode();
    for (int l=1; l<=MP[i][j][k]; l++)
    Frameworks[i].Component[j].Class[k].PublicSection.Method[l].ID=MethodNumber=l;
    Frameworks[i].Component[j].Class[k].PublicSection.Method[l].ID.ParenClass.ClassNodeNumber=k;
}
// instantiate for the protected section of this class
Frameworks[i].Component[j].Class[k].ProtectedSection=new ProtectedSectionNode();
for (int l=1; l<=MR[i][j][k]; l++)
    Frameworks[i].Component[j].Class[k].ProtectedSection.Method[l].ID=MethodNumber=l;
    Frameworks[i].Component[j].Class[k].ProtectedSection.Method[l].ID.ParenClass.ClassNodeNumber=k;
}
// instantiate for the private section of this class
Frameworks[i].Component[j].Class[k].PrivateSection=new PrivateSectionNode();
for (int l=1; l<=MV[i][j][k]; l++)
    Frameworks[i].Component[j].Class[k].PrivateSection.Method[l].ID=MethodNumber=l;
    Frameworks[i].Component[j].Class[k].PrivateSection.Method[l].ID.ParenClass.ClassNodeNumber=k;
}
/*
// STEP#3: Read the entries for each of these objects
//*********************************************************/
//CLASS(1,1,1)

//CLASS(1,2,1)

//CLASS(1,2,2)

286
Appendix B1: A Prototype For CO-Toolkits' Design and Implementation Structures: An Overview


//CLASS(2,1,1)

//CLASS(2,1,2)
//CLASS(2,2,1)

*******************************************************************************

//STEP#4: read in the inheritance references (CClassIDs) for each class

--------------------------------------------------------------------------------

//Class(1,1,1) inherits from two other classes in public — VIOLation — Multiple inheritance
protected static void initializeFrameworks_Example_20 {

    // ********************************************************
    // STEP#1 read values for:
    //     F, C[], c1[],
    //     MP[], MR[], MV[]
    //     OP[], OR[], OV[]
    // ********************************************************

    // number of frameworks
    F=3;
    // number of components per framework
    C[1]=2; C[2]=2; C[3]=1;
    // number of classes per component per framework
    c[1][1]=2; c[1][2]=2;
    c[2][1]=2; c[2][2]=2;
    c[3][1]=1;
    // number of public methods per class per component per framework
    MP[1][1][1]=2; MP[1][1][2]=2;
    MP[1][2][1]=2; MP[1][2][2]=2;
    MP[2][1][1]=2; MP[2][1][2]=2;
    MP[2][2][1]=2; MP[2][2][2]=2;
    MP[3][1][1]=1;

    // number of protected methods per class per component per framework
    MR[1][1][1]=1; MR[1][1][2]=1;
    MR[1][2][1]=1; MR[1][2][2]=1;
    MR[2][1][1]=1; MR[2][1][2]=1;
    MR[2][2][1]=1; MR[2][2][2]=1;
    MR[3][1][1]=1;

    // number of private methods per class per component per framework
    MV[1][1][1]=1; MV[1][1][2]=1;
    MV[1][2][1]=1; MV[1][2][2]=1;
    MV[2][1][1]=1; MV[2][1][2]=1;

289
Appendix B1: A Prototype For CO-Toolkits' Design and Implementation Structures: An Overview

\[ MV[2][2][1]=1; \quad MV[2][2][2]=1; \]
\[ MV[3][1][1]=1; \]

//number of public objects per class per component per framework
\[ OP[1][1][1]=0; \quad OP[1][1][2]=1; \]
\[ OP[1][2][1]=1; \quad OP[1][2][2]=0; \]
\[ OP[2][1][1]=1; \quad OP[2][1][2]=0; \]
\[ OP[2][2][1]=0; \quad OP[2][2][2]=0; \]
\[ OP[3][1][1]=1; \]

//number of protected methods per class per component per framework
\[ OR[1][1][1]=2; \quad OR[1][1][2]=2; \]
\[ OR[1][2][1]=2; \quad OR[1][2][2]=2; \]
\[ OR[2][1][1]=2; \quad OR[2][1][2]=2; \]
\[ OR[2][2][1]=2; \quad OR[2][2][2]=2; \]
\[ OR[3][1][1]=1; \]

//number of private methods per class per component per framework
\[ OV[1][1][1]=4; \quad OV[1][1][2]=4; \]
\[ OV[1][2][1]=4; \quad OV[1][2][2]=4; \]
\[ OV[2][1][1]=4; \quad OV[2][1][2]=4; \]
\[ OV[2][2][1]=4; \quad OV[2][2][2]=4; \]
\[ OV[3][1][1]=1; \]

//number of classes inherited in public per class per component per framework
\[ MinhP[1][1][1]=2; \quad MinhP[1][1][2]=0; \]
\[ MinhP[1][2][1]=1; \quad MinhP[1][2][2]=0; \]
\[ MinhP[2][1][1]=1; \quad MinhP[2][1][2]=0; \]
\[ MinhP[2][2][1]=1; \quad MinhP[2][2][2]=0; \]
\[ MinhP[3][1][1]=1; \]

//number of classes inherited in protected per class per component per framework
\[ MinhR[1][1][1]=1; \quad MinhR[1][1][2]=0; \]
\[ MinhR[1][2][1]=0; \quad MinhR[1][2][2]=1; \]
\[ MinhR[2][1][1]=0; \quad MinhR[2][1][2]=0; \]
\[ MinhR[2][2][1]=0; \quad MinhR[2][2][2]=0; \]
\[ MinhR[3][1][1]=1; \]

//number of classes inherited in private per class per component per framework
\[ MinhV[1][1][1]=1; \quad MinhV[1][1][2]=0; \]
\[ MinhV[1][2][1]=0; \quad MinhV[1][2][2]=0; \]
\[ MinhV[2][1][1]=0; \quad MinhV[2][1][2]=0; \]
\[ MinhV[2][2][1]=0; \quad MinhV[2][2][2]=0; \]
\[ MinhV[3][1][1]=1; \]

//number of classes inherited from shared pools in public per class per component per framework
\[ MinhPS[1][1][1]=0; \quad MinhPS[1][1][2]=0; \]
\[ MinhPS[1][2][1]=0; \quad MinhPS[1][2][2]=0; \]
\[ MinhPS[2][1][1]=0; \quad MinhPS[2][1][2]=0; \]
\[ MinhPS[2][2][1]=1; \quad MinhPS[2][2][2]=0; \]
\[ MinhPS[3][1][1]=1; \]

//number of classes inherited from shared pools in protected per class per component per framework
\[ MinhRS[1][1][1]=0; \quad MinhRS[1][1][2]=0; \]
\[ MinhRS[1][2][1]=0; \quad MinhRS[1][2][2]=0; \]
\[ MinhRS[2][1][1]=0; \quad MinhRS[2][1][2]=0; \]
\[ MinhRS[2][2][1]=0; \quad MinhRS[2][2][2]=1; \]
\[ MinhRS[3][1][1]=1; \]

290
//number of classes inherited from shared pools in private per class per component per framework
MinhVS[1][1][1]=1; MinhVS[1][1][2]=0;
MinhVS[1][2][1]=0; MinhVS[1][2][2]=0;
MinhVS[2][1][1]=0; MinhVS[2][1][2]=0;
MinhVS[2][2][1]=0; MinhVS[2][2][2]=0;
MinhVS[3][1][1]=1;

//****************************
//STEP2: instantiate for the Frameworks
//objects and the embedded Component, MainSection,
//Class, PublicSection, ProtectedSection, PrivateSection,
//Method, etc. etc. objects
//ALSO: Assign ID values for each object
//*************************

// instantiate the framework objects
for (int i=1; i<=F; i++){
Frameworks[i]=new FrameworkNode();
Frameworks[i].FrameworkNodeNumber=i;
}

//instantiate the component objects for this new framework object
for (int j=1; j<=C[i]; j++){
Frameworks[i].Component[j]=new ComponentNode();
Frameworks[i].Component[j].ID=ComponentNodeNumber=j;
Frameworks[i].Component[j].ID=ParentFrameworkNodeNumber=i;
Frameworks[i].Component[j].MainSection="main"+i+j+";"+j;
// instantiate the class objects for this component object
for (int k=1; k<=C[i][j]; k++){
Frameworks[i].Component[j].Class[k]=new ClassNode();
Frameworks[i].Component[j].Class[k].ID=ClassNodeNumber=k;
Frameworks[i].Component[j].Class[k].ID=ComponentNodeNumber=i;
Frameworks[i].Component[j].Class[k].ID=ParentComponent.ComponentNodeNumber=j;
Frameworks[i].Component[j].Class[k].ID=ParentClass.ParentFrameworkNodeNumber=i;
// instantiate for the public section of this class
Frameworks[i].Component[j].Class[k].PublicSection=new PublicSectionNode();
for (int l=1; l<=MP[i][j][k]; l++){
Frameworks[i].Component[j].Class[k].PublicSection.Method[l].ReturnedValue="void";
Frameworks[i].Component[j].Class[k].PublicSection.Method[l].ID=MethodNodeNumber=l;
Frameworks[i].Component[j].Class[k].PublicSection.Method[l].ID=ParentClass.ClassNodeNumber=k;
}
// instantiate for the protected section of this class
Frameworks[i].Component[j].Class[k].ProtectedSection=new ProtectedSectionNode();
for (int l=1; l<=MP[i][j][k]; l++){
Frameworks[i].Component[j].Class[k].ProtectedSection.Method[l].ReturnedValue="void";
Frameworks[i].Component[j].Class[k].ProtectedSection.Method[l].ID=MethodNodeNumber=l;
Frameworks[i].Component[j].Class[k].ProtectedSection.Method[l].ID=ParentClass.ClassNodeNumber=k;
}
// instantiate for the private section of this class

Frameworks[i].Component[j].Class[k].PrivateSection=new PrivateSectionNode();
for (int l=1; l<=MV[i][j][k][l]; l++)
    Frameworks[i].Component[j].Class[k].PrivateSection.Method[l].ReturnedValueType="void";
    Frameworks[i].Component[j].Class[k].PrivateSection.Method[l].ID.MethodNodeNumber=i;
    Frameworks[i].Component[j].Class[k].PrivateSection.Method[l].ID.ParentClass.ClassNodeNumber=k;

//********************************************************************
// STEP#3: Read the entries for each of these objects
//********************************************************************

//CLASS(111,1)
//Class(111,1) public section contains only two methods and no objects
//MP[1][1][1]=2 && OP[1][1][1]=0
//get in ARGS for this method

//get in ARGS for this method

//Class(1,1,1) protected section contains one method and 2 objects
//MR[1][1][1]=2 && OR[1][1][1]=2
//get in ARGS for this method


//Class(1,1,1) private section contains 1 method and 4 objects
//MV[1][1][1]=1 && OV[1][1][1]=4
//ARGS for this method

//CLASS(1,1,2)
//Class(1,1,2) public section contains 2 methods and 1 object == VIOLATION
//get ARGS

//get ARGS

//Class(1,1,2) protected section contains 1 method and 2 objects
//MR[1][1][2]=1 && OR[1][1][2]=2
//get ARGS

//Class(1,1,2) private section contains 1 method and 4 objects
//MV[1][1][2]=1 && OV[1][1][2]=4
//get ARGS

//CLASS(1,2,1)
//Class(1,2,1) public section contains 2 methods and 1 object == VIOLATION
//MP[1][2][1]=2 && OP[1][2][1]=1

293
Appendix B1: A Prototype For CO-Toolkits’ Design and Implementation Structures: An Overview

//Class[1,2,2] private section contains 1 method and 4 objects
//MV[1][2][2]=1 && OV[1][2][2]=4
//get ARGS


//CLASS(2,1,1)
//Class(2,1,1) public section contains 2 methods and 1 object — VIOLATION
//MP[2][1][1]=2 && OP[2][1][1]=1
// get ARGS

// get ARGS

//Class(2,1,1) protected section contains 1 method and 2 objects
//MR[2][1][1]=1 && OR[2][1][1]=2
// get ARGS

//Class(2,1,1) private section contains 1 method and 4 objects
//MV[2][1][1]=1 && OV[2][1][1]=4
// get ARGS

Appendix B1: A Prototype For CO-Toolkits’ Design and Implementation Structures: An Overview

```
//get ARGS

//Class(2,2,1) private section contains 1 method and 4 objects
//MV[2][2][1]=1 & & OV[2][2][1]=4
//get ARGS

//CLASS(2,2,2)
//Class(2,2,2) public section contains 2 methods and no object
//MP[2][2][1]=2 & & OP[2][2][1]=0
//get ARGS

//get ARGS

//get ARGS

//Class(2,2,2) private section contains 1 method and 4 objects
//MV[2][2][1]=1 & & OV[2][2][1]=4
//get ARGS

```
//CLASS(3,1,1)
//Class(3,1,1) public section contains 1 methods and 1 object == VIOLATION
//MP[3][1][1]=1 & OP[3][1][1]=1
//get ARGS
//
//Class(3,1,1) protected section contains 1 method and 1 objects
//MR[3][1][1]=1 & OR[3][1][1]=1
//get ARGS
//
//Class(3,1,1) private section contains 1 method and 1 objects
//MV[3][1][1]=1 & OV[3][1][1]=1
//get ARGS
//
// *********************************************************************
// STEP#4: read in the inheritance references (ClassIDs) for each class
// *********************************************************************
//Class(1,1,1) inherits from two other classes in public — VIOLATION — Multiple inheritance
//Class(1,1,1) inherits another class in protected — Violation — non-public inheritance
//Class(1,1,1) inherits from another class in private — Violation — non-public inheritance
Appendix B1: A Prototype For CO-Toolkits' Design and Implementation Structures: An Overview

Number=1;
Frameworks[1].Component[1].Class[1].PrivateInheritedSharedClasses[1].ClassNameNodeNumber=1;
// Class(1,1,2) does not inherit from any other class // Class(1,2,1) inherits only from one other class in public
Frameworks[1].Component[2].Class[1].PublicInheritedClasses[1].ClassNameNodeNumber=1;
Frameworks[1].Component[2].Class[1].PublicInheritedClasses[1].ParentComponent.ParentFrameworkNodeNumber=1; // Class(1,2,2) inherits only from one another class in protected
Frameworks[1].Component[2].Class[2].ProtectedInheritedClasses[1].ClassNameNodeNumber=1;
Frameworks[2].Component[1].Class[1].PublicInheritedClasses[1].ClassNameNodeNumber=1;
Frameworks[2].Component[1].Class[1].PublicInheritedClasses[1].ParentComponent.ParentFrameworkNodeNumber=2; // Class(2,1,2) does not inherit from any other class // Class(2,2,1) inherits from one other class in public
Frameworks[2].Component[2].Class[1].PublicInheritedSharedClasses[1].ClassNameNodeNumber=1;
// Class(2,2,2) does not inherit from another class from any framework
// Class(2,2,2) inherits in protected another class from shared pools
// Class(3,1,1) inherits from two other classes in public — VIOLATION — Multiple inheritance
Frameworks[3].Component[1].Class[1].PublicInheritedClasses[1].ClassNameNodeNumber=1;
// Class(3,1,1) inherits another class in public from shared pools
Frameworks[3].Component[1].Class[1].PublicInheritedSharedClasses[1].ClassNameNodeNumber=1;
// Class(3,1,1) inherits another class in protected — VIOLATION — non-public inheritance

299
Appendix B1: A Prototype For CO-Toolkits’ Design and Implementation Structures: An Overview

Frameworks[3].Component[1].Class[1].ProtectedInheritedClasses[1].ClassName=2;
Number=1;
NodeNumber=1;
// Class(3,1,1) inherits another class in protected from shared pools
Frameworks[3].Component[1].Class[1].ProtectedInheritedSharedClasses[1].ClassName=2;
ComponentNodeNumber=2;
// Class(3,1,1) inherits another class in private — Violation — non-public inheritance
ComponentNodeNumber=2;
Number=2;
// Class(3,1,1) inherits another class in private from shared pools
ComponentNodeNumber=1;
ber=1;
}

protected static void initializeFrameworks_Example_3_ACE0 {
//**********************************************************
// STEP#1 read values for:
// F, C[], e[],
// MP[[]][[]], MR[[]][[]], MV[[]][[]]
// OP[[]][[]], OR[[]][[]], OV[[]][[]]
//**********************************************************

//number of frameworks
F=15;

//number of components per framework
C[1]=9; // [ASX]
C[2]=2; // [CORBA]
C[3]=3; // [Collections]
C[4]=9; // [Concurrency]
C[5]=5; // [Connection]
C[6]=44; // [IPC]
C[7]=2; // [Log_Msg]
C[8]=7; // [Memory]
C[9]=4; // [Misc]
C[10]=7; // [Name_Service]
C[11]=1; // [OS]
C[12]=8; // [Reactor]
C[13]=7; // [System_V_IPC]
C[14]=2; // [Timers]
C[15]=7;

//number of classes per component per framework
c[1][1]=1; c[1][2]=3; c[1][3]=1; c[1][4]=4; c[1][5]=1;
c[1][6]=2; c[1][7]=2; c[1][8]=3; c[1][9]=3;
c[2][1]=3; c[2][2]=1;
c[3][1]=10; c[3][2]=8; c[3][3]=4;
c[4][1]=1;

300
Appendix B1: A Prototype For CO-Toolkits’ Design and Implementation Structures: An Overview

c[4][2]=1;
c[4][3]=4;
c[4][4]=23;
c[4][5]=1;
c[4][6]=14;
c[4][7]=1;
c[4][8]=6;
c[4][9]=1; c[5][1]=3; c[5][2]=2; c[5][3]=1; c[5][4]=5; c[5][5]=1;
c[6][1]=1; c[6][2]=1; c[6][3]=1; c[6][4]=1; c[6][5]=1;
c[6][16]=1; c[6][17]=1; c[6][18]=1; c[6][19]=1; c[6][20]=1;
c[6][21]=1; c[6][22]=1; c[6][23]=1; c[6][24]=2; c[6][25]=1;
c[6][26]=1; c[6][27]=1; c[6][28]=1; c[6][29]=1; c[6][30]=1;
c[6][31]=1; c[6][32]=1; c[6][33]=1; c[6][34]=1; c[6][35]=1;
c[6][36]=1; c[6][37]=1; c[6][38]=1; c[6][39]=3; c[6][40]=1;
c[6][41]=1; c[6][42]=1; c[6][43]=1; c[6][44]=1; c[7][1]=1; c[7][2]=1;
c[8][1]=1; c[8][2]=3; c[8][3]=3; c[8][4]=9; c[8][5]=1;
c[8][6]=1; c[8][7]=1; c[9][1]=2; c[9][2]=1; c[9][3]=1; c[9][4]=1;
c[10][1]=2; c[10][2]=2; c[10][3]=1; c[10][4]=2; c[10][5]=2;
c[10][6]=2; c[10][7]=1;
c[11][1]=5; c[12][1]=1; c[12][2]=1; c[12][3]=2; c[12][4]=8; c[12][5]=3;
c[12][6]=7; c[12][7]=2; c[12][8]=1;
c[15][5]=1;
c[15][6]=3; c[15][7]=2;// class names

// [ASX]
className[1][1][1] = "ACE_Module_Link";
className[1][2][1] = "ACE_Map_Manager";
className[1][2][2] = "ACE_Map_Iterator";
className[1][2][3] = "ACE_Map_Reverse_Iterator";
className[1][3][1] = "ACE_Message_Block";
className[1][4][1] = "ACE_Notification_Strategy";
className[1][4][2] = "ACE_Message_Queue";
className[1][4][3] = "ACE_Message_Queue_Iterator";
className[1][4][4] = "ACE_Message_Queue_Reverse_Iterator";
className[1][5][1] = "ACE_Module";
className[1][6][1] = "Driver";
className[1][6][2] = "ACE_Multiplexor";
className[1][7][1] = "ACE_Stream";
className[1][7][2] = "ACE_Stream_Iterator";
className[1][8][1] = "ACE_Stream_Head";
className[1][8][2] = "ACE_Stream_Tail";
className[1][8][3] = "ACE_Thru_Task";
className[1][9][1] = "ACE_Task_Flags";
className[1][9][2] = "ACE_Task_Base";
className[1][9][3] = "ACE_Task_Exit";// [CORBA]
className[2][1][1] = "ACE_CORBA_Handler";
className[2][1][2] = "ACE_ST_CORBA_Handler";
className[2][1][3] = "ACE_MT_CORBA_Handler";
className[2][2][1] = "ACE_CORBA_Ref";// [Collections]
className[3][1][1] = "ACE_Set_Node";
className[3][1][2] = "ACE_Unbounded_Set_Iterator";
className[3][1][3] = "ACE_Set_Node";

301
Appendix B1: A Prototype For CO-Toolkits’ Design and Implementation Structures: An Overview

className[3][1][4]= "ACE_Unbounded_Set";
className[3][1][5]= "ACE_Fixed_Set";
className[3][1][6]= "ACE_Fixed_Set_4ter";
className[3][1][7]= "ACE_Fixed_Set";
className[3][1][8]= "ACE_Bounded_Set";
className[3][1][9]= "ACE_Bounded_Set_4ter";
className[3][1][10]= "ACE_Bounded_Set";
className[3][2][1]= "ACE_Stack_Node";
className[3][2][2]= "ACE_Queue_Node";
className[3][2][3]= "ACE_Bounded_Stack";
className[3][2][4]= "ACE_Fixed_Stack";
className[3][2][5]= "ACE_Stack_Node";
className[3][2][6]= "ACE_Bounded_Stack";
className[3][2][7]= "ACE_Queue_Node";
className[3][2][8]= "ACE_Bounded_Queue";
className[3][3][1]= "ACE_Allocator";
className[3][3][2]= "ACE_CString";
className[3][3][3]= "ACE_SString";
className[3][3][4]= "ACE_WString"; // [Concurrency]
className[4][1][1]= "ACE_ACTIVATION_Queue";
className[4][2][1]= "ACE_Method_Object";
className[4][3][1]= "ACE_Process_Descriptor";
className[4][3][2]= "ACE_Process_Control";
className[4][3][3]= "ACE_Process_Manager";
className[4][3][4]= "ACE_Process_Control";
className[4][4][1]= "ACE_File_Lock";
className[4][4][2]= "ACE_Semaphore";
className[4][4][3]= "ACE_Process_Semaphore";
className[4][4][4]= "ACE_RW_Mutex";
className[4][4][5]= "ACE_Mutex";
className[4][4][6]= "ACE_Process_Mutex";
className[4][4][7]= "ACE_RW_Process_Mutex";
className[4][4][8]= "ACE_Null_BARRIER";
className[4][4][9]= "ACE_Null_Mutex";
className[4][4][10]= "ACE_Null.Condition_Mutex";
className[4][4][11]= "ACE_Null_Mutex.Guard";
className[4][4][12]= "ACE.Event";
className[4][4][13]= "ACE.Manual_Event";
className[4][4][14]= "ACE.Auto_Event";
className[4][4][15]= "ACE_Thread_Mutex";
className[4][4][16]= "ACE_Thread.Mutex.Guard";
className[4][4][17]= "ACE.Condition_Thread_Mutex";
className[4][4][18]= "ACE.Recursive_Thread_Mutex";
className[4][4][19]= "ACE_RW_Thread_Mutex";
className[4][4][20]= "ACE_Thread_Semaphore";
className[4][4][21]= "ACE_BARRIER";
className[4][4][22]= "ACE_Process.Barrier";
className[4][4][23]= "ACE_Deadline_Barrier";
className[4][5][1]= "ACE_Synch_Options";
className[4][6][1]= "ACE_Test_and_Set";
className[4][6][2]= "ACE.Atomic.Op";
className[4][6][3]= "ACE_TSS";
className[4][6][4]= "ACE_Null_SYNCH";
className[4][6][5]= "ACE.Guard";
className[4][6][6]= "ACE_Write.Guard";
className[4][6][7]= "ACE_Read.Guard";
className[4][6][8]= "ACE_TSS.Guard";
className[4][6][9]= "ACE_TSS_Write.Guard";

302
classList[4][6][10] = "ACE_DSS_Read_Guard";
classList[4][6][11] = "ACE_Condition";
classList[4][6][12] = "ACE_Process_Condition";
classList[4][6][13] = "ACE_Thread_Condition";
classList[4][6][14] = "ACE_MT_SYNCH";
classList[4][7][1] = "ACE_Thread";
classList[4][8][1] = "ACE_Task_Base";
classList[4][8][2] = "ACE_Thread_Manager";
classList[4][8][3] = "ACE_Thread_Descriptor";
classList[4][8][4] = "ACE_Thread_Control";
classList[4][8][5] = "ACE_Thread_Manager";
classList[4][8][6] = "ACE_Thread_Control";
classList[4][9][1] = "ACE_Token"/\[Connection\]
classList[5][1][1] = "ACE_Acceptor";
classList[5][1][2] = "ACE_Strategy_Acceptor";
classList[5][1][3] = "ACE_Oneshot_Acceptor";
classList[5][2][1] = "ACE_Svc_Tuple";
classList[5][2][2] = "ACE_Connector";
classList[5][3][1] = "ACE_Dynamic_Service";
classList[5][4][1] = "ACE_Reactor";
classList[5][4][2] = "ACE_ReactorEx";
classList[5][4][3] = "ACE_Notification_Strategy";
classList[5][4][4] = "ACE_Reactor_Notification_Strategy";
classList[5][4][5] = "ACE_ReactorEx_Notification_Strategy";
classList[5][5][1] = "ACE_Svc_Handler"/\[IPC\]
classList[6][1][1] = "ACE_IO_SAP";
classList[6][2][1] = "ACE_DEV";
classList[6][3][1] = "ACE_DEV_Connector";
classList[6][4][1] = "ACE_DEV_IO";
classList[6][5][1] = "ACE_FILE";
classList[6][6][1] = "ACE_FILE_Connector";
classList[6][7][1] = "ACE_FILE_IO";
classList[6][8][1] = "ACE_IPC_SAP";
classList[6][9][1] = "ACE_Addr";
classList[6][10][1] = "ACE_DEV_Addr";
classList[6][11][1] = "ACE_FILE_Addr";
classList[6][12][1] = "ACE_INET_Addr";
classList[6][13][1] = "ACE_SPIPE_Addr";
classList[6][14][1] = "ACE_UNIX_Addr";
classList[6][15][1] = "ACE_FIFO";
classList[6][16][1] = "ACE_FIFO_Recv";
classList[6][17][1] = "ACE_FIFO_Recv_Msg";
classList[6][18][1] = "ACE_FIFO_Send";
classList[6][19][1] = "ACE_FIFO_Send_Msg";
classList[6][20][1] = "ACE_LSOCK";
classList[6][21][1] = "ACE_LSOCK_Acceptor";
classList[6][22][1] = "ACE_LSOCK_CODgram";
classList[6][23][1] = "ACE_LSOCK_Connector";
classList[6][24][1] = "ACE_LSOCK_Dgram";
classList[6][24][2] = "ACE_LSOCK_Stream";
classList[6][25][1] = "ACE.SOCK";
classList[6][26][1] = "ACE.SOCK_Acceptor";
classList[6][27][1] = "ACE.SOCK_CODgram";
classList[6][28][1] = "ACE.SOCK_Connector";
classList[6][29][1] = "ACE.SOCK_Dgram";
classList[6][30][1] = "ACE.SOCK_Dgram_Bcast";
classList[6][31][1] = "ACE.SOCK_Dgram_Mcast";
classList[6][32][1] = "ACE.SOCK_IO";

303
Appendix B1: A Prototype For CO-Toolkits' Design and Implementation Structures: An Overview

className[6][33][1] = "ACE.SOCK_Stream";
className[6][34][1] = "ACE_SPIPE";
className[6][35][1] = "ACE_SPIPE_Acceptor";
className[6][36][1] = "ACE_SPIPE_Connector";
className[6][37][1] = "ACE_SPIPE_Stream";
className[6][38][1] = "ACE_TLI";
className[6][39][1] = "ACE_TLI_Request_Queue";
className[6][39][2] = "ACE_TLI_Request_Queue";
className[6][39][3] = "ACE_TLI_Acceptor";
className[6][40][1] = "ACE_TLI_Connector";
className[6][41][1] = "ACE_TLI_Stream";
className[6][42][1] = "ACE_UPipe_Acceptor";
className[6][43][1] = "ACE_UPipe_Connector";
className[6][44][1] = "ACE_UPipe_Stream"; // [Log_Msg]
className[7][1][1] = "ACE_Log_Msg";
className[7][2][1] = "ACE_Log_Record"; // [Memory]
className[8][1][1] = "ACE_Mem_Map";
className[8][2][1] = "ACE_Allocator";
className[8][2][2] = "ACE_Name_Node";
className[8][2][3] = "ACE_Control_Block";
className[8][3][1] = "ACE_Allocator_Adapter";
className[8][3][2] = "ACE_Malloc";
className[8][3][3] = "ACE_Malloc_iterator";
className[8][4][1] = "ACE_Sbrk_Memory_Pool_Options";
className[8][4][2] = "ACE_Sbrk_Memory_Pool";
className[8][4][3] = "ACE_Shared_Memory_Pool_Options";
className[8][4][4] = "ACE_Shared_Memory_Pool";
className[8][4][5] = "ACE_Local_Memory_Pool_Options";
className[8][4][6] = "ACE_Local_Memory_Pool";
className[8][4][7] = "ACE_MMAP_Memory_Pool_Options";
className[8][4][8] = "ACE_MMAP_Memory_Pool";
className[8][4][9] = "ACE_Lite_MMAP_Memory_Pool";
className[8][5][1] = "ACE_Shared_Memory";
className[8][5][1] = "ACE_Shared_Memory_MM";
className[8][7][1] = "ACE_Shared_Memory_V"; // [Misc]
className[9][1][1] = "auto_ptr";
className[9][1][2] = "auto_array_ptr";
className[9][2][1] = "ACE_Dynamic";
className[9][3][1] = "ACE_Get_Opt";
className[9][4][1] = "ACE_Singleton"; // [Name_Service]
className[10][1][1] = "ACE_NS_String";
className[10][1][2] = "ACE_NS_Internal";
className[10][2][1] = "ACE_Name_Space_Map";
className[10][2][2] = "ACE_Local_Name_Space";
className[10][3][1] = "ACE_Name_Proxy";
className[10][4][1] = "ACE_Name_Request";
className[10][4][2] = "ACE_Name_Reply";
className[10][5][1] = "ACE_Name_Binding";
className[10][5][2] = "ACE_Name_Space";
className[10][6][1] = "ACE_Naming_Context";
className[10][6][2] = "ACE_Name_Options";
className[10][7][1] = "ACE_Remote_Name_Space"; // [OS]
className[11][1][1] = "ACE_TSS_Ref";
className[11][1][2] = "ACE_TSS_Info";
className[11][1][3] = "ACE_TSS_Cleanup";
className[11][1][4] = "ACE_Thread_Adapter";
className[11][1][5] = "ACE_OS"; // [Reactor]  
className[12][1][1] = "ACE_Event_Handler";

304
Appendix B1: A Prototype For CO-Toolkits' Design and Implementation Structures: An Overview

className[12][2][1] = "ACE_Event_Handler_T";
className[12][3][1] = "ACE_Handle_Set";
className[12][3][2] = "ACE_Handle_Set_Enumerator";
className[12][4][1] = "ACE_Overlapped_IO";
className[12][4][2] = "ACE_Proactor";
className[12][4][3] = "ACE_Overlapped_File";
className[12][4][4] = "ACE_Reactor_Notify";
className[12][4][5] = "ACE_Reactor_Token";
className[12][4][6] = "ACE_Handler_Repository";
className[12][4][7] = "ACE_Handler_Repository_Enumerator";
className[12][4][8] = "ACE_Reactor";
className[12][5][1] = "ACE_Reactor_Token";
className[12][5][2] = "ACE_Reactor_Token_Notify";
className[12][5][3] = "ACE_Reactor_Ex";
className[12][6][1] = "ACE_Sig_Handlers_Set";
className[12][6][2] = "ACE_Sig_Set";
className[12][6][3] = "ACE_Sig_Action";
className[12][6][4] = "ACE_Sig_Guard";
className[12][6][5] = "ACE_Sig_Handler";
className[12][6][6] = "ACE_Sig_Adapter";
className[12][6][7] = "ACE_Sig_Handlers";
className[12][7][1] = "ACE_Time_Value";
className[12][7][2] = "ACE_Countdown_Time";
className[12][8][1] = "ACE_Timer_Queue"; // [System_V_IPC]
className[13][1][1] = "ACE_SV_Message";
className[13][1][2] = "ACE_SV_Message_Queue";
className[13][2][1] = "ACE_Typed_SV_Message";
className[13][3][1] = "ACE_Typed_SV_Message_Queue";
className[13][4][1] = "ACE_SV_Semaphore_Complex";
className[13][5][1] = "ACE_SV_Semaphore_Simple";
className[13][6][1] = "ACE_SV_Semaphore_Base";
className[13][7][1] = "ACE_SV_Shared_Memory"; // [Timers]
className[14][1][1] = "ACE_Profile_Timer";
className[14][1][2] = "ACE_Profile_Timer";
className[14][2][1] = "ACE_High_Res_Timer"; // [Token_Services]
className[15][1][1] = "ACE_TOKEN_CONST";
className[15][1][2] = "ACE_TPOQ_Entry";
className[15][1][3] = "ACE_TSS_TPOQ_Entry";
className[15][1][4] = "ACE_TPOQ_Reader";
className[15][1][5] = "ACE_Token_Proxy_Queue";
className[15][1][6] = "ACE_Tokens";
className[15][1][7] = "ACE_Local_Mutex";
className[15][1][8] = "ACE_Mutex_Token";
className[15][1][9] = "ACE_RW_Token";
className[15][1][10] = "ACE_Token_Name";
className[15][1][11] = "ACE_Token_Proxy";
className[15][1][12] = "ACE_Null_Token";
className[15][1][13] = "ACE_Local_Mutex";
className[15][1][14] = "ACE_Local_RWLock";
className[15][1][15] = "ACE_Local_RWLock";
className[15][2][1] = "ACE_Token_Name";
className[15][3][1] = "ACE_Remote_Token_Proxy";
className[15][3][2] = "ACE_Remote_Mutex";
className[15][3][3] = "ACE_Remote_RWLock";
className[15][3][4] = "ACE_Remote_RWLock";
className[15][3][5] = "ACE_TSS_Connection";
className[15][4][1] = "ACE_Token_Collection";
className[15][5][1] = "ACE_Token_Manager";
className[15][6][1] = "ACE_Mutex_Invariants";
Appendix B1: A Prototype For CO-Toolkits' Design and Implementation Structures: An Overview

className[15][6][2] = "ACE_RWLock_Invariants";
className[15][6][3] = "ACE_Token_Invariant_Manager";
className[15][7][1] = "ACE_Token_Request";
className[15][7][2] = "ACE_Token_Reply";

// Following I will zero all MPs, MRS, etc. since that it is not easy
// to extract all these details from ACE.
// This example on ACE will check for inter-class inheritance relation

//number of public methods per class per component per framework
for (int i=1; i<=F; i++)
    for (int j=1; j<=C[i]; j++)
        for (int k=1; k<=e[i][j]; k++)
            MP[i][j][k]=0;

//number of protected methods per class per component per framework
for (int i=1; i<=F; i++)
    for (int j=1; j<=C[i]; j++)
        for (int k=1; k<=c[i][j]; k++)
            MR[i][j][k]=0;

//number of private methods per class per component per framework
for (int i=1; i<=F; i++)
    for (int j=1; j<=C[i]; j++)
        for (int k=1; k<=c[i][j]; k++)
            MV[i][j][k]=0;

//number of public objects per class per component per framework
for (int i=1; i<=F; i++)
    for (int j=1; j<=C[i]; j++)
        for (int k=1; k<=c[i][j]; k++)
            OP[i][j][k]=0;

//number of protected objects per class per component per framework
for (int i=1; i<=F; i++)
    for (int j=1; j<=C[i]; j++)
        for (int k=1; k<=c[i][j]; k++)
            OR[i][j][k]=0;

//number of private objects per class per component per framework
for (int i=1; i<=F; i++)
    for (int j=1; j<=C[i]; j++)
        for (int k=1; k<=c[i][j]; k++)
            GV[i][j][k]=0;

306
Appendix B1: A Prototype For CO-Toolkits' Design and Implementation Structures: An Overview

```c
// number of classes inherited in public per class per component per framework
for (int i=1; i<=F; i++){
    for (int j=1; j<=C[i]; j++){
        for (int k=1; k<=c[i][j]; k++){
            MinhP[i][j][k]=0;
        }
    }
}

// number of classes inherited in protected per class per component per framework
for (int i=1; i<=F; i++){
    for (int j=1; j<=C[i]; j++){
        for (int k=1; k<=c[i][j]; k++){
            MinhR[i][j][k]=0;
        }
    }
}

// number of classes inherited in private per class per component per framework
for (int i=1; i<=F; i++){
    for (int j=1; j<=C[i]; j++){
        for (int k=1; k<=c[i][j]; k++){
            MinhV[i][j][k]=0;
        }
    }
}

// number of classes inherited from shared pools in public per class per component per framework
for (int i=1; i<=F; i++){
    for (int j=1; j<=C[i]; j++){
        for (int k=1; k<=c[i][j]; k++){
            MinhPS[i][j][k]=0;
        }
    }
}

// number of classes inherited from shared pools in protected per class per component per framework
for (int i=1; i<=F; i++){
    for (int j=1; j<=C[i]; j++){
        for (int k=1; k<=c[i][j]; k++){
            MinhRS[i][j][k]=0;
        }
    }
}

// number of classes inherited from shared pools in private per class per component per framework
for (int i=1; i<=F; i++){
    for (int j=1; j<=C[i]; j++){
        for (int k=1; k<=c[i][j]; k++){
            MinhVS[i][j][k]=0;
        }
    }
}

// **********************************************

// STEP#2: instantiate for the Frameworks
// objects and the emebded Component, MainSection,
// Class, PublicSection, ProtectedSection, PrivateSection,
// Method, etc. etc. objects
// ALSO: Assign ID values for each object
```

307
// instantiate the framework objects
for (int i=1; i<=F; i++){
    Frameworks[i]=new FrameworkNode();
    Frameworks[i].FrameworkNodeNumber=i;
}

// instantiate the component objects for this new framework object
for (int j=1; j<=C[i]; j++){
    Frameworks[i].Component[j]=new ComponentNode();
    Frameworks[i].Component[j].ID.ComponentNodeNumber=j;
    Frameworks[i].Component[j].ID.FrameworkNodeNumber=i;
    Frameworks[i].Component[j].MainSection="main_"+i+_""+j;
    // instantiate the class objects for this component object
    for (int k=1; k<=c[i][j]; k++){
        Frameworks[i].Component[j].Class[k]=new ClassNode();
        Frameworks[i].Component[j].Class[k].ID.ClassNodeNumber=k;
        Frameworks[i].Component[j].Class[k].ID.Name=className[i][j][k];
        Frameworks[i].Component[j].Class[k].ID.ParentComponent.ComponentNodeNumber=j;
        Frameworks[i].Component[j].Class[k].ID.ParentClass.FrameworkNodeNumber=i;
    }
    // instantiate for the public section of this class
    Frameworks[i].Component[j].Class[k].PublicSection=new PublicSectionNode();
    for (int l=1; l<=MP[i][j][k]; l++){
        Frameworks[i].Component[j].Class[k].PublicSection.Method[l].ReturnedValueTypo="void";
        Frameworks[i].Component[j].Class[k].PublicSection.Method[l].ID.MethodNodeNumber=l;
    }
    // instantiate for the protected section of this class
    Frameworks[i].Component[j].Class[k].ProtectedSection=new ProtectedSectionNode();
    for (int l=1; l<=MR[i][j][k]; l++){
        Frameworks[i].Component[j].Class[k].ProtectedSection.Method[l].ReturnedValueTypo="void";
        Frameworks[i].Component[j].Class[k].ProtectedSection.Method[l].ID.MethodNodeNumber=l;
    }
    // instantiate for the private section of this class
    Frameworks[i].Component[j].Class[k].PrivateSection=new PrivateSectionNode();
    for (int l=1; l<=MV[i][j][k]; l++){
        Frameworks[i].Component[j].Class[k].PrivateSection.Method[l].ReturnedValueTypo="void";
    }
}

308
Appendix B1: A Prototype For CO-Toolkits' Design and Implementation Structures: An Overview

// STEP#4: read the inheritance references (ClassIDs) for each class

// **********************************************************************************************

// {AX} NONE// {CORBA} // class ACE_ST_CORBA_Handler

    --> [2][1][1]


    Minhp[2][1][2]=1; // class ACE_MT_CORBA_Handler

    --> [2][1][1]


    Minhp[2][1][3]=1;

    // {Collections} NONE// {Concurrency}

    // class ACE_RW_Process_Mutex  --> [4][4][6]


    Frameworks[4].Component[4].Class[7].PublicInheritedClasses[1].ParentComponent.Type="Class7"

    Minhp[4][4][7]=1; // class ACE_Manual_Event  --> [4][4][12]


    Minhp[4][4][13]=1; // class ACE_Auto_Event  --> [4][4][12]


    Minhp[4][4][14]=1; // class ACE_RW_Thread_Mutex

    --> [4][4][4]


    Minhp[4][4][19]=1; // class ACE_Thread_Semaphore

    --> [4][4][2]

MinhP[6][2][1]=1;

// class ACE_DEV_Connector --> [6][2][1]
Frameworks[6].Component[3].Class[1].PublicInheritedClasses[1].ClassNumber=1;
MinhP[6][3][1]=1;

// class ACE_DEV_IO --> [6][2][1]
Frameworks[6].Component[4].Class[1].PublicInheritedClasses[1].ClassNumber=1;
MinhP[6][4][1]=1; // class ACE_FILE --> [6][1][1]
Frameworks[6].Component[5].Class[1].PublicInheritedClasses[1].ClassNumber=1;
MinhP[6][5][1]=1; // class ACE_FILE_Connector --> [6][5][1]
Frameworks[6].Component[6].Class[1].PublicInheritedClasses[1].ClassNumber=1;
MinhP[6][6][1]=1; // class ACE_FILE_IO --> [6][5][1]
Frameworks[6].Component[7].Class[1].PublicInheritedClasses[1].ClassNumber=1;
MinhP[6][7][1]=1; // class ACE_DEV.Addr --> [6][9][1]
Frameworks[6].Component[10].Class[1].PublicInheritedClasses[1].ClassNumber=1;
MinhP[6][10][1]=1; // class ACE_FILE.Addr --> [6][9][1]
MinhP[6][11][1]=1; // class ACE_INET.Addr --> [6][9][1]
Frameworks[6].Component[12].Class[1].PublicInheritedClasses[1].ClassNumber=1;
Appendix B1: A Prototype For CO-Toolkits' Design and Implementation Structures: An Overview

MinhP[6][12][1]=1; // class ACE_PIPE.Addr ——> [6][9][1]
Frameworks[6].Component[13].Class[1].PublicInheritedClasses[1].ClassNodeNumber=1;
MinhP[6][13][1]=1; // class ACE_UNIX.Addr ——> [6][9][1]
Frameworks[6].Component[14].Class[1].PublicInheritedClasses[1].ClassNodeNumber=1;
MinhP[6][14][1]=1; // class ACE_FIFO ——> [6][8][1]
Frameworks[6].Component[15].Class[1].PublicInheritedClasses[1].ClassNodeNumber=1;
MinhP[6][15][1]=1; // class ACE_FIFO_Recv ——> [6][15][1]
Frameworks[6].Component[16].Class[1].PublicInheritedClasses[1].ClassNodeNumber=1;
MinhP[6][16][1]=1; // class ACE_FIFO_Recv_Msg
—> [6][16][1]
Frameworks[6].Component[17].Class[1].PublicInheritedClasses[1].ClassNodeNumber=1;
MinhP[6][17][1]=1; // class ACE_FIFO_Send ——> [6][15][1]
Frameworks[6].Component[18].Class[1].PublicInheritedClasses[1].ClassNodeNumber=1;
MinhP[6][18][1]=1; // class ACE_FIFO_Send_Msg
—> [6][18][1]
Frameworks[6].Component[19].Class[1].PublicInheritedClasses[1].ClassNodeNumber=1;
MinhP[6][19][1]=1; // class ACE_LSOCK_Acceptor
—> [6][26][1]

312
null
Frameworks[6].Component[25].Class[1].PublicInheritedClasses[1].ClassName='ACE_SOCK_Acceptor'

Frameworks[6].Component[26].Class[1].PublicInheritedClasses[1].ClassName='new ClassID();'

Frameworks[6].Component[26].Class[1].PublicInheritedClasses[1].ClassName='new ClassID();'

Frameworks[6].Component[26].Class[1].PublicInheritedClasses[1].ClassName='ACE_SOCK_Codgram'

Frameworks[6].Component[27].Class[1].PublicInheritedClasses[1].ClassName='new ClassID();'

Frameworks[6].Component[27].Class[1].PublicInheritedClasses[1].ClassName='new ClassID();'

Frameworks[6].Component[27].Class[1].PublicInheritedClasses[1].ClassName='ACE_SOCK_Connector'

Frameworks[6].Component[28].Class[1].PublicInheritedClasses[1].ClassName='new ClassID();'

Frameworks[6].Component[28].Class[1].PublicInheritedClasses[1].ClassName='new ClassID();'

Frameworks[6].Component[28].Class[1].PublicInheritedClasses[1].ClassName='ACE_SOCK_Dgram'

Frameworks[6].Component[28].Class[1].PublicInheritedClasses[1].ClassName='new ClassID();'

Frameworks[6].Component[28].Class[1].PublicInheritedClasses[1].ClassName='new ClassID();'

Frameworks[6].Component[29].Class[1].PublicInheritedClasses[1].ClassName='ACE_SOCK_Dgram_Bcast'

Frameworks[6].Component[30].Class[1].PublicInheritedClasses[1].ClassName='new ClassID();'

Frameworks[6].Component[30].Class[1].PublicInheritedClasses[1].ClassName='new ClassID();'

Frameworks[6].Component[30].Class[1].PublicInheritedClasses[1].ClassName='ACE_SOCK_Dgram_Mcast'

Frameworks[6].Component[31].Class[1].PublicInheritedClasses[1].ClassName='new ClassID();'

Frameworks[6].Component[31].Class[1].PublicInheritedClasses[1].ClassName='new ClassID();'

Frameworks[6].Component[31].Class[1].PublicInheritedClasses[1].ClassName='ACE_SOCK_IO'
number = 25;
MinhP[6][32][1] = 1; // 6–33–1 class ACE_SOCK_Stream

Frameworks[6].Component[33].Class[1].PublicInheritedClasses[1].ClassName = "Stream";
MinhP[6][33][1] = 1; // 6–34–1 class ACE_SPIPE

Frameworks[6].Component[34].Class[1].PublicInheritedClasses[1].ClassName = "Pipe";
MinhP[6][34][1] = 1; // 6–35–1 class ACE_SPIPE_Acceptor

Frameworks[6].Component[35].Class[1].PublicInheritedClasses[1].ClassName = "Acceptor";
MinhP[6][35][1] = 1; // 6–36–1 class ACE_SPIPE_Connector

Frameworks[6].Component[36].Class[1].PublicInheritedClasses[1].ClassName = "Connector";
MinhP[6][36][1] = 1; // 6–37–1 class ACE_SPIPE_Stream

Frameworks[6].Component[37].Class[1].PublicInheritedClasses[1].ClassName = "Stream";
MinhP[6][37][1] = 1; // 6–38–1 class ACE_TLI

Frameworks[6].Component[38].Class[1].PublicInheritedClasses[1].ClassName = "TLI";
MinhP[6][38][1] = 1; // 6–39–3 class ACE_TLI_Acceptor


MinhP[6][39][1] = 1; // 6–39–3 class ACE_TLI_Acceptor

Frameworks[6].Component[39].Class[3].PublicInheritedClasses[1].ClassName = "TLI";
Appendix B1: A Prototype For CO-Toolkits’ Design and Implementation Structures: An Overview

Minhp[6][39][3]=1; //6-40-1  class ACE_TLI_Connector

Frameworks[6].Component[40].Class[1].PublicInheritedClasses[1].ClassName=1;
Minhp[6][40][1]=1; //6-41-1  class ACE_TLI_Stream

Frameworks[6].Component[41].Class[1].PublicInheritedClasses[1].ClassName=1;
Minhp[6][41][1]=1; //6-42-1  class ACE_UPipe_Acceptor

Frameworks[6].Component[42].Class[1].PublicInheritedClasses[1].ClassName=1;
Minhp[6][42][1]=1; //6-43-1  class ACE_UPipe_Connector

Frameworks[6].Component[43].Class[1].PublicInheritedClasses[1].ClassName=1;
Minhp[6][43][1]=1; //6-44-1  class ACE_UPipe_Stream

Frameworks[6].Component[44].Class[1].PublicInheritedClasses[1].ClassName=1;
Minhp[6][44][1]=1;

// [Log_Msg] NONE// [Memory]
//8-4-4  class ACE_Shared_Memory_Pool  ----> [12][1][1]
Frameworks[8].Component[4].Class[4].PublicInheritedClasses[1].ClassName=1;
Minhp[8][4][4][4]=1; //8-4-8  class ACE_MMAP_Memory_Pool

Frameworks[8].Component[4].Class[8].PublicInheritedClasses[1].ClassName=1;
Minhp[8][4][4][8]=1; //8-4-9  class ACE_Lite_MMAP_Memory_Pool ----> [8][4][9]

316
Appendix B1: A Prototype For CO-Toolkits' Design and Implementation Structures: An Overview

Frameworks[8].Component[4].Class[9].PublicInheritedClasses[1].ClassNameNumber=9;
MinhP[8][8][4][9]=1; //8-6-1 class ACE_Shared_Memory_MM

Frameworks[8].Component[6].Class[1].PublicInheritedClasses[1].ClassNameNumber=1;
MinhP[8][8][6][1]=1; //8-7-1 class ACE_Shared_Memory_SV

Frameworks[8].Component[7].Class[1].PublicInheritedClasses[1].ClassNameNumber=1;
MinhP[8][8][7][1]=1;
// [Misc] NONE// [Name_Service]
//10-2-2 class ACE_Local_Name_Space --> [10][5][2]
Frameworks[10].Component[2].Class[2].PublicInheritedClasses[1].ClassNameNumber=2;
MinhP[10][2][2]=1; //10-3-1 class ACE_Name_Proxy

Frameworks[10].Component[3].Class[1].PublicInheritedClasses[1].ClassNameNumber=1;
MinhP[10][3][1]=1; //10-7-1 class ACE_Remote_Name_Space

Frameworks[10].Component[7].Class[1].PublicInheritedClasses[1].ClassNameNumber=2;
MinhP[10][7][1]=1;
// [OS] NONE// [Reactor]
//12-2-1 class ACE_Event_Handler_T --> [12][1][1]
Frameworks[12].Component[2].Class[1].PublicInheritedClasses[1].ClassNameNumber=1;
MinhP[12][2][1]=1; //12-4-2 class ACE_Proactor

317
Frameworks[12].Component[4].Class[2].PublicInheritedClasses[1].ClassNameNumber=1;
MinhP[12][4][2]=1; //12-4-4 class ACE_Reactor_Notify
        --> [12][1][1]
Frameworks[12].Component[4].Class[4].PublicInheritedClasses[1].ClassNameNumber=1;
MinhP[12][4][4]=1; //12-5-2 class ACE_ReactorEx_Notify
        --> [12][1][1]
Frameworks[12].Component[5].Class[2].PublicInheritedClasses[1].ClassNameNumber=1;
MinhP[12][5][2]=1; //12-6-6 class ACE_Sig_Adapter
        --> [12][1][1]
Frameworks[12].Component[6].Class[6].PublicInheritedClasses[1].ClassNameNumber=1;
MinhP[12][6][6]=1; //12-6-7 class ACE_Sig_Handlers
        --> [12][6][5]
Frameworks[12].Component[6].Class[7].PublicInheritedClasses[1]= new ClassID();
Frameworks[12].Component[6].Class[7].PublicInheritedClasses[1].ClassNameNumber=5;
MinhP[12][6][7]=1;
// [System_V_IPC]
//13-5-1 class ACE_SV_Semaphore_Complex --> private [13][6][1]
Frameworks[13].Component[5].Class[1].PublicInheritedClasses[1].ClassNameNumber=1;
MinhP[13][5][1]=1;
// [Timers] NONE/ [Token_Services]
//15-1-8 class ACE_Mutex_Token --> [15][1][6]
Frameworks[15].Component[1].Class[8].PublicInheritedClasses[1].ClassNameNumber=6;
MinhP[15][1][8]=1; //15-1-9 class ACE_RW_Token
        --> [15][1][6]

318
Frameworks[15].Component[1].Class[9].PublicInheritedClasses[1].ClassName=ClassNodeNumber=6;
MinhP[15][1][10]=1; //15-1-12 class ACE_Null_Token

Frameworks[15].Component[1].Class[10].PublicInheritedClasses[1].ClassName=ClassNodeNumber=11;
MinhP[15][1][10]=1; //15-1-13 class ACE_Local_Mutex

MinhP[15][1][13]=1; //15-1-14 class ACE_Local_RLock

Frameworks[15].Component[1].Class[14].PublicInheritedClasses[1].ClassName=ClassNodeNumber=11;
MinhP[15][1][14]=1; //15-1-15 class ACE_Local_WLock

Frameworks[15].Component[1].Class[15].PublicInheritedClasses[1].ClassName=ClassNodeNumber=11;
MinhP[15][1][15]=1; //15-3-1 class ACE_Remote_Token_Proxy

MinhP[15][3][1]=1; //15-3-2 class ACE_Remote_Mutex

Frameworks[15].Component[3].Class[2].PublicInheritedClasses[1].ClassName=ClassNodeNumber=1;
MinhP[15][3][2]=1; //15-3-3 class ACE_Remote_RLock

Frameworks[15].Component[3].Class[3].PublicInheritedClasses[1].ClassName=ClassNodeNumber=1;
protected static void initializeSharedPools_Example_10 {
    // **************************************
    // STEP#1 read values for:
    //   F, C[], c[1].
    //   MIP, MR[], MV[],
    //   OP[], OR[], OV[],
    // **************************************

    // number of shared pools
    S=2;
    // number of shared classes per shared pool
    SC[1]=2;
    SC[2]=2;
    // number of public methods per class per shared pool
    SMP[1][1]=2; SMP[1][2]=2;
    SMP[2][1]=2; SMP[2][2]=2;
    // number of protected methods per class per shared pool
    SMR[1][1]=1; SMR[1][2]=1;
    SMR[2][1]=1; SMR[2][2]=1;
    // number of private methods per class per shared pool
    SMV[1][1]=1; SMV[1][2]=1;
    SMV[2][1]=1; SMV[2][2]=1;
    // number of public objects per class per shared pool
    SOP[1][1]=0; SOP[1][2]=1;
    SOP[2][1]=1; SOP[2][2]=0;
    // number of protected methods per class per shared pool
    SOR[1][1]=2; SOR[1][2]=2;
    SOR[2][1]=2; SOR[2][2]=2;
    // number of private methods per class per shared pool
    SOV[1][1]=4; SOV[1][2]=4;
    SOV[2][1]=4; SOV[2][2]=4;
    // number of classes inherited in public per class per shared pool
    SMinhP[1][1]=1; SMinhP[1][2]=0;
    SMinhP[2][1]=1; SMinhP[2][2]=0;
    // number of classes inherited in protected per class per shared pool
    SMinhR[1][1]=0; SMinhR[1][2]=0;
    SMinhR[2][1]=0; SMinhR[2][2]=0;
}

320
Appendix B1: A Prototype For CO-Toolkits' Design and Implementation Structures: An Overview

//number of classes inherited in private per class per shared pool
SMinhV[1][1]=0; SMiNHV[1][2]=0;
SMiNHV[2][1]=0; SMiNHV[2][2]=0;  // number of classes inherited in public from frameworks per class per
// shared pool
SMiNHPF[1][1]=0; SMiNHPF[1][2]=0;
SMiNHPF[2][1]=0; SMiNHPF[2][2]=0;
// number of classes inherited in protected from frameworks per class per shared pool
SMiNHRF[1][1]=0; SMiNHRF[1][2]=0;
SMiNHRF[2][1]=0; SMiNHRF[2][2]=0;
// number of classes inherited in private from frameworks per class per shared pool
SMiNHVF[1][1]=0; SMiNHVF[1][2]=0;
SMiNHVF[2][1]=0; SMiNHVF[2][2]=0;

// STEP#2: instantiate for the Frameworks
// objects and the embedded
// Class, PublicSection, ProtectedSection, PrivateSection,
// Method, etc. etc. objects
// ALSO: Assign ID values for each object

// instantiate the framework objects
for (int i=1; i<=5; i++){
    SharedPools[i]=new SharedPoolNode();
    SharedPools[i].SharedPoolNodeNumber=i;
    // instantiate the shared classes objects for this new shared pool object
    for (int j=1; j<=SC[i]; j++){
        SharedPools[i].Class[j]=new SharedClassNode();
        SharedPools[i].Class[j].ClassName="s_class"+i+j;
        SharedPools[i].Class[j].ID.ClassNodeNumber=j;
        SharedPools[i].Class[j].ID.ParentSharedPoolNodeNumber=i;
        // instantiate for the public section of this class
        SharedPools[i].Class[j].PublicSection=new SharedPublicSectionNode();
        for (int k=1; k<=SMP[i][j]; k++){
            SharedPools[i].Class[j].PublicSection.Method[k]=new SharedMethodNode();
            SharedPools[i].Class[j].PublicSection.Method[k].ReturnedValueType="void";
            SharedPools[i].Class[j].PublicSection.Method[k].ID.MethodNodeNumber=k;
            SharedPools[i].Class[j].PublicSection.Method[k].ID.ParentSharedClass.ClassNodeNumber=j;
            SharedPools[i].Class[j].PublicSection.Method[k].ID.ParentSharedClass.ParentSharedPoolNodeNumber=i;
        }
        // instantiate for the protected section of this class
        SharedPools[i].Class[j].ProtectedSection=new SharedProtectedSectionNode();
        for (int k=1; k<=SMR[i][j]; k++){
            SharedPools[i].Class[j].ProtectedSection.Method[k]=new SharedMethodNode();
            SharedPools[i].Class[j].ProtectedSection.Method[k].ReturnedValueType="void";
            SharedPools[i].Class[j].ProtectedSection.Method[k].ID.MethodNodeNumber=k;
            SharedPools[i].Class[j].ProtectedSection.Method[k].ID.ParentSharedClass.ClassNodeNumber=j;
            SharedPools[i].Class[j].ProtectedSection.Method[k].ID.ParentSharedClass.ParentSharedPoolNodeNumber=i;
        }
        // instantiate for the private section of this class
        SharedPools[i].Class[j].PrivateSection=new SharedPrivateSectionNode();
        for (int k=1; k<=SMV[i][j]; k++){
            SharedPools[i].Class[j].PrivateSection.Method[k]=new SharedMethodNode();
            SharedPools[i].Class[j].PrivateSection.Method[k].ReturnedValueType="void";
            SharedPools[i].Class[j].PrivateSection.Method[k].ID.MethodNodeNumber=k;
            SharedPools[i].Class[j].PrivateSection.Method[k].ID.ParentSharedClass.ClassNodeNumber=k;
            SharedPools[i].Class[j].PrivateSection.Method[k].ID.ParentSharedClass.ParentSharedPoolNodeNumber=i;
        }
    }
}
// STEP#3: Read the entries for each of these objects
// **************************************************************
// SharedCLASS(1,1)
SharedPools[1].Class[1].PublicSection.Method[1].MethodName = "s_p_mthd111";
SharedPools[1].Class[1].PublicSection.Method[1].ARGS_IN_ArgType[1] = "s_class12";
SharedPools[1].Class[1].PublicSection.Method[2].MethodName = "s_p_mthd112";
SharedPools[1].Class[1].ProtectedSection.Method[1].MethodName = "s_r_mthd111";
SharedPools[1].Class[1].ProtectedSection.Object[1] = "s_r_obj112";
SharedPools[1].Class[1].ProtectedSection.ObjectType[1] = "class12";

// SharedCLASS(1,2)
// SharedPools[1].Class[1].PublicSection.Method[1].MethodName = "s_p_mthd111";
// SharedPools[1].Class[1].PublicSection.Method[1].ARGS_IN_ArgType[1] = "s_class12";
// SharedPools[1].Class[1].PublicSection.Method[2].MethodName = "s_p_mthd112";
// SharedPools[1].Class[1].ProtectedSection.Method[1].MethodName = "s_r_mthd111";

// SHAREDCLASS(1,2)
// SharedPools[1].Class[1].PrivateSection.Method[1].MethodName = "s_v_mthd111";
// SharedPools[1].Class[1].PrivateSection.ObjectType[1] = "s_class21";
Appendix B1: A Prototype For CO-Toolkits' Design and Implementation Structures: An Overview

SharedPools[1].Class[2].PrivateSection.Method[1].MethodName="s_v_mthd121”;
SharedPools[1].Class[2].PrivateSection.ObjectType[1]="s_class1”;

//SharedCLASS(2,1)
SharedPools[2].Class[1].PublicSection.Method[1].MethodName="s_p_mthd211”;
SharedPools[2].Class[1].PublicSection.Method[1].ReturnValueType="T8”;

SharedPools[2].Class[1].PublicSection.Method[2].MethodName="s_p_mthd212”;
SharedPools[2].Class[1].ProtectedSection.Method[1].MethodName="s_r_mthd211”;
SharedPools[2].Class[1].ProtectedSection.Method[1].ReturnValueType="T1”;
SharedPools[2].Class[1].ProtectedSection.ObjectType[1]="s_class12”;
SharedPools[2].Class[1].PrivateSection.Method[1].MethodName="s_v_mthd211”;
SharedPools[2].Class[1].PrivateSection.ObjectType[1]="s_class1”;

//SharedCLASS(2,2)
SharedPools[2].Class[2].PublicSection.Method[1].MethodName="s_p_mthd221”;
SharedPools[2].Class[2].ProtectedSection.Method[1].MethodName="s_r_mthd221”;
SharedPools[2].Class[2].PrivateSection.Method[1].MethodName="s_v_mthd221”;

323
Appendix B1: A Prototype For CO-Toolkits’ Design and Implementation Structures: An Overview

// STEP#4: read in the inheritance references (ClassIDs) for each class

// ***********************************************
//SharedClass(1,1) inherits from two other classes in public —
SharedPools[1].Class[1].PublicInheritedSharedClasses[1].ClassNodeNumber=2;
SharedPools[1].Class[1].PublicInheritedSharedClasses[1].ParentSharedPoolNodeNumber=1;//Class(2,1) inherits only from one other class in public

SharedPools[2].Class[1].PublicInheritedSharedClasses[1].ClassNodeNumber=2;

protected static void initializeSharedPools_Example_20 {

    // ***********************************************
    // STEP#1 read values for:
    //   F, C[](c[]),
    //   MP[][[]], MR[][[]], MV[][[]]
    //   OP[][[]], OR[][[]], OV[][[]]

    // number of shared pools
    S=2;
    // number of shared classes per shared pool
    SC[1]=2;
    SC[2]=2;
    // number of public methods per class per shared pool
    SMP[1][1]=2; SMP[1][2]=2;
    SMP[2][1]=2; SMP[2][2]=2;
    // number of protected methods per class per shared pool
    SMR[1][1]=1; SMR[1][2]=1;
    SMR[2][1]=1; SMR[2][2]=1;
    // number of private methods per class per shared pool
    SMV[1][1]=1; SMV[1][2]=1;
    SMV[2][1]=1; SMV[2][2]=1;
    // number of public objects per class per shared pool
    SOP[1][1]=0; SOP[1][2]=1;
    SOP[2][1]=1; SOP[2][2]=0;
    // number of protected methods per class per shared pool
    SOR[1][1]=2; SOR[1][2]=2;
    SOR[2][1]=2; SOR[2][2]=2;
    // number of private methods per class per shared pool
    SOV[1][1]=4; SOV[1][2]=4;
    SOV[2][1]=4; SOV[2][2]=4;
    // number of classes inherited in public per class per shared pool
    SMinhP[1][1]=1; SMinhP[1][2]=0;
    SMinhP[2][1]=1; SMinhP[2][2]=0;
    // number of classes inherited in protected per class per shared pool
    SMinhR[1][1]=0; SMinhR[1][2]=0;
    SMinhR[2][1]=0; SMinhR[2][2]=1;
    // number of classes inherited in private per class per shared pool
    SMinhV[1][1]=1; SMinhV[1][2]=0;
    SMinhV[2][1]=0; SMinhV[2][2]=0;//number of classes inherited in public from frameworks per class per shared pool
    SMinhPF[1][1]=1; SMinhPF[1][2]=0;
    SMinhPF[2][1]=0; SMinhPF[2][2]=0;
    // number of classes inherited in protected from frameworks per class per shared pool
    SMinhRF[1][1]=0; SMinhRF[1][2]=0;
    SMinhRF[2][1]=0; SMinhRF[2][2]=1;
    // number of classes inherited in private from frameworks per class per shared pool
    SMinhVF[1][1]=0; SMinhVF[1][2]=0;

    324
SMinhVF[2][1]=0; SMinhVF[2][2]=0;
*****************************************************************************

// STEP#2: instantiate for the Frameworks
// objects and the embedded
// Class, PublicSection, ProtectedSection, PrivateSection,
// Method, etc. etc. objects
// ALSO: Assign ID values for each object
*****************************************************************************

// instantiate the framework objects
for (int i=1; i<=S; i++) {
    SharedPools[i]=new SharedPoolNode();
    SharedPools[i].SharedPoolNodeNumber=i;
    // instantiate the shared classes objects for this new shared pool object
    for (int j=1; j<=SC[i]; j++) {
        SharedPools[i].Class[j]=new SharedClassNode();
        SharedPools[i].Class[j].ClassName="s_class"+i+j;
        SharedPools[i].Class[j].ID.ClassNodeNumber=j;
        SharedPools[i].Class[j].ID.ParentSharedPoolNodeNumber=i;
        // instantiate for the public section of this class
        SharedPools[i].Class[j].PublicSection=new SharedPublicSectionNode();
        for (int k=1; k<=SMP[i][j]; k++) {
            SharedPools[i].Class[j].PublicSection.Method[k]=new SharedMethodNode();
            SharedPools[i].Class[j].PublicSection.Method[k].ReturnedValueTypetype=void"
            SharedPools[i].Class[j].PublicSection.Method[k].ID.MethodNodeNumber=k;
            SharedPools[i].Class[j].PublicSection.Method[k].ID.ParentSharedClass.ClassNodeNumber=j;
            SharedPools[i].Class[j].PublicSection.Method[k].ID.ParentSharedClass.ParentSharedPoolNodeNumber=i;
        }
        // instantiate for the protected section of this class
        SharedPools[i].Class[j].ProtectedSection=new SharedProtectedSectionNode();
        for (int k=1; k<=SMR[i][j]; k++) {
            SharedPools[i].Class[j].ProtectedSection.Method[k]=new SharedMethodNode();
            SharedPools[i].Class[j].ProtectedSection.Method[k].ReturnedValueTypetype=void"
            SharedPools[i].Class[j].ProtectedSection.Method[k].ID.MethodNodeNumber=k;
            SharedPools[i].Class[j].ProtectedSection.Method[k].ID.ParentSharedClass.ClassNodeNumber=j;
            SharedPools[i].Class[j].ProtectedSection.Method[k].ID.ParentSharedClass.ParentSharedPoolNodeNumber=i;
        }
        // instantiate for the private section of this class
        SharedPools[i].Class[j].PrivateSection=new SharedPrivateSectionNode();
        for (int k=1; k<=SMV[i][j]; k++) {
            SharedPools[i].Class[j].PrivateSection.Method[k]=new SharedMethodNode();
            SharedPools[i].Class[j].PrivateSection.Method[k].ReturnedValueTypetype=void"
            SharedPools[i].Class[j].PrivateSection.Method[k].ID.MethodNodeNumber=k;
            SharedPools[i].Class[j].PrivateSection.Method[k].ID.ParentSharedClass.ClassNodeNumber=k;
            SharedPools[i].Class[j].PrivateSection.Method[k].ID.ParentSharedClass.ParentSharedPoolNodeNumber=i;
        }
    }
}

*****************************************************************************

// STEP#3: Read the entries for each of these objects
*****************************************************************************

//SharedClass(1,1) public section contains only two methods and no objects
//SMP[1][1]=2 & & SOP[1][1]=0
    SharedPools[1].Class[1].PublicSection.Method[1].MethodName="s_p_mthd111";
    //get in ARGs for this method

325
Appendix B1: A Prototype For CO-Toolkits’ Design and Implementation Structures: An Overview

SharedPools[1].Class[1].PublicSection.Method[1].ARGS.IN_ArgType[1]="class112";
SharedPools[1].Class[1].PublicSection.Method[1].ReturnedValueType="T5";

//-------------
SharedPools[1].Class[1].PublicSection.Method[2].MethodName="s_p_mthd112";
//get in ARGS for this method
SharedPools[1].Class[1].PublicSection.Method[2].ReturnedValueType="T7";//SharedClass(1,1) protected section contains one method and 2 objects
//SMR[1][1]=1 & SOR[1][1]=2
SharedPools[1].Class[1].ProtectedSection.Method[1].MethodName="s_r_mthd111";
//get in ARGS for this method

//-------------
SharedPools[1].Class[1].ProtectedSection.Object[1]="s_r_obj112";
SharedPools[1].Class[1].ProtectedSection.Object[1]="class112";

//SharedClass(1,1) private section contains 1 method and 4 objects
//SMV[1][1]=1 & S0V[1][1]=4
SharedPools[1].Class[1].PrivateSection.Method[1].MethodName="s_v_mthd111";
//ARGS for this method
SharedPools[1].Class[1].PrivateSection.Method[1].ARGS.OUT_ArgType[1]="s_class22";

//-------------

//SharedCLASS(1,2)
//SharedClass(1,2) public section contains 2 methods and 1 object === VIOLATION
//get ARGS

//-------------
//get ARGS

//-------------
326
Appendix B1: A Prototype For CO-Toolkits’ Design and Implementation Structures: An Overview

SharedPools[1].Class[2].PublicSection.ObjectType[1]="class212="/SharedClass(1,2) protected section contains 1 method and 2 objects
//SMR[1][2]=1 & SOR[1][2]=2
SharedPools[1].Class[2].ProtectedSection.Method[1].MethodName="s_r_mthd121";
//get ARGS
SharedPools[1].Class[2].ProtectedSection.Method[1].ReturnedValueType="T4";

//
SharedPools[1].Class[2].ProtectedSection.ObjectType[1]="s_class12";
SharedPools[1].Class[2].ProtectedSection.ObjectType[2]="T5="/SharedClass(1,2) private section contains 1 method and 4 objects
//SMV[1][2]=1 & SOV[1][2]=4
SharedPools[1].Class[2].PrivateSection.Method[1].MethodName="s_v_mthd121";
//get ARGS

//

//SharedCLASS(2,1)
//SharedClass(2,1) public section contains 2 methods and 1 object — VIOLATION
//SMP[2][1]=2 & SOP[2][1]=1
SharedPools[2].Class[1].PublicSection.Method[1].MethodName="s_p_mthd211";
//get ARGS
SharedPools[2].Class[1].PublicSection.Method[1].ReturnedValueType="T8";

//
//get ARGS

//
//SMR[2][1]=1 & SOR[2][1]=2
SharedPools[2].Class[1].ProtectedSection.Method[1].MethodName="s_r_mthd211";
//get ARGS
SharedPools[2].Class[1].ProtectedSection.Method[1].ReturnedValueType="T1";

//
SharedPools[2].Class[1].ProtectedSection.ObjectType[1]="s_class12";
//SharedClass(2,1) private section contains 1 method and 4 objects
//MV[2][1][1]=1 & OV[2][1][1]=4
SharedPools[2].Class[1].PrivateSection.Method[1].MethodName="s_v_mthd211";
//get ARGS

//
SharedPools[2].Class[1].PrivateSection.ObjectType[1]="s_class21";

327
Appendix B1: A Prototype For CO-Toolkits’ Design and Implementation Structures: An Overview

//SharedCLASS(2,2)
//SharedClass(2,2) public section contains 2 methods and no object
//SMP[2][2]=2 && SOP[2][2]=0
SharedPools[2].Class[2].PublicSection.Method[1].MethodName="s_p_mthd221";
// get ARG
//____________________________________________________________________

// get ARG
//____________________________________________________________________

//SharedClass(2,2) protected section contains 1 method and 2 objects
SharedPools[2].Class[2].ProtectedSection.Method[1].MethodName="s_r_mthd221";
// get ARG
//____________________________________________________________________

//SharedClass(2,2) private section contains 1 method and 4 objects
SharedPools[2].Class[2].PrivateSection.Method[1].MethodName="s_v_mthd221";
// get ARG
//____________________________________________________________________

//STEP#4: read in the inheritance references (ClassIDs) for each class
//____________________________________________________________________

//SharedClass(1,1) inherits from two other classes in public
SharedPools[1].Class[1].PublicInheritedSharedClasses[1].ClassName=1;
SharedPools[1].Class[1].PublicInheritedSharedClasses[1].ParentSharedPoolNodeNumber=2;
SharedPools[1].Class[1].PublicInheritedSharedClasses[1].ParentSharedPoolNodeNumber=2;

//Class(1,1) inherits from another class in private — Violation — non-public inheritance
SharedPools[1].Class[1].PrivateInheritedSharedClasses[1].ClassName=1;

// Class(1,2) does not inherit from any other class/Class(2,1) inherits only from one other class in public

328
SharedPools[2].Class[1].PublicInheritedSharedClasses[1].ClassName=1;
SharedPools[2].Class[1].PublicInheritedSharedClasses[1].ParentSharedPoolNodeNumber=2;//Class(2,2) inherits only from one another class in protected
SharedPools[2].Class[2].ProtectedInheritedSharedClasses[1].ClassName=2;
SharedPools[2].Class[2].ProtectedInheritedSharedClasses[1].ParentSharedPoolNodeNumber=2;//Class(2,2) inherits another class in protected from frameworks
SharedPools[2].Class[2].ProtectedInheritedClasses[1].ClassName=1;
}

protected static void initializeSharedPools_Example_3_ACE0()
{
    //**************************************************************************
    // STEP #1 read values for:
    // F, C[], c[],
    // MP[], MR[], MV[]
    // OP[], OR[], OV[]
    //**************************************************************************
    //number of shared pools
    S=1;
    //number of shared classes per shared pool
    SC[1]=1; //number of public methods per class per shared pool
    SMP[1][1]=0;
    //number of protected methods per class per shared pool
    SMR[1][1]=0;
    //number of private methods per class per shared pool
    SMV[1][1]=0;
    //number of public objects per class per shared pool
    SOP[1][1]=0;
    //number of protected methods per class per shared pool
    SOR[1][1]=0;
    //number of private methods per class per shared pool
    SOV[1][1]=0;
    //number of classes inherited in public per class per shared pool
    SMinkP[1][1]=0;
    //number of classes inherited in protected per class per shared pool
    SMinkR[1][1]=0;
    //number of classes inherited in private per class per shared pool
    SMinkV[1][1]=0;
    //number of classes inherited in public from frameworks per class per shared pool
    SMinkFP[1][1]=0;
    //number of classes inherited in protected from frameworks per class per shared pool
    SMinkFP[1][1]=0;
    //number of classes inherited in private from frameworks per class per shared pool
    SMinkVF[1][1]=0;//**************************************************************************
    // STEP #2: instantiate for the Frameworks
    // objects and the embedded
    // Class, PublicSection, ProtectedSection, PrivateSection,
    // Method, etc. etc. objects
    // ALSO: Assign ID values for each object
    //**************************************************************************
    for (int i=1; i<=S; i++)
    {
        SharedPools[i]=new SharedPoolNode();
        SharedPools[i].SharedPoolNodeNumber=i;
        // instantiate the shared classes objects for this new shared pool object
        for (int j=1; j<=SC[i]; j++)
        {
            SharedPools[i].Class[j]=new SharedClassName();
        }
    }

    //**************************************************************************
SharedPools[i].Class[j].ClassName="dummy_shared_class"+i+j;
SharedPools[i].Class[j].ID.ClassNodeNumber=j;
SharedPools[i].Class[j].ID.ParentSharedPoolNodeNumber=i;
//@ instantiate for the public section of this class
SharedPools[i].Class[j].PublicSection=new SharedPublicSectionNode();
for (int k=1; k<=SMP[i][j];k++){
    SharedPools[i].Class[j].PublicSection.Method[k]=new SharedMethodNode();
    SharedPools[i].Class[j].PublicSection.Method[k].ReturnedValue("void");
    SharedPools[i].Class[j].PublicSection.Method[k].ID.MethodNodeNumber=k;
    SharedPools[i].Class[j].PublicSection.Method[k].ID.ParentSharedClass.ClassNodeNumber=j;
    SharedPools[i].Class[j].PublicSection.Method[k].ID.ParentSharedClass.ParentSharedPoolNodeNumber=i;
}
//@ instantiate for the protected section of this class
SharedPools[i].Class[j].ProtectedSection=new SharedProtectedSectionNode();
for (int k=1; k<=SMR[i][j];k++){
    SharedPools[i].Class[j].ProtectedSection.Method[k]=new SharedMethodNode();
    SharedPools[i].Class[j].ProtectedSection.Method[k].ReturnedValue("void");
    SharedPools[i].Class[j].ProtectedSection.Method[k].ID.MethodNodeNumber=k;
    SharedPools[i].Class[j].ProtectedSection.Method[k].ID.ParentSharedClass.ClassNodeNumber=j;
    SharedPools[i].Class[j].ProtectedSection.Method[k].ID.ParentSharedClass.ParentSharedPoolNodeNumber=i;
}
//@ instantiate for the private section of this class
SharedPools[i].Class[j].PrivateSection=new SharedPrivateSectionNode();
for (int k=1; k<=SMV[i][j];k++){
    SharedPools[i].Class[j].PrivateSection.Method[k]=new SharedMethodNode();
    SharedPools[i].Class[j].PrivateSection.Method[k].ReturnedValue("void");
    SharedPools[i].Class[j].PrivateSection.Method[k].ID.MethodNodeNumber=k;
    SharedPools[i].Class[j].PrivateSection.Method[k].ID.ParentSharedClass.ClassNodeNumber=j;
    SharedPools[i].Class[j].PrivateSection.Method[k].ID.ParentSharedClass.ParentSharedPoolNodeNumber=i;
    }

/**
 * This method was created by a SmartGuide.
 */
private static void reportStructure(String output) {
    System.out.print(output);
    return;
}
Appendix B2

A Prototype For CO–Toolkits’ Design and Implementation Structures:
Example#1 – An Artificial Example

B2.1 Overview

This Appendix presents the first artificial example that has been used to demonstrate our prototype for CO–Toolkits’ design and implementation structures. Section B2.3 shows that this example passes the prototype test (i.e. each predicate returns STATUS=true).

B2.2 Example#1: Design and Implementation Structure

[Design and Implementation Structure: As Extracted From The Current Example Framework]

Component(1,1)
Class(1,1.1): class111
Public Inheritance From Classes: Class(1,2.2):class122

Public Section Methods & Objects
MethodName: p_mhdl1111
ReturnedValueType: void
IN_ArgNames(Types): X(T1) Y(T2)
OUT_ArgNames(Types): Z(T3)
MethodName: p_mhdl1112
ReturnedValueType: T4
IN_ArgNames(Types): A(T3) B(T2)
OUT_ArgNames(Types):

Protected Section Methods & Objects
MethodName: r_mhdl1111
ReturnedValueType: void
IN_ArgNames(Types): X(T2)
OUT_ArgNames(Types): Y(T3)
Objects: r_obj1112(class112) r_obj1112(T12)

Private Section Methods & Objects
MethodName: v_mhdl1111
Objects: r_obj1221(T111) r_obj1222(T6)
Private Section Methods & Objects
MethodName: v_mthd1221
ReturnedValueType: T1
IN_ArgNames(Types):
OUT_ArgNames(Types):
Objects: v_obj1221(class111) v_obj1222(class221) v_obj1223(class122) v_obj1224(T8)
Framework(2)
Component(2,1)
Class(2,1,1): class211
Public Inheritance From Classes: Class(2,1,2): class212
Public Section Methods & Objects
MethodName: p_mthd2111
ReturnedValueType: T1
IN_ArgNames(Types):
OUT_ArgNames(Types):
Objects: r_obj2111(class112) r_obj2112(T13)
Private Section Methods & Objects
MethodName: v_mthd2111
ReturnedValueType: T2
IN_ArgNames(Types):
OUT_ArgNames(Types):
Objects: v_obj2111(class121) v_obj2112(class122) v_obj2113(T7) v_obj2114(T6)
Class(2,1,2): class212
Public Section Methods & Objects
MethodName: p_mthd2121
ReturnedValueType: T1
IN_ArgNames(Types):
OUT_ArgNames(Types):
MethodName: p_mthd2122
ReturnedValueType: T3
IN_ArgNames(Types):
OUT_ArgNames(Types):
Protected Section Methods & Objects
MethodName: r_mthd2121
ReturnedValueType: T2
IN_ArgNames(Types):
OUT_ArgNames(Types):
Objects: r_obj2121(T11) r_obj2122(T2)
Private Section Methods & Objects
MethodName: v_mthd2121
ReturnedValueType: T7
IN_ArgNames(Types):
OUT_ArgNames(Types):
Objects: v_obj2121(class212) v_obj2122(class221) v_obj2123(class222) v_obj2124(class211)
Component(2,2)
Class(2,2,1): class221
Public Inheritance From Classes: Class(2,2,2): class222
Public Section Methods & Objects
MethodName: p_mthd2221
ReturnedValueType: T1
IN_ArgNames(Types):
OUT_ArgNames(Types):
MethodName: p_mthd2222
ReturnedValueType: T1
IN_ArgNames(Types):
OUT_ArgNames(Types):
OUT_ArgNames(Types):
Protected Section Methods & Objects
MethodName: r_mhd2211
   ReturnedValueType: T5
IN_ArgNames(Types):
   OUT_ArgNames(Types):
   Objects: r_obj2211(T9) r_obj2212(T3)
Private Section Methods & Objects
MethodName: v_mhd2211
   ReturnedValueType: T5
IN_ArgNames(Types):
   OUT_ArgNames(Types):
   Objects: v_obj2211(T8) v_obj2212(T10) v_obj2213(class211) v_obj2214(class221)
Class(2,2,2): class222
Public Section Methods & Objects
MethodName: p_mhd2221
   ReturnedValueType: T2
IN_ArgNames(Types): X(T5)
   OUT_ArgNames(Types):
   MethodName: p_mhd2222
   ReturnedValueType: T8
IN_ArgNames(Types): T(T5)
   OUT_ArgNames(Types):
Protected Section Methods & Objects
MethodName: r_mhd2221
   ReturnedValueType: T4
IN_ArgNames(Types): R(T5)
   OUT_ArgNames(Types):
   Objects: r_obj2221(T11) r_obj2222(T21)
Private Section Methods & Objects
MethodName: v_mhd2221
   ReturnedValueType: T5
IN_ArgNames(Types):
   OUT_ArgNames(Types):
   Objects: v_obj2221(T7) v_obj2222(class122) v_obj2223(class121) v_obj2224(class222)
SharedPool(1)
Class(1,1): s_class11
Public Inheritance From Classes: SharedClass(1,2)
Public Section Methods & Objects
MethodName: s_p_mhd111
   ReturnedValueType: T5
IN_ArgNames(Types): X(class112) Y(s_class12)
   OUT_ArgNames(Types):
   MethodName: s_p_mhd112
   ReturnedValueType: T7
IN_ArgNames(Types): A(T1) B(T2)
   OUT_ArgNames(Types):
   Objects:
Protected Section Methods & Objects
MethodName: s_r_mhd111
   ReturnedValueType: void
IN_ArgNames(Types): X(T7)
   OUT_ArgNames(Types): Y(T2)
   Objects: s_r_obj112(class112) s_r_obj112(T2)
Private Section Methods & Objects
MethodName: s_v_mhd111
   ReturnedValueType: void
IN_ArgNames(Types): W(T6)
   OUT_ArgNames(Types): G(s_class22)
   Objects: s_v_obj111(s_class21) s_v_obj112(class221) s_v_obj113(T9) s_v_obj114(T8)
Class(1,2): s_class12
Public Section Methods & Objects
MethodName: s_p_mhd1121
   ReturnedValueType: T4
IN_ArgNames(Types): A(T3)
Appendix B2: A Prototype For CO-Toolkits’ Design and Implementation Structures:

Example#1 - An Artificial Example

OUT_ArgNames(Types):
MethodName: s_p_mhd122
ReturnedValueType: T4
IN_ArgNames(Types): A(T5)
OUT_ArgNames(Types):
Objects: s_p_obj121(T5)

Protected Section Methods & Objects
MethodName: s_r_mhd121
ReturnedValueType: T4
IN_ArgNames(Types):
OUT_ArgNames(Types):
Objects: s_r_obj121(s_class12) s_r_obj122(T5)

Private Section Methods & Objects
MethodName: s_v_mhd121
ReturnedValueType: T3
IN_ArgNames(Types):
OUT_ArgNames(Types):
Objects: s_v_obj121(s_class11) s_v_obj122(T6) s_v_obj123(T7) s_v_obj124(T1)

SharedPool(2)
Class(2,1): s_class21

Public Inheritance From Classes: SharedClass(2,2)

Public Section Methods & Objects
MethodName: s_p_mhd211
ReturnedValueType: T8
IN_ArgNames(Types):
OUT_ArgNames(Types):
MethodName: s_p_mhd212
ReturnedValueType: T5
IN_ArgNames(Types):
OUT_ArgNames(Types):
Objects: s_p_obj211(T12)

Protected Section Methods & Objects
MethodName: s_r_mhd211
ReturnedValueType: T1
IN_ArgNames(Types):
OUT_ArgNames(Types):
Objects: s_r_obj211(s_class12) s_r_obj212(T13)

Private Section Methods & Objects
MethodName: s_v_mhd211
ReturnedValueType: T2
IN_ArgNames(Types):
OUT_ArgNames(Types):
Objects: s_v_obj211(s_class21) s_v_obj212(class22) s_v_obj213(T7) s_v_obj214(T6)

Class(2,2): s_class22

Public Section Methods & Objects
MethodName: s_p_mhd221
ReturnedValueType: void
IN_ArgNames(Types):
OUT_ArgNames(Types):
MethodName: s_p_mhd222
ReturnedValueType: T3
IN_ArgNames(Types):
OUT_ArgNames(Types):
Objects:

Protected Section Methods & Objects
MethodName: s_r_mhd221
ReturnedValueType: T2
IN_ArgNames(Types):
OUT_ArgNames(Types):
Objects: s_r_obj221(T1) s_r_obj222(T2)

Private Section Methods & Objects
MethodName: s_v_mhd221
ReturnedValueType: T7
B2.3 Example#1: Output Report

Checking for Main predicate 5_18: GoodDesignAndImplementation
Checking for Predicate 5_19: GoodUseOfRelationsAmongComponents
Checking for predicate 5_20: SafeInheritance
Checking for predicate 5_23: InheritanceMayNotExistAcrossFrameworks
Checking for predicate 5_23 is complete. STATUS: true
Checking for predicate 5_24: SafeInheritance_AmongComponents_WithinTheSameFramework
Checking for predicate 5_31: A_ComponentMayInheritFromOnlyOneOtherComponentFromTheSameFramework
Checking for predicate 5_31 is complete. STATUS: true
Checking for predicate 5_32: OnlyPublicInheritanceMayExistAcrossComponentsWithinTheSameFramework
Checking for predicate 5_32 is complete. STATUS: true
Checking for predicate 5_24 is complete. STATUS: true
Checking for Predicate 5_25: SafeInheritance_AmongClasses_WithinTheSameComponent
Checking for Predicate 5_33: NoMultipleInheritanceMayExistAmongClassesWithinTheSameComponent
Checking for predicate 5_33 is complete. STATUS: true
Checking for predicate 5_34: InheritanceMayBe_Public/Protected/or/Private_WithinTheSameComponent
Checking for predicate 5_34 is complete. STATUS: true
Checking for Predicate 5_25 is complete. STATUS: true
Checking for Predicate 5_26: SafeInheritance_AmongClasses_WithinTheSharedPools
Checking for predicate 5_35: InheritanceMayNotExistAcrossSharedPools
Checking for Predicate 5_35 is complete. STATUS: true
Checking for predicate 5_36: NoMultipleInheritanceMayExistAmongClassesWithinTheSameSharedPool
Checking for predicate 5_36 is complete. STATUS: true
Checking for predicate 5_26 is complete. STATUS: true
Checking for predicate 5_20 is complete. STATUS: true
Checking for predicate 5_21: GoodAssociation
Checking for Predicate 5_27: GoodAssociationOnTheComponentLevel
Checking for predicate 5_37: ComponentsMayHaveOnlyUniDirectionalAssociationWithClassesFromTheSharedPools
Checking for predicate 5_37 is complete. STATUS: true
Checking for predicate 5_38: AssociationMayNotExistAcrossFrameworks
Checking for predicate 5_38 is complete. STATUS: true
Checking for Predicate 5_27 is complete. STATUS: true
Checking for Predicate 5_28: GoodAssociationOnInternalClassesLevel
Checking for predicate 5_38: AssociationAndInheritanceMayNotExistBetweenTwoClassesAtTheSameTime
Checking for predicate 5_38 is complete. STATUS: true
Checking for Predicate 5_28 is complete. STATUS: true
Checking for predicate 5_21 is complete. STATUS: true
Checking for predicate 5_22: GoodUseOfOtherRelations
Checking for predicate 5_22 is complete. STATUS: true
Checking for Main predicate 5_18: GoodDesignAndImplementation is complete. STATUS: true
Appendix B3

A Prototype For CO–Toolkits' Design and Implementation Structures: Example#2 – An Artificial Example

B3.1 Overview

This Appendix presents the second artificial example that has been used to demonstrate our prototype for CO–Toolkits' design and implementation structures. Section B3.3 presents the rules violations reported by this prototype for this example.

B3.2 Example#2: Design and Implementation Structure

Design and Implementation Structure: As Extracted From The Current Example
Framework(1)
  Component(1,1): class111
    Public Inheritance From Classes: Class(1,2,2):class122 Class(2,2,1):class221
    Private Inheritance From Classes: Class(2,2,2):class212
    Protected Inheritance From Classes: Class(1,2,1):class211 SharedClass(2,1):x_class21
    Public Section Methods & Objects
      MethodName: p_meth1111
        Returned Value Type: void
        IN_ArgNames(Types): X(T1) Y(T2)
        OUT_ArgNames(Types): Z(T3)
      MethodName: p_meth1112
        Returned Value Type: T4
        IN_ArgNames(Types): A(T3) B(T2)
        OUT_ArgNames(Types): 
    Protected Section Methods & Objects
      MethodName: r_meth1111
        Returned Value Type: void
        IN_ArgNames(Types): X(T2)
        OUT_ArgNames(Types): Y(T3)
      Objects: r_obj1112(class112) r_obj1113(T12)
    Private Section Methods & Objects
      MethodName: v_meth1111
        Returned Value Type: void
        IN_ArgNames(Types): Z(T1)
        OUT_ArgNames(Types): W(T5)
Objects: v_obj1111(class221) v_obj1112(class222) v_obj1113(T9) v_obj1114(T5)

Class(1.1.2): class121
Public Section Methods & Objects
MethodName: p_meth1111
  ReturnedValueTypes: T4
  IN_ArxNameTypes: A(T3)
  OUT_ArxNameTypes:
MethodName: p_meth1122
  ReturnedValueTypes: T4
  IN_ArxNameTypes: A(T3) B(T2)
  OUT_ArxNameTypes:
Protected Section Methods & Objects
MethodName: r_meth1121
  ReturnedValueTypes: T4
  IN_ArxNameTypes:
  OUT_ArxNameTypes:
Objects: r_obj1121(class212) r_obj1122(T5)
Private Section Methods & Objects
MethodName: v_meth1121
  ReturnedValueTypes: T3
  IN_ArxNameTypes: A(T3)
  OUT_ArxNameTypes:
Objects: v_obj1121(class111) v_obj1122(T6) v_obj1123(T7) v_obj1124(T11)

Component(1.2)
Class(1.2.1): class121
Public Inheritance From Classes: Class(1.2.1):class121
Public Section Methods & Objects
MethodName: p_meth1121
  ReturnedValueTypes: void
  IN_ArxNameTypes:
  OUT_ArxNameTypes:
MethodName: p_meth1122
  ReturnedValueTypes: void
  IN_ArxNameTypes: X(T1)
  OUT_ArxNameTypes: Z(T3)
Protected Section Methods & Objects
MethodName: r_meth1211
  ReturnedValueTypes: T3
  IN_ArxNameTypes:
  OUT_ArxNameTypes:
Objects: r_obj1211(class212) r_obj1212(T10)
Private Section Methods & Objects
MethodName: v_meth1211
  ReturnedValueTypes: T2
  IN_ArxNameTypes:
  OUT_ArxNameTypes:
Objects: v_obj1211(class212) v_obj1212(T6) v_obj1213(T2) v_obj1214(class222)

Class(1.2.2): class122
Protected Inheritance From Classes: Class(2.2.1):class221
Public Section Methods & Objects
MethodName: p_meth1221
  ReturnedValueTypes: T2
  IN_ArxNameTypes: Y(T2)
  OUT_ArxNameTypes:
MethodName: p_meth1222
  ReturnedValueTypes: T1
  IN_ArxNameTypes: Y(T1)
  OUT_ArxNameTypes:
Protected Section Methods & Objects
MethodName: r_meth1221
  ReturnedValueTypes: T1
  IN_ArxNameTypes: X(T4)
  OUT_ArxNameTypes:
Objects: r_obj1221(T71) r_obj1222(T6)
Private Section Methods & Objects
MethodName: v_meth1221
  ReturnedValueTypes: T1
  IN_ArxNameTypes:
  OUT_ArxNameTypes:
Objects: v_obj1221(class111) v_obj1222(class221) v_obj1223(T12) v_obj1224(T8)

Framework(2)
Component(2.1)
Class(2.1.1): class211
Public Inheritance From Classes: Class(2.1.1):class211
Public Section Methods & Objects
MethodName: p_meth2111
  ReturnedValueTypes: T3
  IN_ArxNameTypes: Z(T6)
  OUT_ArxNameTypes:
MethodName: p_meth2112
  ReturnedValueTypes: T2
  IN_ArxNameTypes: W(T1)
  OUT_ArxNameTypes:
Appendix B3: A Prototype For CO-Toolkits’ Design and Implementation Structures:

Example#2 – An Artificial Example

B3.3 Example#2: Output Report

Checking for Main predicate_1: GoodDesignAndImplementation
Checking for predicate_1: GoodUseOfRelationsAmongComponents
Checking for predicate_20: SafeInheritance
Checking for predicate_22: InheritanceMayNotExistAcrossFrameworks
VIOLATION(Predicate_22): Class(2,1) inherits from Class(2,2) from same framework
Checking for predicate_23: Class(1,1) inherits in public from Class(2,2,1) from another framework
VIOLATION(Predicate_23): Class(1,1) inherits in protected from Class(2,2,1) from another framework
VIOLATION(Predicate_23): Class(1,1) inherits in private from Class(2,2,1) from another framework
VIOLATION(Predicate_23): Class(1,1) inherits in protected from Class(2,2,1) from another framework
VIOLATION(Predicate_23): Class(1,1) inherits in private from Class(2,2,1) from another framework
Checking for predicate_23 is complete. STATUS: false
Checking for predicate_24: SafeInheritance_AmongComponents_WithinTheSameFramework
Checking for predicate_30: AComponentMayInheritFromOnlyOneOtherComponentInTheSameFramework
VIOLATION(Predicate_30): Class(1,1) inherits from Class(2,2,1) from different framework
VIOLATION(Predicate_30): Class(1,1) inherits from Class(2,2,1) from different framework
VIOLATION(Predicate_30): Class(1,1) inherits from Class(2,2,1) from different framework
VIOLATION(Predicate_30): Class(1,1) inherits from Class(2,2,1) from different framework
VIOLATION(Predicate_30): Class(1,1) inherits from Class(2,2,1) from different framework
Checking for predicate_30 is complete. STATUS: false
Checking for predicate_31: OnlyPublicInheritanceMayExistAcrossComponentsWithinTheSameFramework
VIOLATION(Predicate_31): Class(1,1) inherits from Class(2,2) from different framework
VIOLATION(Predicate_31): Class(1,1) inherits from Class(2,2) from different framework
VIOLATION(Predicate_31): Class(1,1) inherits from Class(2,2) from different framework
VIOLATION(Predicate_31): Class(1,1) inherits from Class(2,2) from different framework
VIOLATION(Predicate_31): Class(1,1) inherits from Class(2,2) from different framework
Checking for predicate_31 is complete. STATUS: false
Checking for predicate_34 is complete. STATUS: false

341
Checking for Predicate 5.35: SafeInheritance Among Classes Within The Same Component
Checking for Predicate 5.32: NoMultipleInheritance May Exist Among Classes Within The Same Component
VIOLATION(Predicate 5.32): Class(1,1,1) inherits from Class(2,2,1) from different framework
VIOLATION(Predicate 5.32): Class(1,1,1) inherits from Class(2,1,2) from different framework
VIOLATION(Predicate 5.32): Class(1,1,1) inherits from Class(2,1,1) from different framework
VIOLATION(Predicate 5.32): Class(1,2,2) inherits from Class(2,2,1) from different framework
VIOLATION(Predicate 5.32): Class(3,1,1) inherits from Class(2,2,1) from different framework
VIOLATION(Predicate 5.32): Class(3,1,1) inherits from Class(1,1,2) from different framework
VIOLATION(Predicate 5.32): Class(3,1,1) inherits from Class(2,2,1) from different framework
Checking for predicate 5.32 is complete. STATUS: false
Checking for Predicate 5.33: Inheritance May Be Public/Private/Public Within The Same Component
Checking for Predicate 5.33 is complete. STATUS: true
Checking for Predicate 5.32 is complete. STATUS: false
Checking for predicate 5.36: SafeInheritance Among Classes Within The Shared Pools
Checking for predicate 5.34: Inheritance May Not Exist Across Shared Pools
VIOLATION(Predicate 5.34): SharedClass(1,1) inherits from SharedClass(2,2) from another shared pool
Checking for Predicate 5.34 is complete. STATUS: false
Checking for predicate 5.35: NoMultipleInheritance May Exist Among Classes Within The Same Shared Pool
VIOLATION(Predicate 5.35): SharedClass(1,1) inherits from SharedClass(2,2) from another shared pool
Checking for predicate 5.35 is complete. STATUS: false
Checking for predicate 5.36 is complete. STATUS: false
Checking for predicate 5.20 is complete. STATUS: false
Checking for predicate 5.21: GoodAssociation
Checking for Predicate 5.27: GoodAssociation On The Component Level
Checking for predicates 5.36: Components May Have Only Unidirectional Association With Classes From The Shared Pools
Checking for predicate 5.36 is complete. STATUS: true
Checking for predicate 5.37: Association May Not Exist Across Frameworks
Checking for predicate 5.37 is complete. STATUS: true
Checking for Predicate 5.27 is complete. STATUS: true
Checking for Predicate 5.38: GoodAssociation On Internal Classes Level
Checking for predicate 5.38: Association And Inheritance May Not Exist Between Two Classes At The Same Time
Checking for predicate 5.38 is complete. STATUS: true
Checking for Predicate 5.28 is complete. STATUS: true
Checking for predicate 5.21 is complete. STATUS: true
Checking for predicate 5.22: GoodUseOfOtherRelations
Checking for predicate 5.22 is complete. STATUS: true
Checking for Predicate 5.19 is complete. STATUS: false
Checking for Main predicate 5.18: GoodDesign And Implementation is complete. STATUS: false

342
Appendix B4

A Prototype For CO–Toolkits’ Design and Implementation Structures:
Example#3 – ACE Sample Inheritance Structure

B4.1 Overview

This Appendix presents the third example that has been used to demonstrate our prototype for CO–Toolkits’ design and implementation structures. This example was based on a sample ACE inheritance structure. Section B4.3 presents the rules violations reported by this prototype for this example.

B4.2 Example#3: ACE Sample Design and Implementation Structure

Design and Implementation Structure: As Extracted From The Current Example Framework(1)
Component(1,1)
  Class(1,1,1): ACE_Module_Link
    Public Section Methods & Objects
    Protected Section Methods & Objects
    Objects:
  Private Section Methods & Objects
Component(1,2)
  Class(1,2,1): ACE_Map_Manager
    Public Section Methods & Objects
    Protected Section Methods & Objects
    Objects:
  Private Section Methods & Objects
  Objects:
  Class(1,2,2): ACE_Map_Iterator
    Public Section Methods & Objects
    Protected Section Methods & Objects
    Objects:
Private Section Methods & Objects
Objects:

Class(1,2,3): ACE_Map_Reverse_Iterator
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component(1,3)
Class(1,3,1): ACE_Message_Block
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component(1,4)
Class(1,4,1): ACE_Notification_Strategy
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class(1,4,2): ACE_Message_Queue
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class(1,4,3): ACE_Message_Queue_Iterator
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class(1,4,4): ACE_Message_Queue_Reverse_Iterator
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component(1,5)
Class(1,5,1): ACE_Module
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component(1,6)
Class(1,6,1): Driver
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class(1,6,2): ACE_Multiplexer
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component(1,7)
Class(1,7,1): ACE_Stream
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class(1,7,2): ACE_Stream_Iterator
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component(1,8)
Class(1,8,1): ACE_Stream_Head
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class(1,8,2): ACE_Stream_Tail
Public Section Methods & Objects
Appendix B4: A Prototype For CO-Toolkits' Design and Implementation Structures: Example#3 – ACE Sample Inheritance Structure

Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class (1,8,3): ACE_Thru_Task
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component (1,9)
Class (1,9,1): ACE_Task_Settings
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class (1,9,2): ACE_Task_Base
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class (1,9,3): ACE_Task_Exit
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Framework (2)
Component (2,1)
Class (2,1,1): ACE_CORBA_Handler
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class (2,1,2): ACE_ST_CORBA_Handler
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class (2,1,3): ACE_MT_CORBA_Handler
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component (2,2)
Class (2,2,1): ACE_CORBA_Ref
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Framework (3)
Component (3,1)
Class (3,1,1): ACE_Set_Node
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class (3,1,2): ACE_Unbounded_Set_Iterator
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class (3,1,3): ACE_Set_Node
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class (3,1,4): ACE_Unbounded_Set
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
  Objects:
Class(3.1.5): ACE_Fixed_Set
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects
  Objects:
Class(3.1.6): ACE_Fixed_Set_Iterator
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects
  Objects:
Class(3.1.7): ACE_Bounded_Set
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects
  Objects:
Class(3.1.8): ACE_Bounded_Set_Iterator
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects
  Objects:
Class(3.1.9): ACE_Bounded_Set_Iterator
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects
  Objects:
Component(3.2)
Class(3.2.1): ACE_Stack_Node
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects
  Objects:
Class(3.2.2): ACE_Queue_Node
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects
  Objects:
Class(3.2.3): ACE_Bounded_Stack
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects
  Objects:
Class(3.2.4): ACE_Bounded_Stack
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects
  Objects:
Class(3.2.5): ACE_Stack_Node
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects
  Objects:
Class(3.2.6): ACE_Unbounded_Stack
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects
  Objects:
Class(3.2.7): ACE_Queue_Node
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects
  Objects:
Class(3.2.8): ACE_Unbounded_Queue
Appendix B4: A Prototype For CO-Toolkits’ Design and Implementation Structures:
Example#3 – ACE Sample Inheritance Structure

Component(3.3)
Class(3.3.1): ACE_Allocator
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class(3.3.2): ACE_CString
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class(3.3.3): ACE_UString
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class(3.3.4): ACEWSTR
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Framework(4)
Component(4.1)
Class(4.1.1): ACE_Activation_Queue
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component(4.2)
Class(4.2.1): ACE_Method_Object
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component(4.3)
Class(4.3.1): ACE_Process_Descriptor
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class(4.3.2): ACE_Process_Control
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class(4.3.3): ACE_Process_Manager
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class(4.3.4): ACE_Process_Control
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component(4.4)
Class(4.4.1): ACE_File_Lock
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class(4.4.2): ACE_Semaphore
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Appendix B4: A Prototype For CO-Toolkits' Design and Implementation Structures:

Example #3 – ACE Sample Inheritance Structure

Objects:
Private Section Methods & Objects
Objects:
Class(4,5,9): ACE_Guard
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(4,6,10): ACE_Write_Guard
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(4,6,7): ACE_Read_Guard
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(4,6,8): ACE_TSS_Guard
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(4,6,9): ACE_TSS_Write_Guard
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(4,6,10): ACE_TSS_Read_Guard
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(4,6,11): ACE_COND
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(4,6,12): ACE_Process_COND
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(4,6,13): ACE_Thread_COND
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(4,6,14): ACE_MT_SYNCH
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Component(4,7)
Class(4,7,1): ACE_Thread
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Component(4,8)
Class(4,8,1): ACE_Task_Base
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(4,8,2): ACE_Thread_Manager
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Appendix B4: A Prototype For CO-Toolkits' Design and Implementation Structures:
Example#3 – ACE Sample Inheritance Structure

Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Component(6.8)
Class(6.8.1): ACE_IPC_SAP
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Component(6.9)
Class(6.9.1): ACE_Addr
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Component(6.10)
Class(6.10.1): ACE_DEV_Addr
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Component(6.11)
Class(6.11.1): ACE_FILE_Addr
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Component(6.12)
Class(6.12.1): ACE_INET_Addr
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Component(6.13)
Class(6.13.1): ACE_SPIPE_Addr
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Component(6.14)
Class(6.14.1): ACE_UNIX_Addr
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Component(6.15)
Class(6.15.1): ACE_FIFO
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Component(6.16)
Class(6.16.1): ACE_FIFO_Recv
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Component(6.17)
Class(6.17.1): ACE_FIFO_Rcv_Msg
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Appendix B4: A Prototype For CO-Toolkits' Design and Implementation Structures:

Example #3 – ACE Sample Inheritance Structure

Objects:

Class (6.18)
Component (6.18): ACE_FIFO_Send
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class (6.15.1): ACE_FIFO

Class (6.18): ACE_FIFO_Send
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component (6.19)
Class (6.19.1): ACE_FIFO_Send_Msg
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component (6.20)
Class (6.20.1): ACE_LSOCK
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component (6.21)
Class (6.21.1): ACE_LSOCK_Acceptor
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component (6.22)
Class (6.22.1): ACE_LSOCK_Dgram
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component (6.23)
Class (6.23.1): ACE_LSOCK_Connector
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component (6.24)
Class (6.24.1): ACE_LSOCK_Dgram
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component (6.25)
Class (6.25.1): ACE.SOCK
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component (6.26)
Class (6.26.1): ACE.SOCK_Acceptor
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component (6.27)
Class (6.27.1): ACE.SOCK_Dgram
Public Inheritance From Classes:

Class (6.27.1): ACE.SOCK_IO
Appendix B4: A Prototype For CO–Toolkits Design and Implementation Structures:
Example #3 – ACE Sample Inheritance Structure

Component (6.28)
Class (6.28.1): ACE_SOCK_Connector
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class (6.28.1): ACE_SOCK

Component (6.29)
Class (6.29.1): ACE_SOCK_Dgram
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class (6.29.1): ACE_SOCK

Component (6.30)
Class (6.30.1): ACE_SOCK_Dgram_Bcast
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class (6.30.1): ACE_SOCK_Dgram

Component (6.31)
Class (6.31.1): ACE_SOCK_Dgram_Mcast
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class (6.31.1): ACE_SOCK_Dgram

Component (6.32)
Class (6.32.1): ACE_SOCK_IO
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class (6.32.1): ACE_SOCK

Component (6.33)
Class (6.33.1): ACE_SOCK_Sream
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class (6.33.1): ACE_SOCK_IO

Component (6.34)
Class (6.34.1): ACE_SPIPE
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class (6.34.1): ACE_SPIPE

Component (6.35)
Class (6.35.1): ACE_SPIPE_Acceptor
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class (6.35.1): ACE_SPIPE

Component (6.36)
Class (6.36.1): ACE_SPIPE_Connector
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class (6.36.1): ACE_SPIPE

Component (6.37)
Class (6.37.1): ACE_SPIPE_Stream
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects

Class (6.37.1): ACE_SPIPE
Appendix B4: A Prototype For CO–Toolkits' Design and Implementation Structures:  
Example#3 – ACE Sample Inheritance Structure

Objects:
Private Section Methods & Objects
Objects:

Component(6,38)
Class(6,38.1): ACE_TLI
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component(6,39)
Class(6,39.1): ACE_TLI_Request_Queue
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class(6,39.2): ACE_TLI_Request_Queue
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Class(6,39.3): ACE_TLI_Acceptor
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component(6,40)
Class(6,40.1): ACE_TLI_Connector
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component(6,41)
Class(6,41.1): ACE_TLI_Stream
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component(6,42)
Class(6,42.1): ACE_UPipe_Acceptor
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component(6,43)
Class(6,43.1): ACE_UPipe_Connector
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component(6,44)
Class(6,44.1): ACE_UPipe_Stream
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Framework(7)
Component(7,1)
Class(7,1.1): ACE_Log_Msg
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component(7,2)
Class(7,2.1): ACE_Log_Record
Public Section Methods & Objects
Objects:
Appendix B4: A Prototype For CO-Toolkits’ Design and Implementation Structures:

Example #3 – ACE Sample Inheritance Structure

Protected Section Methods & Objects
    Objects:
Private Section Methods & Objects
    Objects:

Framework(1)
Component(2.1):
Class(8.1.1): ACE_Mem_Map
    Public Section Methods & Objects
    Protected Section Methods & Objects
    Objects:
    Private Section Methods & Objects
    Objects:

Component(2.2):
Class(8.2.1): ACE_Allocator
    Public Section Methods & Objects
    Protected Section Methods & Objects
    Objects:
    Private Section Methods & Objects
    Objects:
Class(8.2.2): ACE_Name_Node
    Public Section Methods & Objects
    Protected Section Methods & Objects
    Objects:
    Private Section Methods & Objects
    Objects:
Class(8.2.3): ACE_Control_Block
    Public Section Methods & Objects
    Protected Section Methods & Objects
    Objects:
    Private Section Methods & Objects
    Objects:

Component(8.3):
Class(8.3.1): ACE_Allocator_Adapter
    Public Section Methods & Objects
    Protected Section Methods & Objects
    Objects:
    Private Section Methods & Objects
    Objects:
Class(8.3.2): ACE_Malloc
    Public Section Methods & Objects
    Protected Section Methods & Objects
    Objects:
    Private Section Methods & Objects
    Objects:
Class(8.3.3): ACE_Malloc_Iterator
    Public Section Methods & Objects
    Protected Section Methods & Objects
    Objects:
    Private Section Methods & Objects
    Objects:

Component(8.4):
Class(8.4.1): ACE_Shark_Memory_Pool_Options
    Public Section Methods & Objects
    Protected Section Methods & Objects
    Objects:
    Private Section Methods & Objects
    Objects:
Class(8.4.2): ACE_Shark_Memory_Pool
    Public Section Methods & Objects
    Protected Section Methods & Objects
    Objects:
    Private Section Methods & Objects
    Objects:
Class(8.4.3): ACE_Shared_Memory_Pool_Options
    Public Section Methods & Objects
    Protected Section Methods & Objects
    Objects:
    Private Section Methods & Objects
    Objects:
Class(8.4.4): ACE_Shared_Memory_Pool
    Public Inheritance From Classes:
    Public Section Methods & Objects
    Protected Section Methods & Objects
    Objects:
    Private Section Methods & Objects
    Objects:
Class(8.4.5): ACE_Local_Memory_Pool_Options
    Public Section Methods & Objects
    Protected Section Methods & Objects
    Objects:
    Private Section Methods & Objects
    Objects:
Appendix B4: A Prototype For CO-Toolkits' Design and Implementation Structures:

Example#3 – ACE Sample Inheritance Structure

```plaintext
Class(8,4.6): ACE_Local_Memory_Pool
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects

Class(8,4.7): ACE_MMAP_Memory_Pool_Options
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects

Class(8,4.8): ACE_MMAP_Memory_Pool
  Public Inheritance From Classes:
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects

Class(8,4.9): ACE_Lite_MMAP_Memory_Pool
  Public Inheritance From Classes:
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects

Component(8,5): ACE_Shared_Memory
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects

Component(8,6): ACE_Shared_Memory_MM
  Public Inheritance From Classes:
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects

Component(8,7): ACE_Shared_Memory_SV
  Public Inheritance From Classes:
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects

Framework(9)
Component(9,1): auto_ptr
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects

Component(9,1.1): auto_array
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects

Component(9,2): ACE_Dynamic
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects

Component(9,3): ACE_Get_Opt
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects

Component(9,4): ACE_Singleton
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
```

Class(12,1.1): ACE_Event_Handler

Class(8,4.5): ACE_Lite_MMAP_Memory_Pool

Class(8,5.1): ACE_Shared_Memory

Class(8,5.1): ACE_Shared_Memory

Class(8,5.1): ACE_Shared_Memory

Class(8,5.1): ACE_Shared_Memory

Class(8,5.1): ACE_Shared_Memory

358
Appendix B4: A Prototype For CO-Toolkits' Design and Implementation Structures:

Example#3 – ACE Sample Inheritance Structure

Private Section Methods & Objects

Framework(10)

Component(10.1)
  Class(10.1.1): ACE_NS_String
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects

Component(10.2)
  Class(10.2.1): ACE_Name_Space_Map
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects

Class(10.2.2): ACE_Local_Name_Space
  Public Inheritance From Classes:
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects

Component(10.3)
  Class(10.3.1): ACE_Name_Proxy
  Public Inheritance From Classes:
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects

Component(10.4)
  Class(10.4.1): ACE_Name_Request
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects

Class(10.4.2): ACE_Name_Reply
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects

Component(10.5)
  Class(10.5.1): ACE_Name_Binding
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects

Class(10.5.2): ACE_Name_Space
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects

Component(10.6)
  Class(10.6.1): ACE_Naming_Context
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects

Class(10.6.2): ACE_Name_Options
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects

Component(10.7)
  Class(10.7.1): ACE_Remote_Name_Space
  Public Inheritance From Classes:
  Public Section Methods & Objects
  Protected Section Methods & Objects

Class(10.5.2): ACE_Name_Space
Appendix B4: A Prototype For CO-Toolkits’ Design and Implementation Structures:

Example #3 – ACE Sample Inheritance Structure

Objects:
- Private Section Methods & Objects
- Objects:

Framework (11)
  Component (11.1)
  Class (11.1.1): ACE_TSS_Ref
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects
  Objects:
  Class (11.1.2): ACE_TSS_Info
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects
  Objects:
  Class (11.1.3): ACE_TSS_Cleanup
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects
  Objects:
  Class (11.1.4): ACE_Thread_Adapter
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects
  Objects:
  Class (11.1.5): ACE_OS
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects
  Objects:

Framework (12)
  Component (12.1)
  Class (12.1.1): ACE_Event_Handler
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects
  Objects:
  Class (12.1.2): ACE_Event_Handler_T
  Public Inheritance From Classes:
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects
  Objects:
  Component (12.2)
  Class (12.2.1): ACE_Handle_Set
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects
  Objects:
  Class (12.2.2): ACE_Handle_Set_Iterator
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects
  Objects:

Component (12.3)
  Class (12.3.1): ACE_Overlapped_IO
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects
  Objects:
  Class (12.3.2): ACE_Proactor
  Public Inheritance From Classes:
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:
  Private Section Methods & Objects
  Objects:
  Class (12.3.3): ACE_Overlapped_File
  Public Section Methods & Objects
  Protected Section Methods & Objects
  Objects:

Class (12.1.1): ACE_Event_Handler
Private Section Methods & Objects
Objects:
Class(12,4,4): ACE_Reactor_Notify
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(12,4,5): ACE_Reactor_Token
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(12,4,6): ACE_Handler_Repository
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(12,4,7): ACE_Handler_Repository_Iterator
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(12,4,8): ACE_Reactor
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Component(12,5)
Class(12,5,1): ACE_Reactor_EA_Token
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(12,5,2): ACE_Reactor_EA_Notify
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(12,5,3): ACE_Reactor_EA
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Component(12,6)
Class(12,6,1): ACE_Sig_Handlers_Set
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(12,6,2): ACE_Sig_Set
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(12,6,3): ACE_Sig_Action
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(12,6,4): ACE_Sig_Guard
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(12,6,5): ACE_Sig_Handler
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Appendix B4: A Prototype For CO-Toolkits' Design and Implementation Structures:
Example#3 – ACE Sample Inheritance Structure

Framework(14)
Component(14,1)
Class(14,1.1): ACE_Profile_Timer
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Component(14.2)
Class(14.2.1): ACE_High_Res_Timer
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

Framework(15)
Component(15.1)
Class(15.1.1): ACE_TOKEN_CONST
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(15.1.2): ACE_TPO_Entry
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(15.1.3): ACE_TSS_TPO_Entry
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(15.1.4): ACE_TPO_Error
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(15.1.5): ACE_Token_Proxy_Queue
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(15.1.6): ACE_Tokens
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(15.1.7): ACE_Local_Mutex
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(15.1.8): ACE_Mutex_Token
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(15.1.9): ACE_RW_Token
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:

Class(15.1.6): ACE_Tokens
Class(15.1.9): ACE_RW_Token

363
Appendix B4: A Prototype For CO-Toolkits' Design and Implementation Structures:

Example #3 – ACE Sample Inheritance Structure

Private Section Methods & Objects

Objects:

Class 15.1.10: ACE_Token_Name
Public Inheritance From Classes:

Protected Section Methods & Objects

Objects:

Class 15.1.10: ACE_Token_Name
Public Inheritance From Classes:

Private Section Methods & Objects

Objects:

Class 15.1.10: ACE_Token_Name
Public Inheritance From Classes:

Public Section Methods & Objects

Objects:

Class 15.1.10: ACE_Token_Name
Public Inheritance From Classes:

Protected Section Methods & Objects

Objects:

Class 15.1.10: ACE_Token_Name
Public Inheritance From Classes:

Class 15.1.11: ACE_Token_Proxy
Public Inheritance From Classes:

Protected Section Methods & Objects

Objects:

Class 15.1.11: ACE_Token_Proxy
Public Inheritance From Classes:

Private Section Methods & Objects

Objects:

Class 15.1.11: ACE_Token_Proxy
Public Inheritance From Classes:

Class 15.1.12: ACE_Null_Token
Public Inheritance From Classes:

Protected Section Methods & Objects

Objects:

Class 15.1.12: ACE_Null_Token
Public Inheritance From Classes:

Private Section Methods & Objects

Objects:

Class 15.1.12: ACE_Null_Token
Public Inheritance From Classes:

Class 15.1.13: ACE_Lock
Public Inheritance From Classes:

Protected Section Methods & Objects

Objects:

Class 15.1.13: ACE_Lock
Public Inheritance From Classes:

Private Section Methods & Objects

Objects:

Class 15.1.13: ACE_Lock
Public Inheritance From Classes:

Class 15.1.14: ACE_RLock
Public Inheritance From Classes:

Protected Section Methods & Objects

Objects:

Class 15.1.14: ACE_RLock
Public Inheritance From Classes:

Private Section Methods & Objects

Objects:

Class 15.1.14: ACE_RLock
Public Inheritance From Classes:

Class 15.1.15: ACE_WLock
Public Inheritance From Classes:

Protected Section Methods & Objects

Objects:

Class 15.1.15: ACE_WLock
Public Inheritance From Classes:

Private Section Methods & Objects

Objects:

Class 15.1.15: ACE_WLock
Public Inheritance From Classes:

Component (15.2)

Class 15.2.1: ACE_Token_Name
Public Inheritance From Classes:

Protected Section Methods & Objects

Objects:

Class 15.2.1: ACE_Token_Name
Public Inheritance From Classes:

Private Section Methods & Objects

Objects:

Class 15.2.1: ACE_Token_Name
Public Inheritance From Classes:

Component (15.3)

Class 15.3.1: ACE_Remote_Token_Proxy
Public Inheritance From Classes:

Protected Section Methods & Objects

Objects:

Class 15.3.1: ACE_Remote_Token_Proxy
Public Inheritance From Classes:

Private Section Methods & Objects

Objects:

Class 15.3.1: ACE_Remote_Token_Proxy
Public Inheritance From Classes:

Class 15.3.2: ACE_Remote_Mutex
Public Inheritance From Classes:

Protected Section Methods & Objects

Objects:

Class 15.3.2: ACE_Remote_Mutex
Public Inheritance From Classes:

Private Section Methods & Objects

Objects:

Class 15.3.2: ACE_Remote_Mutex
Public Inheritance From Classes:

Class 15.3.3: ACE_Remote_RLock
Public Inheritance From Classes:

Protected Section Methods & Objects

Objects:

Class 15.3.3: ACE_Remote_RLock
Public Inheritance From Classes:

Private Section Methods & Objects

Objects:

Class 15.3.3: ACE_Remote_RLock
Public Inheritance From Classes:

Class 15.3.4: ACE_Remote_WLock
Public Inheritance From Classes:

Protected Section Methods & Objects

Objects:

Class 15.3.4: ACE_Remote_WLock
Public Inheritance From Classes:

Private Section Methods & Objects

Objects:

Class 15.3.4: ACE_Remote_WLock
Public Inheritance From Classes:

Class 15.3.5: ACE_TSS_Connection
Public Inheritance From Classes:

Protected Section Methods & Objects

Objects:

Class 15.3.5: ACE_TSS_Connection
Public Inheritance From Classes:

364
Appendix B4: A Prototype For CO-Toolkits' Design and Implementation Structures:
Example#3 – ACE Sample Inheritance Structure

Private Section Methods & Objects
Objects:
Component(15,4)
Class(15,5,10): ACE_Token_Collection
Public Inheritance From Classes:
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(15,1,11): ACE_Token_Proxy

Component(15,5)
Class(15,5,11): ACE_Token_Manager
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(15,6,1): ACE_Mutex_Invariants
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(15,6,2): ACE_RWLock_Invariants
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(15,6,3): ACE_Token_Invariant_Manager
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Component(15,7)
Class(15,7,1): ACE_Token_Request
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:
Class(15,7,2): ACE_Token_Reply
Public Section Methods & Objects
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

SharedPool(1)
Class(1,1,12): dummy_shared_class
Public Section Methods & Objects
Objects:
Protected Section Methods & Objects
Objects:
Private Section Methods & Objects
Objects:

B4.3 Example#3: ACE Sample Output Report

Checking for main predicate5_18: GoodDesignAndImplementation
Checking for Predicate5_19: GoodUseOfRelationshipsAmongComponents
Checking for predicate5_30: SafeInheritance
Checking for predicate5_22: InheritanceMayNotExistAcrossFrameworks
VIOLATION(Predicate5_22): Class(4,6,1) inherits in public from Class(12,1,1) from another framework
VIOLATION(Predicate5_22): Class(8,4,4) inherits in public from Class(12,1,1) from another framework
VIOLATION(Predicate5_22): Class(10,3,1) inherits in public from Class(12,1,1) from another framework
Checking for predicate5_23 is complete. STATUS: false
Checking for predicate5_24: SafeInheritance_AmongComponents_WithinTheSameFramework
Checking for predicate5_30: A_ComponentMayInheritFromOnlyOneOtherComponentFromTheSameFramework
VIOLATION(Predicate5_30): Class(4,6,1) inherits from Class(12,1,1) from different framework

365
Appendix B4: A Prototype For CO-Toolkits' Design and Implementation Structures:
Example#3 – ACE Sample Inheritance Structure

VIOLATION(Predicate_5_30): Class(8,4,4) inherits from Class(12,1,1) from different framework
VIOLATION(Predicate_5_30): Class(8,4,8) inherits from Class(12,1,1) from different framework
VIOLATION(Predicate_5_30): Class(10,3,1) inherits from Class(12,1,1) from different framework
Checking for predicate_5_30 is complete. STATUS: false
Checking for predicate_5_31: OnlyPublicInheritanceMayExistAcrossComponentsWithinTheSameFramework
Checking for predicate_5_31 is complete. STATUS: true
Checking for predicate_5_34 is complete. STATUS: false
Checking for Predicate_5_35: SelfInheritance_AmongClasses_WithinTheSameComponent
Checking for Predicate_5_52: NoMultipleInheritanceMayExistAmongClassesWithinTheSameComponent
VIOLATION(Predicate_5_52): Class(2,1,1) has multiple inheritance within the same component
VIOLATION(Predicate_5_32): Class(4,4,13) has multiple inheritance within the same component
VIOLATION(Predicate_5_32): Class(4,4,19) has multiple inheritance within the same component
VIOLATION(Predicate_5_32): Class(4,4,22) has multiple inheritance within the same component
VIOLATION(Predicate_5_32): Class(4,6,1) inherits from Class(12,1,1) from different framework
VIOLATION(Predicate_5_32): Class(5,4,5) has multiple inheritance within the same component
VIOLATION(Predicate_5_32): Class(8,4,4) inherits from Class(12,1,1) from different framework
VIOLATION(Predicate_5_32): Class(8,4,8) inherits from Class(12,1,1) from different framework
VIOLATION(Predicate_5_32): Class(10,3,1) inherits from Class(12,1,1) from different framework
VIOLATION(Predicate_5_32): Class(15,1,9) has multiple inheritance within the same component
VIOLATION(Predicate_5_32): Class(15,1,13) has multiple inheritance within the same component
VIOLATION(Predicate_5_32): Class(15,1,15) has multiple inheritance within the same component
VIOLATION(Predicate_5_32): Class(15,3,3) has multiple inheritance within the same component
Checking for predicate_5_32 is complete. STATUS: false
Checking for Predicate_5_33: InheritanceMayBe_PublicProtectedOr_Private_WithinTheSameComponent
Checking for predicate_5_33 is complete. STATUS: true
Checking for Predicate_5_34 is complete. STATUS: false
Checking for predicate_5_26: SelfInheritance_AmongClasses_WithinTheSharedPools
Checking for predicate_5_34: InheritanceMayNotExistAcrossSharedPools
Checking for Predicate_5_34 is complete. STATUS: true
Checking for Predicate_5_35: NoMultipleInheritanceMayExistAmongClassesWithinTheSameSharedPool
Checking for predicate_5_35 is complete. STATUS: true
Checking for predicate_5_26 is complete. STATUS: true
Checking for predicate_5_20 is complete. STATUS: false
Checking for predicate_5_31: GoodAssociation
Checking for predicate_5_31 is complete. STATUS: true
Checking for predicate_5_22: GoodUseOfOtherRelations
Checking for predicate_5_22 is complete. STATUS: true
Checking for Predicate_5_19 is complete. STATUS: false
Checking for Main predicate_5_18: GoodDesignAndImplementation is complete. STATUS: false
APPENDIX C

SAMPLE DOCUMENTS FOR THE DOCUMENTATION VIEW

(1) Appendix C1: Sample CDD Tables ——— P369–380

(2) Appendix C2: Sample CISD Interface Specifications ——— P381–392
Appendix C: Prototype: View 3 – Documentation View
Appendix C1

Sample CGD Tables

This appendix presents sample tables that can be contained in a CGD. The following table that can be used as a quick reference for finding what components are in each framework as well as what files should be included in the user’s application such that it becomes possible to use a certain component. According to Schmidt in his report as the external examiner for this thesis, this kind of documents can take advantage of hyperlinks.

Table C1–1: List of ACE frameworks and their components

<table>
<thead>
<tr>
<th>FRAMEWORK</th>
<th>SOURCE-FILE</th>
<th>COMPONENT</th>
<th>BASE-CLASS (Component)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IO_Cntl_Msg.h</td>
<td>ACE_IO_Cntl_Msg</td>
<td>N/D</td>
</tr>
<tr>
<td>2</td>
<td>IO_Cntl_Msg.cpp</td>
<td>ACE_Module_Link</td>
<td>N/D</td>
</tr>
<tr>
<td>3</td>
<td>Map_Manager.h</td>
<td>ACE_Allocator</td>
<td>N/D</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>ACE_Map_Iterator</td>
<td>N/D</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>ACE_Map_Manager</td>
<td>N/D</td>
</tr>
<tr>
<td>6</td>
<td>Message_Block.h</td>
<td>ACE_Message_Block</td>
<td>N/D</td>
</tr>
<tr>
<td>7</td>
<td>Message_Queue.h</td>
<td>ACE_Message_Queue</td>
<td>N/D</td>
</tr>
<tr>
<td>8</td>
<td>Module.h</td>
<td>ACE_Module</td>
<td>N/D</td>
</tr>
<tr>
<td>9</td>
<td>Multiplexor.h</td>
<td>Driver</td>
<td>N/D</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>ACE_Multiplexor</td>
<td>N/D</td>
</tr>
<tr>
<td>11</td>
<td>Stream.h</td>
<td>ACE_Stream</td>
<td>N/D</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>ACE_Stream_Iterator</td>
<td>N/D</td>
</tr>
<tr>
<td>13</td>
<td>Stream_Modules.h</td>
<td>ACE_Stream_Head</td>
<td>public ACE_Task&lt;ACE_SYNCH_2&gt;</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>ACE_Stream_Tail</td>
<td>public ACE_Task&lt;ACE_SYNCH_2&gt;</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>ACE_Thru_Task</td>
<td>public ACE_Task&lt;ACE_SYNCH_2&gt;</td>
</tr>
<tr>
<td>16</td>
<td>Task.h</td>
<td>ACE_Task Flags</td>
<td>N/D</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>ACE_Task</td>
<td>public ACE_Service_Object</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>ACE_Task.Exit</td>
<td>N/D</td>
</tr>
</tbody>
</table>
### Table C1-1: List of ACE frameworks and their components (part 2)

<table>
<thead>
<tr>
<th>FRAMEWORK</th>
<th>SOURCE-FILE</th>
<th>COMPONENT</th>
<th>BASE-CLASS (Component)</th>
</tr>
</thead>
<tbody>
<tr>
<td>19 [CORBA]</td>
<td>CORBA_Handler.h</td>
<td>ACE_CORBA_Handler</td>
<td>public ACE_Service_Object</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_ST_CORBA_Handler</td>
<td>public ACE_CORBA_Handler</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_MT_CORBA_Handler</td>
<td>public ACE_CORBA_Handler, public CORBA_1 (ThreadFilter)</td>
</tr>
<tr>
<td></td>
<td>CORBA_Ref.h</td>
<td>ACE_CORBA_Ref</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Unbounded_Set_Iterator</td>
<td>N/D</td>
</tr>
<tr>
<td>22</td>
<td>Set.h</td>
<td>ACE_Set_Node</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Unbounded_Set</td>
<td>N/D</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>ACE_Fixed_Set</td>
<td>N/D</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>ACE_Fixed_Set_iterator</td>
<td>N/D</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td>ACE_Bounded_Set</td>
<td>N/D</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>ACE_Bounded_Set_iterator</td>
<td>N/D</td>
</tr>
<tr>
<td>27</td>
<td></td>
<td>ACE_Bounded_Set_iterator</td>
<td>N/D</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td>ACE_Bounded_Set_iterator</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td>Stack.cpp</td>
<td>ACE_Set_Node</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Bounded_Set</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Fixed_Set</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Unbounded_Stack</td>
<td>N/D</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>ACE_Stack_Node</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Queue_Node</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Unbounded_Queue</td>
<td>N/D</td>
</tr>
<tr>
<td>33</td>
<td>Stack.h</td>
<td>ACE_Bounded_Stack</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Fixed_Stack</td>
<td>N/D</td>
</tr>
<tr>
<td>34</td>
<td></td>
<td>ACE_Unbounded_Stack</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Queue_Node</td>
<td>N/D</td>
</tr>
<tr>
<td>35</td>
<td></td>
<td>ACE_Unbounded_Queue</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td>Stack.cpp</td>
<td>ACE_Bounded_Stack</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Fixed_Stack</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Unbounded_Stack</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Unbounded_Queue</td>
<td>N/D</td>
</tr>
<tr>
<td>37</td>
<td></td>
<td>ACE_Bounded_Stack_iterator</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Fixed_Stack_iterator</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Bounded_Stack_iterator</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Bounded_Queue</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Bounded_Queue</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Fixed_Stack_iterator</td>
<td>N/D</td>
</tr>
<tr>
<td>40</td>
<td></td>
<td>ACE_Bounded_Stack</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Bounded_Queue</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Fixed_Stack_iterator</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Bounded_Queue</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Bounded_Queue</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Fixed_Stack_iterator</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Bounded_Queue</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Bounded_Queue</td>
<td>N/D</td>
</tr>
<tr>
<td>43</td>
<td>Process_MANAGER.h</td>
<td>ACE_Process_Control</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Process_Manager</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Process_Control</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Process_Descriptor</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td>Synch.h</td>
<td>ACE_File_Lock</td>
<td>N/D</td>
</tr>
<tr>
<td>46</td>
<td></td>
<td>ACE_Time_Value</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Mutex</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Thread_Mutex</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Thread_Mutex_Guard</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Condition_Thread_Mutex</td>
<td>N/D</td>
</tr>
</tbody>
</table>

370
<table>
<thead>
<tr>
<th>FRAMEWORK</th>
<th>SOURCE-FILE</th>
<th>COMPONENT</th>
<th>BASE-CLASS (Component)</th>
</tr>
</thead>
<tbody>
<tr>
<td>53</td>
<td></td>
<td>ACERecursiveThreadMutex</td>
<td>N/D</td>
</tr>
<tr>
<td>54</td>
<td></td>
<td>ACE_RW_Mutex</td>
<td>N/D</td>
</tr>
<tr>
<td>55</td>
<td></td>
<td>ACE_RW_Thread_Mutex</td>
<td>public ACE_RW_Mutex</td>
</tr>
<tr>
<td>56</td>
<td></td>
<td>ACE_Semaphore</td>
<td>N/D</td>
</tr>
<tr>
<td>57</td>
<td>Synch.h</td>
<td>ACE_Process_Semaphore</td>
<td>public ACE_Semaphore</td>
</tr>
<tr>
<td>58</td>
<td></td>
<td>ACE_Thread_Semaphore</td>
<td>public ACE_Semaphore</td>
</tr>
<tr>
<td>59</td>
<td></td>
<td>ACE_Barrier</td>
<td>N/D</td>
</tr>
<tr>
<td>60</td>
<td></td>
<td>ACE_Process_BARRIER</td>
<td>public ACE_Barrier</td>
</tr>
<tr>
<td>61</td>
<td></td>
<td>ACE_Thread_BARRIER</td>
<td>public ACE_Barrier</td>
</tr>
<tr>
<td>62</td>
<td></td>
<td>ACE_Process_Mutex</td>
<td>N/D</td>
</tr>
<tr>
<td>63</td>
<td></td>
<td>ACE_RW_Process_Mutex</td>
<td>public ACE_Process_Mutex</td>
</tr>
<tr>
<td>64</td>
<td></td>
<td>ACE_Null_Mutex</td>
<td>N/D</td>
</tr>
<tr>
<td>65</td>
<td></td>
<td>ACE_Null_Process_Mutex</td>
<td>N/D</td>
</tr>
<tr>
<td>66</td>
<td></td>
<td>ACE_Null_Mutex_Guard</td>
<td>N/D</td>
</tr>
<tr>
<td>67</td>
<td>Synch_Options.h</td>
<td>ACE_Synch_Options</td>
<td>N/D</td>
</tr>
<tr>
<td>68</td>
<td></td>
<td>ACE_Time_Value</td>
<td>N/D</td>
</tr>
<tr>
<td>69</td>
<td></td>
<td>ACE_Atomic_Op</td>
<td>N/D</td>
</tr>
<tr>
<td>70</td>
<td></td>
<td>ACE_TSS</td>
<td>N/D</td>
</tr>
<tr>
<td>71</td>
<td></td>
<td>ACE_NULL_SYNCH</td>
<td>N/D</td>
</tr>
<tr>
<td>72</td>
<td></td>
<td>ACE_Guard</td>
<td>N/D</td>
</tr>
<tr>
<td>73</td>
<td></td>
<td>ACE_Write_Guard</td>
<td>public ACE_Guard&lt;LOCK&gt;</td>
</tr>
<tr>
<td>74</td>
<td></td>
<td>ACE_Read_Guard</td>
<td>public ACE_Guard&lt;LOCK&gt;</td>
</tr>
<tr>
<td>75</td>
<td></td>
<td>ACE_TSS_Guard</td>
<td>N/D</td>
</tr>
<tr>
<td>76</td>
<td></td>
<td>ACE_TSS_Write_Guard</td>
<td>ACE_TSS_Guard&lt;LOCK&gt;</td>
</tr>
<tr>
<td>77</td>
<td></td>
<td>ACE_TSS_Read_Guard</td>
<td>ACE_TSS_Guard&lt;LOCK&gt;</td>
</tr>
<tr>
<td>78</td>
<td></td>
<td>ACE_CONDITION</td>
<td>N/D</td>
</tr>
<tr>
<td>79</td>
<td></td>
<td>ACE_Process_Condition</td>
<td>public ACE_Condition&lt;MUTEX&gt;</td>
</tr>
<tr>
<td>80</td>
<td></td>
<td>ACE_Thread_Condition</td>
<td>public ACE_Condition&lt;MUTEX&gt;</td>
</tr>
<tr>
<td>81</td>
<td></td>
<td>ACE_MT_SYNCH</td>
<td>N/D</td>
</tr>
<tr>
<td>82</td>
<td>Thread.h</td>
<td>ACE_Thread</td>
<td>N/D</td>
</tr>
<tr>
<td>83</td>
<td></td>
<td>ACE_Thread_Manager</td>
<td>N/D</td>
</tr>
<tr>
<td>84</td>
<td></td>
<td>ACE_Thread_Descriptor</td>
<td>N/D</td>
</tr>
<tr>
<td>85</td>
<td></td>
<td>ACE_Thread_Control</td>
<td>N/D</td>
</tr>
<tr>
<td>86</td>
<td>Token.h</td>
<td>ACE-Token</td>
<td>N/D</td>
</tr>
<tr>
<td>FRAMEWORK</td>
<td>SOURCE-FILE</td>
<td>COMPONENT</td>
<td>BASE-CLASS (Component)</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>-----------------------</td>
<td>-----------------------------------------</td>
</tr>
<tr>
<td>87</td>
<td></td>
<td>ACE_Acceptor</td>
<td>public ACE_Service_Object</td>
</tr>
<tr>
<td>88</td>
<td>Acceptor.h</td>
<td>ACE_Strategy_Acceptor</td>
<td>public ACE_Acceptor&lt;SVC_HANDLER&gt;</td>
</tr>
<tr>
<td>89</td>
<td></td>
<td>ACE_oneshot_Acceptor</td>
<td>public ACE_Service_Object</td>
</tr>
<tr>
<td>90</td>
<td>Connector.h</td>
<td>ACE_Svc_Tuple</td>
<td>N/D</td>
</tr>
<tr>
<td>91</td>
<td></td>
<td>ACE_Connecto</td>
<td>public ACE_Service_Object</td>
</tr>
<tr>
<td>92</td>
<td>Dynamic_Service.h</td>
<td>ACE_Dynamic_Service</td>
<td>N/D</td>
</tr>
<tr>
<td>93</td>
<td></td>
<td>ACE_Concurrency_Strategy</td>
<td>N/D</td>
</tr>
<tr>
<td>94</td>
<td></td>
<td>ACE_Thread_Strategy</td>
<td>public ACE_Concurrency_Strategy&lt;SVC_HANDLER&gt;</td>
</tr>
<tr>
<td>95</td>
<td></td>
<td>ACE_Process_Strategy</td>
<td>public ACE_Concurrency_Strategy&lt;SVC_HANDLER&gt;</td>
</tr>
<tr>
<td>96</td>
<td></td>
<td>ACE_Accept_Strategy</td>
<td>N/D</td>
</tr>
<tr>
<td>97</td>
<td></td>
<td>ACE_Scheduling_Strategy</td>
<td>N/D</td>
</tr>
<tr>
<td>98</td>
<td>Strategies.h</td>
<td>ACE_Schedule_All_Reactive_Strategy</td>
<td>public ACE_Scheduling_Strategy&lt;SVC_HANDLER&gt;</td>
</tr>
<tr>
<td>99</td>
<td></td>
<td>ACE_Schedule_All_Thraeded_Strategy</td>
<td>public ACE_Scheduling_Strategy&lt;SVC_HANDLER&gt;</td>
</tr>
<tr>
<td>100</td>
<td></td>
<td>ACE_Creation_Strategy</td>
<td>N/D</td>
</tr>
<tr>
<td>101</td>
<td></td>
<td>ACE_Singleton_Strategy</td>
<td>public ACE_Creation_Strategy&lt;SVC_HANDLER&gt;</td>
</tr>
<tr>
<td>102</td>
<td></td>
<td>ACE_DLL_Strategy</td>
<td>public ACE_Creation_Strategy&lt;SVC_HANDLER&gt;</td>
</tr>
<tr>
<td>103</td>
<td>Svc_Handler.h</td>
<td>ACE_Dynamic</td>
<td>N/D</td>
</tr>
<tr>
<td>104</td>
<td></td>
<td>ACE_Svc_Handler</td>
<td>public ACE_Task&lt;ACE_SYNCH&gt;</td>
</tr>
<tr>
<td>105</td>
<td></td>
<td>ACE_Dumpable</td>
<td>N/D</td>
</tr>
<tr>
<td>106</td>
<td>Dump.h</td>
<td>ACE_Dumpable_Ptr</td>
<td>N/D</td>
</tr>
<tr>
<td>107</td>
<td></td>
<td>ACE_OD</td>
<td>N/D</td>
</tr>
<tr>
<td>108</td>
<td>Trace.h</td>
<td>ACE_Trace</td>
<td>N/D</td>
</tr>
<tr>
<td>109</td>
<td>Dump_Th</td>
<td>ACE_Dumpable_Adapter</td>
<td>public ACE_Dumpable</td>
</tr>
<tr>
<td>110</td>
<td>Pipe.h</td>
<td>ACE_Pipe</td>
<td>N/D</td>
</tr>
<tr>
<td>111</td>
<td>[IPC]</td>
<td>IO_SAP.h</td>
<td>N/D</td>
</tr>
<tr>
<td>112</td>
<td>[IPC]</td>
<td>DEV.h</td>
<td>public ACE_IO_SAP</td>
</tr>
<tr>
<td>113</td>
<td>[IO_SAP]</td>
<td>DEV_Connector.h</td>
<td>public ACE_DEV</td>
</tr>
<tr>
<td>114</td>
<td>[DEV_SAP]</td>
<td>DEV_IO.h</td>
<td>public ACE_DEV</td>
</tr>
<tr>
<td>FRAMEWORK</td>
<td>SOURCE-FILE</td>
<td>COMPONENT</td>
<td>BASE-CLASS (Component)</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------</td>
<td>------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>115 [IPC]</td>
<td>FILE.h</td>
<td>ACE_FILE</td>
<td>public ACE_IO_SAP</td>
</tr>
<tr>
<td>116 [IO_SAP]</td>
<td>FILE_Connector.h</td>
<td>ACE_FILE_Connector</td>
<td>public ACE_FILE</td>
</tr>
<tr>
<td>117 [FILE_SAP]</td>
<td>FILE_IO.h</td>
<td>ACE_FILE_IO</td>
<td>public ACE_FILE</td>
</tr>
<tr>
<td>118 [IPC]</td>
<td>IPC_SAP.h</td>
<td>ACE_IPC_SAP</td>
<td>N/D</td>
</tr>
<tr>
<td>119 [IPC]</td>
<td>Addr.h</td>
<td>ACE_Export ACE_Addr</td>
<td>N/D</td>
</tr>
<tr>
<td>120 [IPC]</td>
<td>DEV_Addr.h</td>
<td>ACE_DEV_Addr</td>
<td>public ACE_Addr</td>
</tr>
<tr>
<td>121 [IPC]</td>
<td>FILE_Addr.h</td>
<td>ACE_FILE_Addr</td>
<td>public ACE_Addr</td>
</tr>
<tr>
<td>122 [IPC]</td>
<td>INET_Addr.h</td>
<td>ACE_INET_Addr</td>
<td>public ACE_Addr</td>
</tr>
<tr>
<td>123 [IPC]</td>
<td>SPIDE_Addr.h</td>
<td>ACE_SPIE_Addr</td>
<td>public ACE_Addr</td>
</tr>
<tr>
<td>124 [IPC]</td>
<td>UNIX_Addr.h</td>
<td>ACE_UNIX_Addr</td>
<td>public ACE_Addr</td>
</tr>
<tr>
<td>125 [IPC]</td>
<td>FIFO.h</td>
<td>ACE_FIFO</td>
<td>public ACE_IPC_SAP</td>
</tr>
<tr>
<td>126 [IPC]</td>
<td>FIFO_Recv.h</td>
<td>ACE_FIFO_Recv</td>
<td>public ACE_FIFO</td>
</tr>
<tr>
<td>127 [IPC]</td>
<td>FIFO_Recv_Msg.h</td>
<td>ACE_FIFO_Recv_Msg</td>
<td>public ACE_FIFO</td>
</tr>
<tr>
<td>128 [IPC]</td>
<td>FIFO_Send.h</td>
<td>ACE_FIFO_Send</td>
<td>public ACE_FIFO</td>
</tr>
<tr>
<td>129 [IPC]</td>
<td>FIFO_Send_Msg.h</td>
<td>ACE_FIFO_Send_Msg</td>
<td>public ACE_FIFO</td>
</tr>
<tr>
<td>130 [IPC]</td>
<td>LSOCK.h</td>
<td>ACE_LSOCK</td>
<td>N/D</td>
</tr>
<tr>
<td>131 [IPC]</td>
<td>LSOCK_Acceptor.h</td>
<td>ACE_LSOCK_Acceptor</td>
<td>public ACE_LSOCK_Acceptor</td>
</tr>
<tr>
<td>132 [IPC]</td>
<td>LSOCK_CODgram.h</td>
<td>ACE_LSOCK_CODgram</td>
<td>public ACE_LSOCK_CODgram</td>
</tr>
<tr>
<td>133 [IPC]</td>
<td>LSOCK_Connector.h</td>
<td>ACE_LSOCK_Connector</td>
<td>public ACE_LSOCK_Connector</td>
</tr>
<tr>
<td>134 [IPC]</td>
<td>LSOCK_Dgram.h</td>
<td>ACE_LSOCK_Dgram</td>
<td>public ACE_LSOCK_Dgram, public ACE_LSOCK</td>
</tr>
<tr>
<td>135 [IPC]</td>
<td>LSOCK_Stream.h</td>
<td>ACE_LSOCK_Stream</td>
<td>public ACE_LSOCK_Stream, public ACE_LSOCK</td>
</tr>
<tr>
<td>136 [IPC]</td>
<td>SOCK.h</td>
<td>ACE.SOCK</td>
<td>public ACE_IPC_SAP</td>
</tr>
<tr>
<td>137 [IPC]</td>
<td>SOCK_Acceptor.h</td>
<td>ACE.SOCK_Acceptor</td>
<td>public ACE.SOCK_Acceptor</td>
</tr>
<tr>
<td>138 [IPC]</td>
<td>SOCK_CODgram.h</td>
<td>ACE.SOCK_CODgram</td>
<td>public ACE.SOCK_CODgram</td>
</tr>
<tr>
<td>139 [IPC]</td>
<td>SOCK_Connector.h</td>
<td>ACE.SOCK_Connector</td>
<td>public ACE.SOCK_Connector</td>
</tr>
<tr>
<td>140 [IPC]</td>
<td>SOCK_Dgram.h</td>
<td>ACE.SOCK_Dgram</td>
<td>public ACE.SOCK_Dgram</td>
</tr>
<tr>
<td>141 [IPC]</td>
<td>SOCK_Dgram_Bcast.h</td>
<td>ACE.SOCK_Dgram_Bcast</td>
<td>public ACE.SOCK_Dgram_Bcast</td>
</tr>
<tr>
<td>142 [IPC]</td>
<td>SOCK_Dgram_Mcast.h</td>
<td>ACE.SOCK_Dgram_Mcast</td>
<td>public ACE.SOCK_Dgram_Mcast</td>
</tr>
</tbody>
</table>
### Table C1-1: List of ACE frameworks and their components (part 6)

<table>
<thead>
<tr>
<th>FRAMEWORK</th>
<th>SOURCE-FILE</th>
<th>COMPONENT</th>
<th>BASE-CLASS (Component)</th>
</tr>
</thead>
<tbody>
<tr>
<td>143</td>
<td>[IPC] SOCK_IO.h</td>
<td>ACE.SOCK_IO</td>
<td>public ACE.SOCK</td>
</tr>
<tr>
<td>144</td>
<td>[IPC_SAP] SOCK_Stream.h</td>
<td>ACE.SOCK_Stream</td>
<td>public ACE.SOCK_IO</td>
</tr>
<tr>
<td>145</td>
<td>[IPC] SPIPE.h</td>
<td>ACE_SPIPE</td>
<td>public ACE_IPC_SAP</td>
</tr>
<tr>
<td>146</td>
<td>[IPC_SAP] SPIPE_Acceptor.h</td>
<td>ACE_SPIPE_Acceptor</td>
<td>public ACE_SPIPE</td>
</tr>
<tr>
<td>147</td>
<td>[SPIPE_SAP] SPIPE_Connector.h</td>
<td>ACE_SPIPE_Connector</td>
<td>public ACE_SPIPE</td>
</tr>
<tr>
<td>148</td>
<td></td>
<td>ACE_SPIPE_Stream</td>
<td>public ACE_SPIPE</td>
</tr>
<tr>
<td>149</td>
<td>[IPC] TLI.h</td>
<td>ACE_TLI</td>
<td>public ACE_IPC_SAP</td>
</tr>
<tr>
<td>150</td>
<td>[IPC_SAP] TLI_Acceptor.h</td>
<td>ACE_TLI_Request_Call</td>
<td>N/D</td>
</tr>
<tr>
<td>151</td>
<td>[TLI_SAP] TLI_Acceptor.cpp</td>
<td>ACE_TLI_Acceptor</td>
<td>public ACE_TLI</td>
</tr>
<tr>
<td>152</td>
<td></td>
<td>ACE_TLI_Request_Call</td>
<td>N/D</td>
</tr>
<tr>
<td>153</td>
<td>[IPC] UPIPE_Acceptor.h</td>
<td>ACE_UPIPE_Acceptor</td>
<td>public ACE_SPIPE_Acceptor</td>
</tr>
<tr>
<td>154</td>
<td>[UPIPE_SAP] UPIPE_Connector.h</td>
<td>ACE_UPIPE_Connector</td>
<td>public ACE_SPIPE</td>
</tr>
<tr>
<td>155</td>
<td></td>
<td>ACE_UPIPE_Stream</td>
<td>public ACE_SPIPE</td>
</tr>
<tr>
<td>156</td>
<td>[Log_Msg] Log_Msg.h</td>
<td>ACE_Log_Msg</td>
<td>N/D</td>
</tr>
<tr>
<td>157</td>
<td></td>
<td>ACE_Log_Record</td>
<td>N/D</td>
</tr>
<tr>
<td>158</td>
<td>[Memory] Mem_Map.h</td>
<td>ACE_Mem_Map</td>
<td>N/D</td>
</tr>
<tr>
<td>159</td>
<td>[Mem_Map] Alloc.h</td>
<td>ACE_ALLOCATOR</td>
<td>N/D</td>
</tr>
<tr>
<td>160</td>
<td>[Memory] Alloc.h</td>
<td>ACE_Name_Node</td>
<td>N/D</td>
</tr>
<tr>
<td>161</td>
<td>[Shared_Malloc] Alloc.h</td>
<td>ACE_Control_Block</td>
<td>N/D</td>
</tr>
<tr>
<td>162</td>
<td>[Memory] Alloc.h</td>
<td>ACE_ALLOCATOR_ADAPTER</td>
<td>public ACE_ALLOCATOR</td>
</tr>
<tr>
<td>163</td>
<td>[Shared_Malloc] Alloc.h</td>
<td>ACE_MALLOC_ITERATOR</td>
<td>N/D</td>
</tr>
<tr>
<td>164</td>
<td>Memory_Pool.h</td>
<td>ACE_Sbrk_Memory_Pool</td>
<td>N/D</td>
</tr>
<tr>
<td>165</td>
<td></td>
<td>ACE_SHARED_MEMORY_POOL</td>
<td>public ACE_EVENT_HANDLER</td>
</tr>
<tr>
<td>166</td>
<td></td>
<td>ACE_LOCAL_MEMORY_POOL</td>
<td>N/D</td>
</tr>
<tr>
<td>167</td>
<td></td>
<td>ACE_MMAPPED_MEMORY_POOL</td>
<td>public ACE_EVENT_HANDLER</td>
</tr>
<tr>
<td>168</td>
<td>[Memory] Shared_Memory.h</td>
<td>ACE_SHARED_MEMORY</td>
<td>N/D</td>
</tr>
<tr>
<td>169</td>
<td>[Shared_Memory] Shared_Memory_MM.h</td>
<td>ACE_SHARED_MEMORY_MM</td>
<td>public ACE_SHARED_MEMORY</td>
</tr>
<tr>
<td>170</td>
<td></td>
<td>ACE_SHARED_MEMORY_SV</td>
<td>public ACE_SHARED_MEMORY</td>
</tr>
</tbody>
</table>
## Table C1-1: List of ACE frameworks and their components (part 7)

<table>
<thead>
<tr>
<th>FRAMEWORK</th>
<th>SOURCE-FILE</th>
<th>COMPONENT</th>
<th>BASE-CLASS (Component)</th>
</tr>
</thead>
<tbody>
<tr>
<td>174</td>
<td>ARGV.h</td>
<td>ACE_ARGV</td>
<td>N/D</td>
</tr>
<tr>
<td>175</td>
<td></td>
<td>auto_ptr</td>
<td>N/D</td>
</tr>
<tr>
<td>176</td>
<td>Auto_Ptr.h</td>
<td>auto_array_ptr</td>
<td>N/D</td>
</tr>
<tr>
<td>177</td>
<td>Date_Time.h</td>
<td>ACE_Date_Time</td>
<td>N/D</td>
</tr>
<tr>
<td>178</td>
<td>Dynamic.h</td>
<td>ACE_Dynamic</td>
<td>N/D</td>
</tr>
<tr>
<td>179</td>
<td>Get_Opt.h</td>
<td>ACE_Get_Opt</td>
<td>N/D</td>
</tr>
<tr>
<td>180</td>
<td>Obstack.h</td>
<td>ACE_Obchunk</td>
<td>N/D</td>
</tr>
<tr>
<td>181</td>
<td></td>
<td>ACE_Obstack</td>
<td>N/D</td>
</tr>
<tr>
<td>182</td>
<td>Read_Buffer.h</td>
<td>ACE_Read_Buffer</td>
<td>N/D</td>
</tr>
<tr>
<td>183</td>
<td>Singleton.h</td>
<td>ACE_Singleton</td>
<td>N/D</td>
</tr>
<tr>
<td>[Misc]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>184</td>
<td>Local_Name_Space.h</td>
<td>ACE_Name_Space_Map</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Export ACE_NS_String</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Export ACE_NS_Internal</td>
<td>N/D</td>
</tr>
<tr>
<td>185</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>186</td>
<td>Local_Name_Space.cpp</td>
<td>ACE_Name_Space_Map</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Export ACE_Local_Name_Space</td>
<td>public ACE_Name_Space</td>
</tr>
<tr>
<td>187</td>
<td>Name_Proxy.h</td>
<td>ACE_Name_Proxy</td>
<td>public ACE_Name_Space</td>
</tr>
<tr>
<td></td>
<td>Name_Request_Reply.h</td>
<td>ACE_Name_Request</td>
<td>public ACE_Name_Space</td>
</tr>
<tr>
<td></td>
<td>ACE_Name_Reply</td>
<td>N/D</td>
<td></td>
</tr>
<tr>
<td>188</td>
<td>Name_Space.h</td>
<td>ACE_Name_Binding</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Name_Space</td>
<td>N/D</td>
</tr>
<tr>
<td>189</td>
<td>Naming_Context.h</td>
<td>ACE_Name_Options</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Naming_Context</td>
<td>public ACE_Name_Space</td>
</tr>
<tr>
<td></td>
<td>Remote_Name_Space.h</td>
<td>ACE_Remote_Name_Space</td>
<td>public ACE_Name_Space</td>
</tr>
<tr>
<td>[Name_Service]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>190</td>
<td>ACE.h</td>
<td>ACE_Time_Value</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE</td>
<td>N/D</td>
</tr>
<tr>
<td>191</td>
<td>OS.h</td>
<td>ACE_Static_Svc_##X</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_OS</td>
<td>N/D</td>
</tr>
<tr>
<td>192</td>
<td>OS.cpp</td>
<td>ACE_TSS_Ref</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_TSS_Info</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_TSS_Cleanup</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Win32_Thread_Adapter</td>
<td>N/D</td>
</tr>
<tr>
<td>FRAMEWORK</td>
<td>SOURCE-FILE</td>
<td>COMPONENT</td>
<td>BASE-CLASS (Component)</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>-----------</td>
<td>------------------------</td>
</tr>
<tr>
<td>205</td>
<td>Event_Handler.h</td>
<td>ACE_Message_Block</td>
<td>N/D</td>
</tr>
<tr>
<td>206</td>
<td>Event_Handler.h</td>
<td>ACE_Event_Handler</td>
<td>N/D</td>
</tr>
<tr>
<td>207</td>
<td>Event_Handler_T.h</td>
<td>ACE_Event_Handler_T</td>
<td>public ACE_Event_Handler</td>
</tr>
<tr>
<td>208</td>
<td>Handle_Set.h</td>
<td>ACE_Handle_Set</td>
<td>N/D</td>
</tr>
<tr>
<td>209</td>
<td>Handle_Set_iterator</td>
<td>ACE_Handle_Set_iterator</td>
<td>N/D</td>
</tr>
<tr>
<td>210</td>
<td>Proactor.h</td>
<td>ACE_Overlapped_File</td>
<td>N/D</td>
</tr>
<tr>
<td>211</td>
<td>Proactor.h</td>
<td>ACE_Overlapped_IO</td>
<td>N/D</td>
</tr>
<tr>
<td>212</td>
<td>Proactor.cpp</td>
<td>ACE_Proactor</td>
<td>public ACE_Event_Handler</td>
</tr>
<tr>
<td>213</td>
<td>Proactor.cpp</td>
<td>ACE_Overlapped_IO</td>
<td>public ACE_OVERLAPPED</td>
</tr>
<tr>
<td>214</td>
<td>Reactor.h</td>
<td>ACE_Reactor</td>
<td>N/D</td>
</tr>
<tr>
<td>215</td>
<td>Reactor.h</td>
<td>ACE_Notification_Handler</td>
<td>public ACE_Event_Handler</td>
</tr>
<tr>
<td>216</td>
<td>Reactor.h</td>
<td>ACE_Reactor-Token</td>
<td>public ACE.REACTOR_MUTEX</td>
</tr>
<tr>
<td>217</td>
<td>Reactor.h</td>
<td>ACE_Handler_Repository</td>
<td>N/D</td>
</tr>
<tr>
<td>218</td>
<td>Reactor.h</td>
<td>ACE_Handler_Repository_iterator</td>
<td>N/D</td>
</tr>
<tr>
<td>219</td>
<td>Signal.h</td>
<td>ACE_Sig_Set</td>
<td>N/D</td>
</tr>
<tr>
<td>220</td>
<td>Signal.h</td>
<td>ACE_Sig_Action</td>
<td>N/D</td>
</tr>
<tr>
<td>221</td>
<td>Signal.h</td>
<td>ACE_Sig_Guard</td>
<td>N/D</td>
</tr>
<tr>
<td>222</td>
<td>Signal.h</td>
<td>ACE_Sig_Handler</td>
<td>N/D</td>
</tr>
<tr>
<td>223</td>
<td>Signal.cpp</td>
<td>ACE_Sig_Handlers</td>
<td>public ACE_Sig_Handler</td>
</tr>
<tr>
<td>224</td>
<td>Signal.cpp</td>
<td>ACE_SH</td>
<td>public ACE_Event_Handler</td>
</tr>
<tr>
<td>225</td>
<td>Signal.cpp</td>
<td>ACE_Sig_Handlers_Set</td>
<td>N/D</td>
</tr>
<tr>
<td>226</td>
<td>Time_Value.h</td>
<td>ACE_Time_Value</td>
<td>N/D</td>
</tr>
<tr>
<td>227</td>
<td>Timer_Queue.h</td>
<td>ACE_Timer_Queue</td>
<td>N/D</td>
</tr>
<tr>
<td>228</td>
<td>ReactorEx.h</td>
<td>ACE_ReactorEx</td>
<td>N/D</td>
</tr>
<tr>
<td>229</td>
<td>ReactorEx.h</td>
<td>ACE_ReactorEx-Token</td>
<td>public ACE.REACTOREX_MUTEX</td>
</tr>
<tr>
<td>230</td>
<td>[Service_Configurator]</td>
<td>ACE_Parse_Node</td>
<td>N/D</td>
</tr>
<tr>
<td>231</td>
<td>Parse_Node.h</td>
<td>ACE_Suspend_Node</td>
<td>public ACE_Parse_Node</td>
</tr>
<tr>
<td>232</td>
<td>Parse_Node.h</td>
<td>ACE_Resume_Node</td>
<td>public ACE_Parse_Node</td>
</tr>
<tr>
<td>233</td>
<td>Parse_Node.h</td>
<td>ACE_Remove_Node</td>
<td>public ACE_Parse_Node</td>
</tr>
<tr>
<td>234</td>
<td>Parse_Node.h</td>
<td>ACE_Static_Node</td>
<td>public ACE_Parse_Node</td>
</tr>
<tr>
<td>235</td>
<td>Parse_Node.h</td>
<td>ACE_Dynamic_Node</td>
<td>public ACE_Static_Node</td>
</tr>
<tr>
<td>236</td>
<td>Parse_Node.h</td>
<td>ACE_Stream_Node</td>
<td>public ACE_Parse_Node</td>
</tr>
<tr>
<td>237</td>
<td>Parse_Node.h</td>
<td>ACE_Location_Node</td>
<td>N/D</td>
</tr>
<tr>
<td>238</td>
<td>Parse_Node.h</td>
<td>ACE_Object_Node</td>
<td>public ACE_Location_Node</td>
</tr>
</tbody>
</table>
Table C1-1: List of ACE frameworks and their components (part 9)

<table>
<thead>
<tr>
<th>FRAMEWORK</th>
<th>SOURCE-FILE</th>
<th>COMPONENT</th>
<th>BASE-CLASS (Component)</th>
</tr>
</thead>
<tbody>
<tr>
<td>239</td>
<td>Parse_Node.h</td>
<td>ACE_Function_Node</td>
<td>public ACE_Location_Node</td>
</tr>
<tr>
<td>240</td>
<td></td>
<td>ACE_Dummy_Node</td>
<td>public ACE_Parse_Node</td>
</tr>
<tr>
<td>241</td>
<td>Service_Config.h</td>
<td>ACE_Service_Repository</td>
<td>N/D</td>
</tr>
<tr>
<td>242</td>
<td></td>
<td>ACE_Service_Record</td>
<td>N/D</td>
</tr>
<tr>
<td>243</td>
<td>Service_Manager.h</td>
<td>ACE Allocator</td>
<td>N/D</td>
</tr>
<tr>
<td>244</td>
<td>Service_Object.h</td>
<td>ACE_Reactor</td>
<td>N/D</td>
</tr>
<tr>
<td>245</td>
<td></td>
<td>ACE_Service_Config</td>
<td>public ACE_Event_Handle</td>
</tr>
<tr>
<td>[Service_Configurator]</td>
<td>Service_Record.h</td>
<td>ACE_Service_Object</td>
<td>public ACE_Event_Handler,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>public ACE_Shared_Object</td>
</tr>
<tr>
<td>247</td>
<td>Service_Record.h</td>
<td>ACE_Service_Type</td>
<td>N/D</td>
</tr>
<tr>
<td>248</td>
<td>Service_Record.h</td>
<td>ACE_Service_Object_Type</td>
<td>public ACE_Service_Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Module_Type</td>
<td>public ACE_Service_Type</td>
</tr>
<tr>
<td>249</td>
<td></td>
<td>ACE_Stream_Type</td>
<td>public ACE_Service_Type</td>
</tr>
<tr>
<td>250</td>
<td>Service_Repository.h</td>
<td>ACE_Service_Record</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACE_Service_Repository</td>
<td>N/D</td>
</tr>
<tr>
<td>251</td>
<td>Shared_Object.h</td>
<td>ACE_Service_Repository_</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Iterator</td>
<td></td>
</tr>
<tr>
<td>252</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>253</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>254</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>255</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>SV_Message.h</td>
<td>ACE_SV_Message</td>
<td>N/D</td>
</tr>
<tr>
<td>[System_V_IPC]</td>
<td>SV_Message_Queue.h</td>
<td>ACE_SV_Message_Queue</td>
<td>N/D</td>
</tr>
<tr>
<td>[System_V_Message_Queue]</td>
<td>Typed_SV_Message.h</td>
<td>ACE_TYPED_SV_Message</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td>Typed_SV_Message_Queue.h</td>
<td>ACE_TYPED_SV_Message_Queue</td>
<td>N/D</td>
</tr>
<tr>
<td>258</td>
<td>SV_Semaphore_Complex.h</td>
<td>ACE_SV_Semaphore_Complex</td>
<td>private ACE_SV_Semaphore_Simple</td>
</tr>
<tr>
<td>[System_V_IPC]</td>
<td>SV_Semaphore_Simple.h</td>
<td>ACE_SV_Semaphore_Simple</td>
<td>N/D</td>
</tr>
<tr>
<td>[System_V_Semaphores]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>260</td>
<td>SV_Shared_Memory.h</td>
<td>ACE_SV_Shared_Memory</td>
<td>N/D</td>
</tr>
<tr>
<td>[System_V_IPC]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SV_Shared_Memory.h</td>
<td>ACE_SV_Shared_Memory</td>
<td>N/D</td>
</tr>
<tr>
<td>[System_V_Shared_Memory]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>262</td>
<td>High_Res_Timer.h</td>
<td>ACE_High_Res_Timer</td>
<td>N/D</td>
</tr>
<tr>
<td>[Timers]</td>
<td>Profile_Timer.h</td>
<td>ACE_Profile_Timer</td>
<td>N/D</td>
</tr>
<tr>
<td></td>
<td>Time_Request_Reply.h</td>
<td>ACE_Time_Request</td>
<td>N/D</td>
</tr>
</tbody>
</table>

377
Table C1-1: List of ACE frameworks and their components (part 10)

<table>
<thead>
<tr>
<th>FRAMEWORK</th>
<th>SOURCE-FILE</th>
<th>COMPONENT</th>
<th>BASE-CLASS (Component)</th>
</tr>
</thead>
<tbody>
<tr>
<td>266</td>
<td>[Timers]</td>
<td>System_Time.h</td>
<td></td>
</tr>
<tr>
<td>267</td>
<td></td>
<td>ACE_Date_Time</td>
<td>N/D</td>
</tr>
<tr>
<td>267</td>
<td></td>
<td>ACE_System_Time</td>
<td>N/D</td>
</tr>
<tr>
<td>268</td>
<td></td>
<td>ACE_TOKEN_CONST</td>
<td>N/D</td>
</tr>
<tr>
<td>269</td>
<td></td>
<td>ACE_Token_Proxy</td>
<td>N/D</td>
</tr>
<tr>
<td>270</td>
<td></td>
<td>ACE_TPOQ_Entry</td>
<td>N/D</td>
</tr>
<tr>
<td>271</td>
<td></td>
<td>ACE_TSS_TPOQ_Entry</td>
<td>public ACE_TSS&lt;ACE_TPOQ_Entry&gt;</td>
</tr>
<tr>
<td>272</td>
<td></td>
<td>ACE_Token_Proxy_Queue</td>
<td>N/D</td>
</tr>
<tr>
<td>273</td>
<td></td>
<td>ACE_ExportACE_Tokens</td>
<td>N/D</td>
</tr>
<tr>
<td>274</td>
<td></td>
<td>ACE_Local_Mutex</td>
<td>N/D</td>
</tr>
<tr>
<td>275</td>
<td></td>
<td>ACE_Mutex_Token</td>
<td>public ACE_Tokens</td>
</tr>
<tr>
<td>276</td>
<td></td>
<td>ACE_RW_Token</td>
<td>public ACE_Tokens</td>
</tr>
<tr>
<td>277</td>
<td></td>
<td>ACE_Token_Proxy</td>
<td>N/D</td>
</tr>
<tr>
<td>278</td>
<td></td>
<td>ACE_Null_Token</td>
<td>public ACE_Token_Proxy</td>
</tr>
<tr>
<td>279</td>
<td></td>
<td>ACE_Local_Mutex</td>
<td>public ACE_Token_Proxy</td>
</tr>
<tr>
<td>280</td>
<td></td>
<td>ACE_Local_RLock</td>
<td>public ACE_Token_Proxy</td>
</tr>
<tr>
<td>281</td>
<td></td>
<td>ACE_Local_WLock</td>
<td>public ACE_Token_Proxy</td>
</tr>
<tr>
<td>282</td>
<td></td>
<td>Local_Tokens_T.h</td>
<td></td>
</tr>
<tr>
<td>283</td>
<td></td>
<td>ACE_Token_Name</td>
<td>N/D</td>
</tr>
<tr>
<td>284</td>
<td></td>
<td>ACE_Remote_Token_Proxy</td>
<td>public ACE_Token_Proxy</td>
</tr>
<tr>
<td>285</td>
<td></td>
<td>ACE_RRemote_Mutex</td>
<td>public ACE_RRemote_Token_Proxy</td>
</tr>
<tr>
<td>286</td>
<td></td>
<td>ACE_REMOTE_RLock</td>
<td>public ACE_REMOTE_Token_Proxy</td>
</tr>
<tr>
<td>287</td>
<td></td>
<td>ACE_REMOTE_WLock</td>
<td>public ACE_REMOTE_Token_Proxy</td>
</tr>
<tr>
<td>288</td>
<td></td>
<td>Token_Collection.h</td>
<td></td>
</tr>
<tr>
<td>289</td>
<td></td>
<td>ACE_TSS_Connection</td>
<td>public ACE_TSS&lt;ACE.SOCK Stream&gt;</td>
</tr>
<tr>
<td>290</td>
<td></td>
<td>Token_Manager.h</td>
<td></td>
</tr>
<tr>
<td>291</td>
<td></td>
<td>ACE_Token_Request</td>
<td>N/D</td>
</tr>
<tr>
<td>292</td>
<td></td>
<td>ACE_Token_Request_Reply.h</td>
<td>N/D</td>
</tr>
<tr>
<td>293</td>
<td></td>
<td>ACE_Token_Reply</td>
<td>N/D</td>
</tr>
</tbody>
</table>

? : This File is not listed under any category in the original document of the ACE categories. This original document will be included as is in the appendices.
N/D : Not Derived from another class
↓ : To continue the cell on the next page
The following is a sample table that can be used as a quick reference for identifying what service(s) can be provided by a certain component(s).

Table C–2: Component–Indexed Table

<table>
<thead>
<tr>
<th>COMPONENT NAMES</th>
<th>Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ComponentName1</td>
<td>Service of Component1</td>
</tr>
<tr>
<td>2 ComponentName2</td>
<td>Service of Component2</td>
</tr>
<tr>
<td>3 ComponentName3</td>
<td>Service of Component3</td>
</tr>
<tr>
<td>......</td>
<td>......</td>
</tr>
<tr>
<td>......</td>
<td>......</td>
</tr>
<tr>
<td>......</td>
<td>......</td>
</tr>
<tr>
<td>......</td>
<td>......</td>
</tr>
<tr>
<td>ComponentNameN</td>
<td>Service of ComponentN</td>
</tr>
</tbody>
</table>

The following is a sample table that can be used as a quick reference for identifying which component(s) can provide a certain service(s).

Table C–3: Service–Indexed Table

<table>
<thead>
<tr>
<th>SERVICE NAMES</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ServiceName1</td>
<td>Component of Service1</td>
</tr>
<tr>
<td>2 ServiceName2</td>
<td>Component of Service2</td>
</tr>
<tr>
<td>3 ServiceName3</td>
<td>Component of Service3</td>
</tr>
<tr>
<td>......</td>
<td>......</td>
</tr>
<tr>
<td>......</td>
<td>......</td>
</tr>
<tr>
<td>......</td>
<td>......</td>
</tr>
<tr>
<td>......</td>
<td>......</td>
</tr>
<tr>
<td>ServiceNameN</td>
<td>Component of ServiceN</td>
</tr>
</tbody>
</table>
Appendix C2

Sample CISD Interface Specifications

C2.1. Overview

A CO-Toolkit’s CISD is intended to provide a precise description (specification) for the interface of each component that is provided in that CO-toolkit. This appendix presents a sample interface specifications for one of the many components (i.e. ACE_Message_Block of the ASX framework) found in ACE.

C2.2. Interface Specification

C2.2.1. CHARACTERISTICS

*Type Specified:* ACE_Message_Block

*Public Primitive System Data Types:*

<table>
<thead>
<tr>
<th>Types</th>
<th>Informal Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>size_t</td>
<td>sizes of objects (such as strings) (unsigned)</td>
</tr>
<tr>
<td>u_long</td>
<td>unsigned long</td>
</tr>
</tbody>
</table>

*Inherited Public Primitive System Data Types:* None

*Public Specification Constant Parameters:* The following table shows all the specification parameters that are used in this class;

381
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Informal Description</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>#DONT_DELETE#</td>
<td>Don’t delete the data on exit since we don’t own it</td>
<td>Constant parameters</td>
</tr>
<tr>
<td>#USER_FLAGS#</td>
<td>user defined flags start here</td>
<td></td>
</tr>
<tr>
<td>#MB_DATA#</td>
<td>regular data</td>
<td>Data and protocol messages</td>
</tr>
<tr>
<td>#MBPROTO#</td>
<td>protocol control</td>
<td>(regular and priority)</td>
</tr>
<tr>
<td>#MB_BREAK#</td>
<td>line break</td>
<td>Control messages</td>
</tr>
<tr>
<td>#MB_PASSFP#</td>
<td>pass file pointer</td>
<td>(regular and priority)</td>
</tr>
<tr>
<td>#MB_EVENT#</td>
<td>post an event to an event queue</td>
<td>Control messages</td>
</tr>
<tr>
<td>#MB_SIG#</td>
<td>generate process signal</td>
<td>(high priority: go to head of queue)</td>
</tr>
<tr>
<td>#MB_IOCTL#</td>
<td>ioctl; set/get params</td>
<td></td>
</tr>
<tr>
<td>#MB_SETOPTS#</td>
<td>set various stream head options</td>
<td>Message class masks</td>
</tr>
<tr>
<td>#MB_IOCACK#</td>
<td>acknowledge ioctl</td>
<td></td>
</tr>
<tr>
<td>#MB_IOCNAK#</td>
<td>negative ioctl acknowledge</td>
<td></td>
</tr>
<tr>
<td>#MB_PCPROTO#</td>
<td>priority proto message</td>
<td></td>
</tr>
<tr>
<td>#MB_PCSIG#</td>
<td>generate process signal</td>
<td></td>
</tr>
<tr>
<td>#MB_READ#</td>
<td>generate read notification</td>
<td></td>
</tr>
<tr>
<td>#MB_FLUSH#</td>
<td>flush your queues</td>
<td></td>
</tr>
<tr>
<td>#MB_STOP#</td>
<td>stop transmission immediately</td>
<td></td>
</tr>
<tr>
<td>#MB_START#</td>
<td>restart transmission after stop</td>
<td></td>
</tr>
<tr>
<td>#MB_HANGUP#</td>
<td>line disconnect</td>
<td></td>
</tr>
<tr>
<td>#MB_ERROR#</td>
<td>fatal error used to set u.u_error</td>
<td></td>
</tr>
<tr>
<td>#MB_PCEVENT#</td>
<td>post an event to an event queue</td>
<td></td>
</tr>
<tr>
<td>#MB_NORMAL#</td>
<td>Normal priority messages</td>
<td></td>
</tr>
<tr>
<td>#MB_PRIORITY#</td>
<td>High priority control messages</td>
<td></td>
</tr>
</tbody>
</table>

**Public Specification Types:**

ACE_Message_Type: enum = { #MB_DATA#, #MBPROTO#, #MB_BREAK#, #MB_PASSFP#, #MB_EVENT#, #MB_SIG#, #MB_IOCTL#, #MB_SETOPTS#, #MB_IOCACK#, #MB_IOCNAK#, #MB_PCPROTO#, #MB_PCSIG#, #MB_READ#, #MB_FLUSH#, #MB_STOP#, #MB_START#, #MB_HANGUP#, #MB_ERROR#, #MB_PCEVENT#, #MB_NORMAL#, #MB_PRIORITY# }

Message Flags: u_long

**Public Specification Variables:** None
C2.2.2. SYNTAX

The Declaration of this class is in the Message_Block.h header file, while the implementations of its access programs are in the Message_Block.i and Message_Block.cpp header files.

C2.2.2.1. Public ACCESS PROGRAMS

<table>
<thead>
<tr>
<th>Program Name</th>
<th>Value</th>
<th>Arg #1</th>
<th>Arg #2</th>
<th>Arg #3</th>
<th>Arg #4</th>
<th>Arg #5</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACE_Message_Block</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACE_Message_Block</td>
<td>char*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>char*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACE_Message_Block</td>
<td>size_t</td>
<td>size_t</td>
<td>ACE_Message_Type</td>
<td>ACE_Message_Block*</td>
<td>char*</td>
<td>ACE_Allocator*</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>~ACE_Message_Block</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>is_data_msg</td>
<td>int</td>
<td>const</td>
<td>: O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>const</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>is_data_msg</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>msg_class</td>
<td>ACE_Message_Type</td>
<td>const</td>
<td>: O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>const</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>msg_class</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>msg_type</td>
<td>ACE_Message_Type</td>
<td>const</td>
<td>: O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>const</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>msg_type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Informal Description
Header File

Message_Block.cpp

Delete all the resources held in the message.
Message_Block.cpp

Find out what type of message this is.
Message_Block.i

Find out what class of message this is (there are two classes, <normal> messages and <high-priority> messages).
Message_Block.i

Get type of the message.
Message_Block.i
<table>
<thead>
<tr>
<th>msg_type</th>
<th>ACE Message Type</th>
<th>Set type of the message.</th>
<th>Message Block.i</th>
</tr>
</thead>
<tbody>
<tr>
<td>set_flags</td>
<td>Message Flags</td>
<td>O</td>
<td>Message Flags</td>
</tr>
<tr>
<td>clr_flags</td>
<td>Message Flags</td>
<td>O</td>
<td>Message Flags</td>
</tr>
<tr>
<td>flags</td>
<td>Message Flags</td>
<td>const</td>
<td>O</td>
</tr>
<tr>
<td>msg_priority</td>
<td>u_long</td>
<td>const</td>
<td>O</td>
</tr>
<tr>
<td>msg_priority</td>
<td>u_long</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clone</td>
<td>ACE Message Block*</td>
<td>O</td>
<td>Message Flags</td>
</tr>
<tr>
<td>copy</td>
<td>int</td>
<td>const char*</td>
<td>size_t</td>
</tr>
<tr>
<td>base</td>
<td>char* const</td>
<td>O</td>
<td>Get message data.</td>
</tr>
<tr>
<td>base</td>
<td>char*</td>
<td>size_t</td>
<td>Message Flags</td>
</tr>
<tr>
<td>end</td>
<td>char* const</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td>Specification</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td>rd_ptr</td>
<td>Get the read pointer.</td>
<td>Message_Block.i</td>
<td></td>
</tr>
<tr>
<td>rd_ptr</td>
<td>Set the read pointer to &lt;ptr&gt;.</td>
<td>Message_Block.i</td>
<td></td>
</tr>
<tr>
<td>rd_ptr</td>
<td>Set the read pointer ahead &lt;n&gt; bytes.</td>
<td>Message_Block.i</td>
<td></td>
</tr>
<tr>
<td>wr_ptr</td>
<td>Get the write pointer.</td>
<td>Message_Block.i</td>
<td></td>
</tr>
<tr>
<td>wr_ptr</td>
<td>Set the write pointer to &lt;ptr&gt;.</td>
<td>Message_Block.i</td>
<td></td>
</tr>
<tr>
<td>wr_ptr</td>
<td>Set the write pointer ahead &lt;n&gt; bytes.</td>
<td>Message_Block.i</td>
<td></td>
</tr>
<tr>
<td>length</td>
<td>The length of a message is computed as the length between the wr_ptr() - rd_ptr(). This is to Get the length of the message.</td>
<td>Message_Block.i</td>
<td></td>
</tr>
<tr>
<td>length</td>
<td>The length of a message is computed as the length between the wr_ptr() - rd_ptr(). This is to Set the length of the message.</td>
<td>Message_Block.i</td>
<td></td>
</tr>
<tr>
<td>size</td>
<td>The size of the allocated buffer is the total amount of space allocated. This is to Get the total amount of space in the message.</td>
<td>Message_Block.i</td>
<td></td>
</tr>
<tr>
<td>size</td>
<td>The size of the allocated buffer is the total amount of space allocated. This is to Set the total amount of space in the message. Returns 0 if successful, else -1.</td>
<td>Message_Block.i</td>
<td></td>
</tr>
<tr>
<td>cont</td>
<td>The continuation field is used to chain together composite messages. This is to Get the continuation field.</td>
<td>Message_Block.i</td>
<td></td>
</tr>
</tbody>
</table>

385
<table>
<thead>
<tr>
<th>cont</th>
<th>ACE_ Message_Block*</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The continuation field is used to chain together composite messages. This is to Set the continuation field.</td>
</tr>
<tr>
<td>next</td>
<td>ACE_ Message_Block*</td>
</tr>
<tr>
<td></td>
<td>const : O</td>
</tr>
<tr>
<td></td>
<td>The &lt;next_&gt; pointer points to the &lt;Message_Block&gt; directly ahead in the Message_Queue. This is to Get link to next message.</td>
</tr>
<tr>
<td>next</td>
<td>ACE_ Message_Block*</td>
</tr>
<tr>
<td></td>
<td>The &lt;next_&gt; pointer points to the &lt;Message_Block&gt; directly ahead in the Message_Queue. This is to Set link to next message.</td>
</tr>
<tr>
<td>prev</td>
<td>ACE_ Message_Block*</td>
</tr>
<tr>
<td></td>
<td>const : O</td>
</tr>
<tr>
<td></td>
<td>The &lt;prev_&gt; pointer points to the &lt;Message_Block&gt; directly ahead in the Message_Queue. This is to Get link to prev message.</td>
</tr>
<tr>
<td>prev</td>
<td>ACE_ Message_Block*</td>
</tr>
<tr>
<td></td>
<td>The &lt;prev_&gt; pointer points to the &lt;Message_Block&gt; directly ahead in the Message_Queue. This is to Set link to prev message.</td>
</tr>
<tr>
<td>dump</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dump the state of an object.</td>
</tr>
</tbody>
</table>

**Note 1:** The const after the type of the access program means that the access program is not allowed to change the private data structure of this class.

**Note 2:** The const before the type of the access program means that the value returned by this access program can be assigned to a constant parameter.

**Note 3:** : O indicates that that function returns a value, and will be listed under the VALUES section.
C2.2.2.2. Inherited Public ACCESS PROGRAMS: None

C2.2.3. CANONICAL REPRESENTATION

C2.2.3.1. CANONICAL REPRESENTATION of This Component/Class

\[ S \equiv \{ <\text{Message Block Field}_{i=1}^{12}> \mid <\text{Fields Types}_{i=1}^{12}> \} \cup \{ \langle \rangle \} \]

**Lexicon for S:**

- `<Message Block Field> = <flags_, base_, cur_size_, max_size_, rd_ptr_, wr_ptr_, type_, priority_, cont_, next_, prev_, allocator_>`
- `<Fields Types> = <Message Flags, char*, size_, size_, char*, char*, ACE_Message_Type, u_long, ACE_Message_Block*, ACE_Message_Block*, ACE_Allocator*>`

C2.2.3.2. References To CANONICAL REPRESENTATIONs Of Inherited Components/Classes

None

C2.2.4. EQUIVALENCES

C2.2.4.1. EQUIVALENCES of This Component

In this section, I will specify the changes to the current state of an object (i.e. denoted by s), that are made by the different access programs in this class module.

\[
\text{ACE Message Block ([s]),} = (s = \langle \text{\"0\"}, \text{\"0\"}, \text{\"0\"}, \text{\"0\"}, \text{\"0\"}, \text{\#MB_NORMAL\#}, \text{\"0\"}, \text{\"0\"}, \text{\"0\"}, \text{\"0\"}>)
\]

\[
\text{ACE Message Block([s], data)} = (s = \langle \text{\#DON\_DELETE\#}, \text{data}, \text{\"0\"}, \text{\"0\"}, \text{\"0\"}, \text{\"0\"}, \text{\#MB_NORMAL\#}, \text{\"0\"}, \text{\"0\"}, \text{\"0\"}, \text{\"0\"}>)
\]

\[
\text{ACE Message Block([s], size, msg type, msg cont, msg alloc, alloc)} =
\]

<table>
<thead>
<tr>
<th>Condition</th>
<th>EQUIVALENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>(msg_data = 0)</code></td>
<td><code>(s = \langle \text{\&quot;0\&quot;}, \text{NAlloc, size, NAlloc, NAlloc, msg type, \&quot;0\&quot;}, \text{msg cont, \&quot;0\&quot;}, \text{\&quot;0\&quot;}, \text{\&quot;0\&quot;}&gt;)</code></td>
</tr>
<tr>
<td>(alloc=0)</td>
<td><strong>Lexicon:</strong> \n</td>
</tr>
<tr>
<td><code>(alloc\#0)</code></td>
<td><code>(s = \langle \text{\&quot;0\&quot;}, \text{Malloc, size, MAlloc, MAlloc, msg type, \&quot;0\&quot;}, \text{msg cont, \&quot;0\&quot;}, \text{\&quot;0\&quot;}, \text{alloc}&gt;)</code></td>
</tr>
<tr>
<td></td>
<td><strong>Lexicon:</strong> \n</td>
</tr>
<tr>
<td><code>(msg_data \#0)</code></td>
<td><code>(s = \langle \text{\#DON\_DELETE\#}, \text{msg data}, \text{size, size, msg data, msg data, \text{msg data, msg type, \&quot;0\&quot;}, \text{msg cont, \&quot;0\&quot;}, \text{\&quot;0\&quot;}&gt;})</code></td>
</tr>
</tbody>
</table>

387
~ACE_Message_Block ([s,]) \equiv (s = \infty)

is_data_msg([s,]) \equiv NC(s)

msg_class([s,]) \equiv NC(s)

msg_type([s,]) \equiv NC(s)

msg_type([s,], type) \equiv (s \mid \((s\_type\_field = \text{type}) \land NC(s\_nontype\_fields)))

set_flags([s,], more_flags) \equiv (s \mid \((s\_flags\_field = \text{more\_flags}) \land NC(s\_nonflags\_fields)))

clr_flags([s,], less_flags) \equiv (s \mid \((s\_flags\_field = \text{less\_flags}) \land NC(s\_nonflags\_fields)))

flags([s,]) \equiv NC(s)

msg\_priority([s,]) \equiv NC(s)

msg\_priority([s,], priority) \equiv \((s\_Priority\_field = \text{priority}) \land NC(s\_nonpriority\_fields)))

clone([s,], mask) \equiv NC(s)

copy([s,], buf, n) \equiv NC(s)

base([s,]) \equiv NC(s)

base([s,], msg\_data, msg\_length, msg\_flags) = (s =<\text{msg\_flags, msg\_data, msg\_length, msg\_length, msg\_data, msg\_data, "0", "0", "0", "0", "0", "0">)

der([s,]) \equiv NC(s)

dr_ptr([s,]) \equiv NC(s)

dr_ptr([s,], new\_ptr) \equiv (s \mid \((s\_RdPtr\_field = \text{new\_ptr}) \land NC(s\_nonRdPtr\_fields)))

dr_ptr([s,], n) \equiv (s \mid \((s\_RdPtr\_field = s\_RdPtr\_field + n) \land NC(s\_nonRdPtr\_fields)))

wr_ptr([s,]) \equiv NC(s)

wr_ptr([s,], new\_ptr) \equiv (s \mid \((s\_WrPtr\_field = \text{new\_ptr}) \land NC(s\_nonWrPtr\_fields)))

wr_ptr([s,], n) \equiv (s \mid \((s\_WrPtr\_field = s\_WrPtr\_field + n) \land NC(s\_nonWrPtr\_fields)))

length([s,]) \equiv NC(s)

length([s,], len) \equiv (s \mid \((s\_WrPtr\_field = s\_RdPtr\_field + \text{len}) \land NC(s\_nonWrPtr\_fields)))
Appendix C2: Sample CISD Interface Specifications

size([s,]) = NC(s)

size([s, len]) =

<table>
<thead>
<tr>
<th>Condition</th>
<th>EQUivalence</th>
</tr>
</thead>
<tbody>
<tr>
<td>(len &lt; s_maxsize_field)</td>
<td>((s_CurSize_field = len) ∧ NC(s_nonCurSize_fields))</td>
</tr>
<tr>
<td>(len ≥ s_maxsize_field)</td>
<td>(s =&lt; NC(Fld1), nBase, nCurSize, nMaxSize, nRdPtr, nWpTR, NC(Fld4), NC(Fld4), NC(Fld6), NC(Fld10), NC(Fld11), NC(Fld12)&gt;</td>
</tr>
</tbody>
</table>

Private Lexicon:

nBase = a new address randomly allocated by the system
nCurSize = nMaxSize = len
nRdPtr = nBase + rd_ptr - base_
nWpTR = nBase + wr_ptr - base_
Fld4 = Message_Block_Field

cont([s,]) = NC(s)

cont([s,], next_block) = (s | ((s_Curfield = next_block) ∧ NC(s_nonCont_fields)))

next([s,]) = NC(s)

next([s,], next_block) = (s | ((s_Next_field = next_block) ∧ NC(s_nonNext_fields)))

prev([s,]) = NC(s)

prev([s,], next_block) = (s | ((s_Prev_field = next_block) ∧ NC(s_nonPrev_fields)))

dump([s,]) = (s =< "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0">)

C2.2.4.2. References To EQUivalences Of Inherited Components

None

C2.2.5. VALUES

C2.2.5.1. VALUES of This Component

| o(is_data_msg([s,])) = | x, x ∈ [MB_DATA#, MB_PROTO#, MB_PCPROTO#] |
| o(msg_class([s,])) = | x, x ∈ [MB_PRIORITY#, MB_NORMAL#] |
| o(msg_type([s,])) = | x, x ∈ ACE_Message_Type enum |
| o(set_flags([s,], more_flags)) = | VAL(s_flags_field) |
| o(set_flags([s,], less_flags)) = | VAL(s_flags_field), VAL(s_flags_field) = more_flags |

389
C2.2.5.2. References To VALUES Of Inherited ACCESS Programs

None

C2.2.6. LEXICON

ERROR(%Error Message%) = An auxiliary function that displays the %Error Message%.

NC(x) = (x' = 'x)

VAL(x) = An auxiliary function that returns the value stored in the variable x

NumFields = 12
<table>
<thead>
<tr>
<th>Field Order</th>
<th>Field Type</th>
<th>Equivalent Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>RdPtrFieldOrder</td>
<td>s_RdPtr_field</td>
<td>Message_Block_FieldRdPtrFieldOrder</td>
</tr>
<tr>
<td>WrPtrFieldOrder</td>
<td>s_WrPtr_field</td>
<td>Message_Block_FieldWrPtrFieldOrder</td>
</tr>
<tr>
<td>TypeFieldOrder</td>
<td>s_type_field</td>
<td>Message_Block_FieldTypeFieldOrder</td>
</tr>
<tr>
<td>PriorityFieldOrder</td>
<td>s_priority_field</td>
<td>Message_Block_FieldPriorityFieldOrder</td>
</tr>
<tr>
<td>ContFieldOrder</td>
<td>s_Cont_field</td>
<td>Message_Block_FieldContFieldOrder</td>
</tr>
<tr>
<td>NextFieldOrder</td>
<td>s_Next_field</td>
<td>Message_Block_FieldNextFieldOrder</td>
</tr>
<tr>
<td>PrevFieldOrder</td>
<td>s_Prev_field</td>
<td>Message_Block_FieldPrevFieldOrder</td>
</tr>
<tr>
<td>AllocatorFieldOrder</td>
<td>s_Allocator_field</td>
<td>Message_Block_FieldAllocatorFieldOrder</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Predicates</th>
<th>Equivalences</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_nonflags_fields</td>
<td>Message_Block_Fieldₖ, ((1 ≤ i ≤ NumFields) ∧ (i ≠ FlagsFieldOrder))</td>
</tr>
<tr>
<td>s_nonbase_fields</td>
<td>Message_Block_Fieldₖ, ((1 ≤ i ≤ NumFields) ∧ (i ≠ BaseFieldOrder))</td>
</tr>
<tr>
<td>s_nonCurSize_fields</td>
<td>Message_Block_Fieldₖ, ((1 ≤ i ≤ NumFields) ∧ (i ≠ CurSizeFieldOrder))</td>
</tr>
<tr>
<td>s_nonMaxSize_fields</td>
<td>Message_Block_Fieldₖ, ((1 ≤ i ≤ NumFields) ∧ (i ≠ MaxSizeFieldOrder))</td>
</tr>
<tr>
<td>s_nonRdPtr_fields</td>
<td>Message_Block_Fieldₖ, ((1 ≤ i ≤ NumFields) ∧ (i ≠ RdPtrFieldOrder))</td>
</tr>
<tr>
<td>s_nonWrPtr_fields</td>
<td>Message_Block_Fieldₖ, ((1 ≤ i ≤ NumFields) ∧ (i ≠ WrPtrFieldOrder))</td>
</tr>
<tr>
<td>s_nonType_fields</td>
<td>Message_Block_Fieldₖ, ((1 ≤ i ≤ NumFields) ∧ (i ≠ TypeFieldOrder))</td>
</tr>
<tr>
<td>s_nonPriority_fields</td>
<td>Message_Block_Fieldₖ, ((1 ≤ i ≤ NumFields) ∧ (i ≠ PriorityFieldOrder))</td>
</tr>
<tr>
<td>s_nonCont_fields</td>
<td>Message_Block_Fieldₖ, ((1 ≤ i ≤ NumFields) ∧ (i ≠ ContFieldOrder))</td>
</tr>
<tr>
<td>s_nonNext_fields</td>
<td>Message_Block_Fieldₖ, ((1 ≤ i ≤ NumFields) ∧ (i ≠ NextFieldOrder))</td>
</tr>
<tr>
<td>s_nonPrev_fields</td>
<td>Message_Block_Fieldₖ, ((1 ≤ i ≤ NumFields) ∧ (i ≠ PrevFieldOrder))</td>
</tr>
<tr>
<td>s_nonAllocator_fields</td>
<td>Message_Block_Fieldₖ, ((1 ≤ i ≤ NumFields) ∧ (i ≠ AllocatorFieldOrder))</td>
</tr>
</tbody>
</table>
Appendix D

ACRONYMS, VARIABLES, SYMBOLS, CONSTANTS, AND AUXILIARY FUNCTIONS

D1.1. Acronyms

API: Application Programming Interface
CGI: Component Guide Document
CISD: Component Interface Specifications Document
CIDSD: Component Internal Design Specifications Document.
CO: Component–Oriented
Fr: Framework.
IPC: Inter–Process Communication
OO: Object–Oriented
OS: Operating Systems
PA: Path A.
Pb: Path B.
SP: Shared–Pool.
TRIO: Telecommunication Research Institute of Ontario
TTS: Table Tool System

D1.2. Variables

CANONICAL_REPRESENTATION_OfComponent_i: Represents the canonical representation for the i" component.
CANONICAL_REPRESENTATION_OfInternalClass_{i,j}: Represents the canonical representation for the {i,j}th internal class of the {i,j}th component.

CGD: Represents the Component Guide Document as described in section 6.2.

CGD_DescriptiveOverview: Represents the first chapter of the proposed CGD document. This chapter should describe the various aspects of the CO–Toolkit under consideration.

CGD_PictorialOverview: Represents the second chapter of the proposed CGD document. This chapter should provide a set of pictorial views that describe several things such as the file structure used for the considered CO–Toolkit, the various relations across the frameworks and among their components (i.e. such as inheritance and association relations), and etc.

CGD_IndexedQuickReferenceTables: Represents the third chapter of the proposed CGD document. This chapter should provide a set of index table that can help users as well as maintainers to find out what service(s) is (are) provided by each component in the CO–Toolkit under consideration.

CIDSD: Represents the Component Internal Design Specifications Document as described in section 6.4.

CIDSD_Chapter_Framework_{i}: Represents the CIDSD chapter that provides the internal design specifications for the {i}th framework.

CIDSD_Chapter_SharedPool_{k}: Represents the CIDSD chapter that provides the internal design specifications for the {k}th shared pool.

CIDSD_Section_Component_{i,j}: Represents the section dedicated for the internal design specifications of the {i,j}th component as a part of the CIDSD chapter for the {i}th framework.

CIDSD_Section_SharedClass_{k,j}: Represents the section dedicated for the internal design specifications of the {k,j}th shared class as a part of the CIDSD chapter for the {k}th shared–pool.

CIDSD_Section_OtherSharedEntity_{k,j}: Represents the section dedicated for the internal design of the {k,j}th shared non–class entities as a part of the CIDSD chapter for the {k}th shared–pool.

CIDSD_SubSection_Main_{i,j}: Represents the sub–section that provides the internal design specifications of the main part of the {i,j}th component.

CIDSD_SubSection_InternalClass_{i,j}: Represents the sub–section that provides the internal design specifications for the {i}th internal class in the {i,j}th component.

CIDSD_SubSection_SharedClass_Public_{k,j}: Represents the sub–section that provides the internal design specifications of the public section of the {k,j}th shared class.

CIDSD_SubSection_SharedClass_Protected_{k,j}: Represents the sub–section that provides the internal design specifications of the protected section of the {k,j}th shared class.
CIDSD_SubSection_SharedClass_Private_i;j: as explained in (6–21), this element represents the internal design specifications of the private section of the $kj^{th}$ shared class.

CIDSD_SubSection_Main_DISPLAY_i;j: Represents the $d^{th}$ DISPLAY of those used to specify the internal design of the Main part of the $ij^{th}$ component.

CIDSD_SubSection_InternalClass_Public_i;j: Represents the sub-section that provides the internal design specifications of the public section of the $i^{th}$ internal class of the $ij^{th}$ component.

CIDSD_SubSection_InternalClass_Protected_i;j: Represents the sub-section that provides the internal design specifications of the protected section of the $i^{th}$ internal class of the $ij^{th}$ component.

CIDSD_SubSection_InternalClass_Private_i;j: Represents the sub-section that provides the internal design specifications of the private section of the $i^{th}$ internal class of the $ij^{th}$ component.

CISD: Represents the Component Interface Specifications Document as described in section 6.3.

CISD_Chapter_Framework_i;j: Represents the CISD chapter that provides the interface specifications for the $i^{th}$ framework.

CISD_Section_Component_i;j: Represents the section dedicated for the interface specifications of $ij^{th}$ component as a part of the CISD chapter for the $i^{th}$ framework.

CISD_SubSection_CHARACTERISTICS_i;j: Represents the general characteristics of the $ij^{th}$ component.

CISD_SubSection_SYNTAX_i;j: Represents the signatures of the access programs of the $ij^{th}$ component.

CISD_SubSection_CANONICAL_REPRESENTATION_i;j: Represents the canonical representation for the $ij^{th}$ component.

CISD_SubSection_EQUIVALENCES_i;j: Represents the equivalences for the $ij^{th}$ component.

CISD_SubSection_VALUES_i;j: Represents the values returned by the access programs of the $ij^{th}$ component.

CISD_SubSection_LEXICON_i;j: Represents the lexicon for the interface specifications of the $ij^{th}$ component.

C_Files_i;j: Represents the set of files that are used to implement $j^{th}$ component in the $i^{th}$ framework of the CO_toolkit that is under consideration.

COToolkit: Represents any Component–Oriented Toolkit that can provide reusable components to be used to build new applications.

CO_Files: Represents the overall set of files that are used to implement the CO_toolkit that is under consideration.
**Component**_i,j_: Represents the _j_\textsuperscript{th} component in the _i_\textsuperscript{th} framework.

**EQUIVALENCES_OfComponent**_i,j_: Represents the set of equivalent states to the canonical one. An object of the _i_\textsuperscript{th} component goes into certain one of these equivalent states whenever a certain access program is invoked on that object.

**Framework**_i_: Represents the _i_\textsuperscript{th} framework.

**Fr_Files**_i_: Represents the set of files that are used to implement _i_\textsuperscript{th} framework of the CO_toolkit that is under consideration.

**Implementation**_i,j_: Represents the implementation of the _i,j_\textsuperscript{th} component.

**InheritedInterface**_i,j_: Represents the inherited interface of the _i,j_\textsuperscript{th} component. This refers to the interface of the methods that are inherited by that component.

**InheritedPublic_ACCESS_ProgramsByComponent**_i,j_: Represents the set of access programs (methods) that are publicly inherited by the _i,j_\textsuperscript{th} component from other components.

**intermediateFile**_i,j_: Represents the .h file of the _j_\textsuperscript{th} component in the _i_\textsuperscript{th} framework of the CO_toolkit that is under consideration.

**Interface**_i,j_: Represents the interface of the _i,j_\textsuperscript{th} component.

**InternalClass**_i,j_: Represents the implementation of the _i_\textsuperscript{th} internal class in the _i,j_\textsuperscript{th} component.

**InternalClassesSet**_i,j_: Represents the set of the internal classes in the _i,j_\textsuperscript{th} component.

**InternalClass_PublicMethod_DISPLAY**_i,j,m_: Represents the _m_\textsuperscript{th} Parnas DISPLAY of those required to specify the internal design of the _m_\textsuperscript{th} public method in the _i_\textsuperscript{th} internal class of the _i,j_\textsuperscript{th} component.

**InternalClass_ProtectedMethod_DISPLAY**_i,j,m_: Represents the _m_\textsuperscript{th} Parnas DISPLAY of those required to specify the internal design of the _m_\textsuperscript{th} protected method in the _i_\textsuperscript{th} internal class of the _i,j_\textsuperscript{th} component.

**InternalClass_PrivateMethod_DISPLAY**_i,j,m_: Represents the _m_\textsuperscript{th} Parnas DISPLAY of those required to specify the internal design of the _m_\textsuperscript{th} private method in the _i_\textsuperscript{th} internal class of the _i,j_\textsuperscript{th} component.

**InternalClass_DataStructureSpecifications**_i,j_: Represents the data structure implemented in the _i_\textsuperscript{th} class of the _i,j_\textsuperscript{th} component.

**internalFile**_i,j_: Represents the .i file of the _j_\textsuperscript{th} component in the _i_\textsuperscript{th} framework of the CO_toolkit that is under consideration.

**LocalInterface**_i,j_: Represents the local interface of the _i,j_\textsuperscript{th} component. This refers to the interface of the local non–inherited methods that belong to that component.

**LocalPublicMethodSignature**_i,j,k_: Represents the signature (interface description) of the _k_\textsuperscript{th} local method in the _i_\textsuperscript{th} class that belongs to the _i,j_\textsuperscript{th} component.
MainPart\_ij: Represents the implementation of the main part of the \( ij \)th component. The main part of a component contains the component's execution entry point (i.e. main function) and may contain other functions that do not belong to the internal classes of that component.

OtherSharedEntitiesSet\_k: Represents the set of other shared entities in the \( k \)th shared–pool.

OtherSharedEntity\_k\_ij: Represents the \( f \)th other shared entity in the \( k \)th shared–pool.

PrivateMethod\_ij\_l: Represents the implementation of the \( f \)th private method of the \( l \)th internal class in the \( ij \)th component.

PrivateObject\_ij\_k: Represents the \( k \)th private object of the \( l \)th internal class in the \( ij \)th component.

PrivateSection\_ij\_l: Represents the implementation of the private section of the \( l \)th internal class of the \( ij \)th component.

ProtectedMethod\_ij\_l: Represents the implementation of the \( f \)th protected method of the \( l \)th internal class in the \( ij \)th component.

ProtectedObject\_ij\_k: Represents the \( k \)th protected object of the \( l \)th internal class of the \( ij \)th component.

ProtectedSection\_ij\_l: Represents the implementation of the protected section of the \( l \)th internal class in the \( ij \)th component.

PublicConstantParameters\_ij: Represents the set of public constant that appear in the interface specifications of the \( ij \)th component.

PublicMethod\_ij\_l: Represents the implementation of the \( k \)th public method of the \( l \)th internal class in the \( ij \)th component.

PubliclyInheritedMethodSignature\_ij\_f: Represents the signature (interface description) of the \( f \)th method that is inherited by the \( l \)th class of the \( ij \)th component.

PublicPrimitiveSystemDataTypes\_ij: Represents the set of primitive data types (i.e. int, boolean etc.) that appear in the interface specifications of the \( ij \)th component.

PublicSection\_ij\_l: Represents the implementation of the public section of the \( l \)th internal class in the \( ij \)th component.

PublicSpecificationTypes\_ij: Represents the set of types that appear in the interface specifications of the \( ij \)th component.

PublicSpecificationVariables\_ij: Represents the set of variables that appear in the interface specifications of the \( ij \)th component.

Public\_ACCESS\_ProgramsInComponent\_ij: Represents the set of public access programs (methods) that are contained in the \( ij \)th component.
\[ P_{\text{INs}_{ij}} \]: Represents the sequence of inputs to the \( k^{th} \) public local method of the \( l^{th} \) internal class in the \( ij^{th} \) component.

\[ P_{\text{OUTs}_{ij}} \]: Represents the sequence of outputs from the \( k^{th} \) public local method in the \( l^{th} \) internal class of the \( ij^{th} \) component.

\[ P_{\text{ReturnedValue}_{ij}} \]: Represents the returned value by the \( k^{th} \) public local method of the \( l^{th} \) internal class in the \( ij^{th} \) component.

\[ P_{\text{INs}_{il}} \]: Represents the sequence of inputs to the \( f^{th} \) publicly inherited method of the \( l^{th} \) internal class in the \( ij^{th} \) component.

\[ P_{\text{OUTs}_{il}} \]: Represents the sequence of outputs from the \( f^{th} \) publicly inherited method of the \( l^{th} \) internal class in the \( ij^{th} \) component.

\[ P_{\text{ReturnedValue}_{il}} \]: Represents the returned value by the \( f^{th} \) publicly inherited method of the \( l^{th} \) internal class in the \( ij^{th} \) component.

\textbf{ReferencesTo\_CANONICAL\_REPRESENTATIONS\_OFInheritedComponents}_{ij} \]: Represents a set of references to the canonical representations of the other components that are inherited from by the \( ij^{th} \) component.

\textbf{ReferencesTo\_EQUIVALENCES\_OFInheritedComponents}_{ij} \]: Represents a set of references to the equivalences of the other components that are inherited from by the \( ij^{th} \) component.

\textbf{ReferencesTo\_VALUES\_OFInheritedComponents}_{ij} \]: Represents a set of references to the values returned by the public access programs (methods) of the other components that are inherited from by the \( ij^{th} \) component.

\textbf{ReferencesTo\_DISPLAYs\_OFPublicMethodsInheritedByInternalClass}_{ij} \]: Represents a set of references to the set of Parnas DISPLAYs that provide the internal specifications for those internal class methods that are inherited by the \( l^{th} \) internal class of the \( ij^{th} \) component.

\[ R_{\text{INs}_{ij}} \]: Represents the sequence of inputs to the \( f^{th} \) protected method of the \( l^{th} \) internal class in the \( ij^{th} \) component.

\[ R_{\text{OUTs}_{ij}} \]: Represents the sequence of outputs from the \( f^{th} \) protected method of the \( l^{th} \) internal class in the \( ij^{th} \) component.

\[ R_{\text{ReturnedValue}_{ij}} \]: Represents the returned value by the \( f^{th} \) protected method of the \( l^{th} \) internal class in the \( ij^{th} \) component.

\textbf{Sh\_Files}_{ij} \]: Represents the set of files that are used to implement \( k^{th} \) shared-pool of the CO_toolkit that is under consideration.
Sh_File<\textsubscript{k}>: Represents the \textit{l}\textsuperscript{th} file in the \textit{k}\textsuperscript{th} shared–pool.

topFile<\textsubscript{i}>: Represents the .cpp file of the \textit{j}\textsuperscript{th} component in the \textit{i}\textsuperscript{th} framework of the CO\_toolkit that is under consideration.

SharedPool<\textsubscript{k}>: Represents the \textit{k}\textsuperscript{th} shared–pool.

SharedClass<\textsubscript{k}>: Represents the \textit{j}\textsuperscript{th} class in the \textit{k}\textsuperscript{th} shared–pool.

SOF<\textsubscript{X}(A, B)>: Represents the implementation of method \textit{X}.

TypeSpecified<\textsubscript{i}>: Represents the specification type of the \textit{i}\textsuperscript{th} component.

VALUES\_Of\Component<\textsubscript{i}>: Represents the set of values that are returned by the public access programs (methods) of the \textit{i}\textsuperscript{th} component.

\textit{V}\_\textit{INs}<\textsubscript{i}j>: Represents the sequence of inputs to the \textit{j}\textsuperscript{th} private method of the \textit{l}\textsuperscript{th} internal class in the \textit{i}\textsuperscript{th} component.

\textit{V}\_\textit{OUTs}<\textsubscript{i}j>: Represents the sequence of outputs from the \textit{j}\textsuperscript{th} private method of the \textit{l}\textsuperscript{th} internal class in the \textit{i}\textsuperscript{th} component.

\textit{V}\_\textit{ReturnedValue}<\textsubscript{i}j>: Represents the returned value by the \textit{j}\textsuperscript{th} private method of the \textit{l}\textsuperscript{th} internal class in the \textit{i}\textsuperscript{th} component.

D1.3. Symbols And Constants

\textbf{F}: Number of Frameworks in the given CO\_toolkit.

\textbf{C}: Number of Components in the \textit{i}\textsuperscript{th} Framework.

\textbf{c}: Represents the number of local classes used to construct the \textit{i}\textsuperscript{th} component.

\textbf{c}: Represents the number of shared classes in the \textit{k}\textsuperscript{th} shared–pool.

\textbf{c}: Represents the number of other shared entities in the \textit{k}\textsuperscript{th} shared–pool.

\textbf{M}\_\textsuperscript{a}: Represents the number of local public methods in \textit{l}\textsuperscript{th} class of the \textit{i}\textsuperscript{th} component.

\textbf{M}\_\textsuperscript{int}: Represents the number of public methods inherited by the \textit{l}\textsuperscript{th} class of the \textit{i}\textsuperscript{th} component.

\textbf{M}\_\textsuperscript{p}: Represents the number of local protected methods in the \textit{l}\textsuperscript{th} class of the \textit{i}\textsuperscript{th} component.

\textbf{M}\_\textsuperscript{p}: Represents the number of local private methods in the \textit{l}\textsuperscript{th} class of the \textit{i}\textsuperscript{th} component.

\textbf{NumberOfDISPLAYsForMainPartOfComponent}: Represents the number of Parnas DISPLAYs that are required to specify the internal design of the main part of the \textit{i}\textsuperscript{th} component.
**NumberOfDISPLAYsForPublicMethod**$_{i,j}$: Represents the number of Parnas DISPLAYs that are required to specify the internal design of the $m$th public method in the $l$th internal class of the $i,j$th component.

**NumberOfDISPLAYsForProtectedMethod**$_{i,j}$: Represents the number of Parnas DISPLAYs that are required to specify the internal design of the $m$th protected method in the $l$th internal class of the $i,j$th component.

**NumberOfFilesInCO_Files**: Represents the total number of files used to implement the given CO-Toolkit.

$O^{*}_{i,j}$: Represents the number of objects in the protected section of the $l$th class of the $i,j$th component.

$O^{*}_{i,j}$: Represents the number of objects in the private section of the $l$th class of the $i,j$th component.

$pm^{\text{in}}_{i,j}$: Represents the number of inputs of the $k$th local public method in the $l$th class of the $i,j$th component.

$pm^{\text{out}}_{i,j}$: Represents the number of outputs of the $k$th local public method in the $l$th class of the $i,j$th component.

$pm^{\text{in}}_{i,j}$: Represents the number of inputs of the $f$th public method that is inherited by the $l$th class of the $i,j$th component.

$pm^{\text{out}}_{i,j}$: Represents the number of outputs of the $f$th public method that is inherited by the $l$th class of the $i,j$th component.

$rm^{\text{in}}_{i,j}$: Represents the number of inputs of the $f$th local protected method in the $l$th class of the $i,j$th component.

$rm^{\text{out}}_{i,j}$: Represents the number of outputs of the $f$th local protected method in the $l$th class of the $i,j$th component.

$S$: Number of Shared-Pools.

$s_{k}$: Number of Files in the $k$th Shared-Pool.

$vm^{\text{in}}_{i,j}$: Represents the number of inputs of the $f$th local private method in the $l$th class of the $i,j$th component.

$vm^{\text{out}}_{i,j}$: Represents the number of outputs of the $f$th local private method in the $l$th class of the $i,j$th component.
D1.4. Auxiliary Functions

\[ \text{Associates}(X, Y) = (\text{UniDirectionalAssociation}(X, Y) \lor \text{BiDirectionalAssociation}(X, Y)) \] : An auxiliary function that returns TRUE if classes X and Y associate with each other.

\[ \text{BiDirectionalAssociation}(X, Y) = \begin{cases} \text{One_to_One_BiDirectionalAssociation}(X, Y) \lor \\ \text{One_to_Man_MANY_BiDirectionalAssociation}(X, Y) \lor \\ \text{Many_to_Man_MANY_BiDirectionalAssociation}(X, Y) \end{cases} \] : An auxiliary function that returns TRUE if X and Y have a form of bi-directional association with each other.

\[ \text{CARD}(\text{FLAG}, X) \] : This is an auxiliary function that returns the number of elements in the set X. For convenience and as shown in the next table, the FLAG is used to indicate the type of elements in the set X.

\[ \text{Category}(X) \] : An auxiliary function that returns the category of the file X. It could be 'top', 'intermediate', 'internal', or 'shared'.

\[ \text{FileName}(X) \] : An auxiliary function that returns the filename of file X.

\[ \text{Fr_Name}(\text{Fr_Files}) \] : An auxiliary function that returns the name of the framework that is implemented using the Fr_Files, file subset.

\[ \text{Frs_PermittedToAccessSP}(k) \] : An auxiliary function that returns the set of frameworks that are permitted to access the \( k \)th shared-pool.

\[ \text{Includes}(X, Y) \] : An auxiliary Boolean function that returns TRUE or FALSE. It returns TRUE when file X is "#include"ing file Y, otherwise it returns FALSE.

\[ \text{InheritanceCategory}(\text{Inherits}(X, Y)) \] : An auxiliary function that returns the inheritance category when component X inherits from component Y. The returned value is either "Public", "Protected", or "Private".

\[ \text{Inherits}(X, Y) \] : An auxiliary boolean function that returns TRUE if component X does inherit from component Y. Otherwise, it returns FALSE.

\[ \text{LENGTH}(\text{FLAG}, Y) \] : This function returns the number of elements in the sequence Y. For convenience and as shown in the next table, the FLAG is used to indicate the type of elements in the sequence Y.

\[ \text{Links}(X, Y) \] : An auxiliary Boolean function that returns TRUE or FALSE. It returns TRUE when file X contains parts of file Y, otherwise, it returns FALSE.

\[ \text{Many_to_Man_MANY_BiDirectionalAssociation}(X, Y) \] : An auxiliary function that returns TRUE if Classes X and Y each contains multiple objects of the other.

\[ \text{One_to_One_BiDirectionalAssociation}(X, Y) \] : An auxiliary function that returns TRUE if Classes X and Y each contains an object of the other.
**APPENDIX D: ACRONYMS, VARIABLES, SYMBOLS, CONSTANTS, AND AUXILIARY FUNCTIONS**

*One_to_One_UniDirectionalAssociation*(X, Y) = An auxiliary function that returns TRUE if Class X contains an object of Class Y while Y does not contain any object of X.

*One_to_Many_BiDirectionalAssociation*(X, Y) = An auxiliary function that returns TRUE if Class X contains a set of objects of Class Y while Y contains only one object of class X.

*One_to_Many_UniDirectionalAssociation*(X, Y) = An auxiliary function that returns TRUE if Class X contains a set of objects of Class Y while Y does not contain any object of X.

*ProvidesInterfaceSpecifications*(X, Y) = This is an auxiliary predicate that returns TRUE if the interface specifications of the Sub−System Y are provided in the portion X of the CISD document.

*ProvidesDesignSpecifications*(X, Y) = This is an auxiliary predicate that returns TRUE if the design and implementation specifications of the Sub−System Y are provided in the portion X of the CIDSD document.

*SEQUENCE*[f₁, f₂, ..., fₙ]*: is used to denote a sequence of file names as per Hoffman’s notation [25].

*Type*(X): An auxiliary function that returns the data type of the argument X.

*UniDirectionalAssociation*(X, Y) = \(( \text{*One_to_One_UniDirectionalAssociation*(X, Y) \lor \text{*One_to_Many_UniDirectionalAssociation*(X, Y)})\): An auxiliary function that returns TRUE if X and Y have a form of uni−directional association with each other.

*UsedToImplement*(X, Y) = This is an auxiliary predicate that returns TRUE if the Source Code of the Sub−System Y is provided in the Set of Files X.
REFERENCES


[8] Bruno Lague, Bell Canada Presentation on the DATRIX ToolKit, CRL–B102, McMaster University, Dec. 5 1996. A user manual for DATRIX is available as well.


REFERENCES

[14] Ciupke, Oliver, “Automatic Detection of Design Problems in Object–Oriented Reengineering”, FZI Forschungszentrum Informatik, 1999. This can be obtained from the author at e-mail: ciupke@fzi.de


REFERENCES


405
REFERENCES


[53] SERG: The Software Engineering Research Group, Communication Research Laboratory, Department of Electrical and Computer Engineering, McMaster University. Under the supervision of Professor David L. Parnas.

[54] Shen, H., and Parnas D.L, a project on formal specifications for OO–programs. A project for Bell Canada under the supervision of Dr. Parnas.


[57] Wong, K., “Rigi User’s Manual Version 5.4.3”, Nov. 1, 1996. This manual is provided by the Rigi software system.

Other Related References


REFERENCES


