A MODEL FOR THE FDA GENERAL PRINCIPLES OF SOFTWARE VALIDATION

By

MARWAN M. ABDEEN

B.Sc. (Princess Sumaya University for Technology)

A Dissertation

Submitted to the School of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree of

Master of Applied Sciences

in

Software Engineering

McMaster University

MASTER OF APPLIED SCIENCES (2007)
(Software Engineering)

McMaster University
Hamilton, Ontario

TITLE: A Model for the FDA General Principles of Software Validation
AUTHOR: Marwan M. Abdeen, B.Sc. (Princess Sumaya University for Technology)
SUPERVISORS: Dr. Wolfram Kahl and Dr. Tom Maibaum
NUMBER OF PAGES: x, 134

Abstract

A Model for the FDA General Principles of Software Validation

Several institutions and agencies around the world, from both the private and public sectors, have adopted the practice of software validation and certification to ensure higher levels of confidence in software. The US Food and Drug Administration (FDA) is one of these organisations. The FDA has published several guidance documents concerning the validation of medical device software or the validation of software used to design, develop, or produce medical devices. In its guidance documents for both the medical software industry and FDA staff, the FDA recommends certain activities to be undertaken and certain deliverables to be prepared. The FDA believes that undertaking these activities as well as preparing these deliverables will achieve higher confidence in the software quality and, accordingly, better validation.

In this thesis, we question the clarity and practicability of the FDA approach for medical software validation. We examine the FDA approach by evaluating the FDA guidance documents. We base our analysis on distinguishing between product and process aspects. By bearing in mind these two concepts, we believe that the FDA validation approach leads to confusion in the minds of both software producers and evaluators and it may well lead to lower quality software than desired being certified, while the FDA considers its approach as the least burdensome. As a demonstration of our claim, we analyse "Quality Planning" as the first activity in the FDA approach. We discuss the corresponding effort for this activity in another product-oriented evaluation approach, the Common Criteria (CC) for Information Technology Security Evaluation. As its main objective, the comparison aims to highlight both the inconsistency and the vagueness of the FDA evaluation function.

The FDA guidance documents use natural language in specifying their recommendations, because of the wide audience to which they are addressed and the consequent need for simplicity. However, the specification is not sufficiently explicit and precise to be able to impose a contract (obligation) between the two parties (software producers and evaluators). Having illustrated the drawbacks of the FDA approach, with comparison to the CC, as they appear in the "Quality Planning" activity, we use the Product/process (P/p) method to model this activity and all subsequent ones. Despite the application of the P/p method herein to medical device certification, a critical area of application, the P/p method takes a general systems approach. It is appropriate to a variety of areas and has proven its applicability in many fields. Thus, reconsideration of these issues by the FDA will lead to a more objective evaluation function and higher confidence in the medical software, which represents the most important concern among the evaluation community and the general public.

Contents

List of Figures

Chapter 1

Introduction

The US Food & Drug Administration (FDA) has expressed its recommendations concerning software validation in the form of guidance documents. To achieve simplicity, the FDA has used natural language in specifying these recommendations. However, the specification is not sufficiently clear and precise. Hence, we introduce in this thesis a simplified representation for the FDA validation approach using Product/process (P/p) modelling.

This chapter starts with describing a problem as it appears in medical software validation, a critical area of application. The statement of purpose section briefly describes this problem and explains the reasons for undertaking this work. It also provides a high level description to the work itself. After that, some essential background which helps in understanding this work is provided. This background helps the reader understand the US Food & Drug Administration (FDA) validation approach which represents the main focus of this thesis. It also describes the validation activities which are recommended by the FDA, to be conducted by software producers during the development of the medical software. After that, section 1.3 introduces the Common Criteria (CC) for Information Technology Security Evaluation as an international standard for specifying and evaluating IT security requirements. Section 1.4 introduces the Product/process (P/p) modelling methodology as an effective method in representing any engineering problem in a simplified, yet explicit way. Finally, section 1.5 describes the information flow of this thesis in terms of chapters and their rationales as well.

1.1 Statement of purpose

Certification has been a concern amongst the software engineering community for the past few decades and is becoming a major concern today. Several organisations, in charge of certification, have published guidance documents to describe this crucial activity. Indeed, these organisations, through their guidance documents, aim to establish a common understanding

between software producers and certifiers (evaluators). The US Food & Drug Administration (FDA) is one of these organisations. The FDA has published several guidance documents concerning the validation of medical software. However, these recommendations are not specified in an explicit and precise manner. In more detail, the FDA validation approach as described in the FDA guidance document [FDA 2002]:

- does not describe the objects that are subject to assessment,
- does not specify the measurable attributes that characterize these objects, and
- does not describe the criteria on which the FDA staff will base their decision, in order to approve or reject the medical software.

In this thesis, we illustrate this problem as it appears in the FDA guidance documents for validating medical device software. We examine the FDA approach by evaluating the guidance documents published by the FDA. We base our analysis on distinguishing between product and process aspects. By bearing in mind these two concepts (product and process), we believe that the FDA validation approach leads to confusion in the minds of both software producers and evaluators and it may well lead to lower quality software than desired being certified, at the time the FDA considers its approach as the least burdensome. As a demonstration of our claim, we analyse "Quality Planning" as the first activity in the FDA approach. The FDA shortcomings become more apparent when the FDA validation approach is compared to another certification methodology, the Common Criteria (CC) for Information Technology (IT) Security Evaluation. The comparison is undertaken on the quality planning activity and does not go over all other subsequent activities defined in the FDA approach. This is due to the fact that the deficiencies underlined in the quality planning activity are repeated in all subsequent activities. As its main objective, the comparison aims to highlight both the inconsistency and the vagueness of the FDA evaluation function.

Having highlighted the drawbacks of the FDA approach, we use the Product/proces (P/p) methodology to model (specify) the FDA validation approach. We restrict the P/p modelling to the FDA approach and we do not carry it on the CC approach. This is because the CC has its own systematic approach for specification and evaluation. Furthermore, the CC has its own explicit evaluation approach which appears in the Common Evaluation Methodology (CEM), a companion document to the CC guidance documents, which is developed to provide an agreed-on methodology for evaluating products which apply the CC.

Since the outstanding problems described in the quality planning activity also appear in other activities defined in the FDA approach, we use the Product/process (P/p) method to model this activity and all subsequent ones. The P/p method is used to provide a simplified, yet explicit specification for the FDA validation approach. Despite its application herein to medical device certification, the P/p model takes a general systems approach. It is appropriate to a variety of areas and has proven its applicability in many fields.

In brief, we describe the shortcomings in the current FDA validation approach as they appear in the FDA guidance document. Moreover, we provide a model (simplified representation) for the FDA validation approach in the sense that the validation activities (processes)

are described, as per the P/p method, in terms of input and output products. These products are then modelled in terms of attributes for which measurement scale types are defined. Furthermore, the relationships among products' attributes are also specified. Such a model (specification) provides an explicit description to the validation activities, to be undertaken during software development, and their deliverables as well. Both software producers and FDA staff can benefit from this model. The model describes products' attributes which software producers would be concerned to measure. The model also describes these products in a concrete structure which would help the FDA in establishing the evaluation criteria for attributes of these products. In essence, modelling the activities of the FDA validation approach does not aim to impose a definitive proposal on the FDA. Instead, it aims to illustrate the existing problems and the way in which these matters could be handled. Reconsideration of these issues by the FDA will lead to a more objective evaluation function and higher confidence in the medical software, which represents the most important concern among the evaluation community and the general public.

1.2 Food & Drug Administration (FDA)

The Food and Drug Administration (FDA) is a public agency that is concerned with the validation of medical device software in the United States of America. In response to the questions about the FDA validation requirements, the FDA has expressed its current thinking about medical software validation through guidance documents. These documents target both the medical software industry and FDA staff. In this context, the following FDA guidance documents are referenced in this thesis:

- *General Principles of Software Validation* [FDA 2002]

  This guidance describes the validation activities which are recommended to be undertaken during the development of the medical software. The validation approach described in this guidance is the main concern of this thesis.

Other documents are:

- *Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices* [FDA 2005]

  This guidance describes the documents that are recommended to be delivered to the FDA as part of the premarket submission for medical devices. Some of the documents described in this guidance are related to the activities illustrated in the previous guidance.

- *Guidance for Industry, FDA Reviewers and Compliance on Off-The-Shelf Software Use in Medical Devices* [FDA 1999]

This guidance describes the documentation that should be submitted to the FDA concerning Off-The-Shelf (OTS) software that is used by the medical software. The submitted documents significantly depend on the level of concern which the OTS software presents.

- *Guidance on General Principles of Process Validation* [FDA 1993]

  This guidance describes the principles of process validation, essentially when validating automated processes and equipments to ensure that they consistently produce products (pharmaceuticals and medical devices) in accord with their predetermined specifications.

In this context, the FDA defines validation as 'confirmation by examination and provision of objective evidence that software specifications conform to user needs and intended uses, and that the particular requirements implemented through software can be consistently fulfilled' [FDA 2002]. According to the FDA, validation is an important activity which has to be undertaken throughout the software development lifecycle. In other words, it occurs at the beginning, end and even during stages of software development. Thus, the FDA guidance documents recommend validation to start early while the software is being developed. In this sense, the FDA guidance document [FDA 2002] considers other activities like planning, verification, testing, traceability and configuration management as important activities which all together participate in reaching a conclusion that the software is validated.

In essence, the FDA validation approach is a generic approach. It appears in the form of recommendations to apply some software engineering practices. These practices are considered to be working hand in hand to support the validation process. The reason behind the FDA taking such a generic approach is due to the 'variety of medical devices, processes, and manufacturing facilities' [FDA 2002]. In other words, the nature of validation is significantly dependant on the medical device itself. Examples of such validation determinant factors are [FDA 2002]:

- availability of production environment for validating the software,
- ability to simulate the production environment,
- availability of supportive devices,
- level of risk,
- any prerequisite regulations/approvals to validation, etc.

The recommendations in the FDA guidance documents aim to achieve higher levels of confidence in the software and, accordingly, assist the software to be certified. These recommendations apply to software [FDA 2002]:

- used as a component, part, or accessory of a medical device,
- that is itself a medical device (e.g., blood establishment software),

- used in the production of a device (e.g., programmable logic controllers in manufacturing equipment),
- used in implementation of the device manufacturer's quality system (e.g., software that records and maintains the device history record).

In its guidance documents, the FDA recommends certain validation activities to be undertaken and certain deliverables to be prepared. These activities and deliverables are subject to evaluation afterwards by the FDA staff. For instance, validating the Software Requirements Specification (SRS) aims to ensure that there are no ambiguous, incomplete, unverifiable and technically infeasible requirements. Such a validation seeks to ensure that these requirements sufficiently describe the users' needs, and that they are sufficient to achieve the users' objectives. In the same manner, testing is another key activity which is thoroughly described in the guidance.

On the other hand, the guidance points out some issues that are interrelated as a result of the nature of software. Examples of such issues are frequent changes and their negative consequence, and personnel turnover in the sense that software maintainers might have not been involved in the original development. Moreover, the FDA guidance stresses on the importance of having well-defined procedures to handle any software change introduced. Validation in this context addresses the newly added software as well as already existing software. In other words and in addition to validating the newly added pieces (components) of code, the effect of these new pieces on the existing ones has also to be checked. Such a check ensures that the new components have no negative impact on the existing ones. Furthermore, the guidance highlights the importance of having independence in the review process in the sense that personnel who participate in validating the software are not the ones who developed it. These are mainly the issues which the FDA guidance documents address with regard to software validation.

In terms of software development, the FDA approach does not favour any specific software development model. In other words, it leaves the choice of the approach, to be used in developing the software, to software producers themselves. This supports the fact that some organisations have their own policies, standards and development approaches that should be followed. This also supports the fact that some approaches may well suit certain types of projects or software. Therefore, the FDA leaves the choice of the software development model to software producers, as long as it sufficiently describes the lifecycle of the software. However, the FDA explicitly requires that validation occurs throughout all stages of the software lifecycle.

1.3 Common Criteria (CC) for Information Technology Security Evaluation

The Common Criteria (CC) for Information Technology (IT) Security Evaluation is an international standard for specifying and evaluating IT security requirements. This standard was developed as a result of a cooperation between six national security and standards organisations in the Netherlands, Germany, France, United Kingdom, Canada and the United States of America. This cooperation aims to define an international standard in order to replace the existing security evaluation criteria in those countries. The main reason for considering the CC is the systematic and consistent approach which the CC follows in specifying security requirements and evaluating their implementation. In this sense, we compare the FDA approach with the CC approach in chapter two. Thus, we provide the necessary background for this certification methodology in this section.

In the CC, IT security requirements can be classified into Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs). SFRs are mainly concerned with describing the functionalities to be implemented by the final product. Whereas SARs are concerned with describing the properties which the final product should possess.

As per the CC terminology, the final product which has to be developed, and after that evaluated, is called Target-Of-Evaluation (TOE). The TOE is defined in [CC 2006a] as 'an IT product or system and its associated administrator and user guidance documentation that is the subject of an evaluation' [CC 2006a]. The requirements which describe the TOE are specified in the Security Target (ST). The Security target is a document which is similar to an SRS document, in which functional requirements are specified using [CC 2006b], and assurance requirements are specified using [CC 2006c]. These requirements are categorized in [CC 2006b] and [CC 2006c], according to the CC taxonomy, into classes, families and components. A class describes the security focus of its members. In other words, families of the same class share the same security concern, but each has a security objective which supports that concern. Components in the same family share the security objective of their family, but differ in the level of rigour in which the security objective is handled.

For instance, the security focus of communication class (Class CFO: Communication), a security functional class defined in [CC 2006b], is to 'assure the identity of a party participating in a data exchange' [CC 2006b]. This class has two families with two different objectives, yet they share the same security concern (communication). Non-repudiation of origin (FCO_NRO: Non-repudiation of origin) is the first family in this class which aims to ensure that the 'originator cannot successfully deny having sent the information' [CC 2006b]. Whereas, the other family (FCO_NRR: Non-repudiation of receipt) aims to ensure that the 'recipient cannot successfully deny receiving the information' [CC 2006b]. Components in the same family solve the security problem as described by their family, but with different level of rigour. In this sense, "non-repudiation of origin" has two components. The first component (FCO_NRO.1 Selective proof of origin) solves the "repudiation of origin" problem in the sense that it requires the "relied-on security enforcer (software or hardware or

even both)" to have entities 'with the capability to request evidence of the origin of information' [CC 2006b]. Whereas, the other component (FCO_NRO.2 Enforced proof of origin) also solves the "repudiation of origin" problem, but it requires the "relied-on security enforcer (software or hardware or even both)" to *always* 'generate evidence of origin for transmitted information' [CC 2006b].

In this context, CC part 2 [CC 2006b] describes the following security functional classes with their families and components: security audit, communication, cryptographic support, user data protection, identification and authentication, security management, privacy, protection of the Toe Security Functionality (TSF), resource utilisation, and trusted path/channels.

On the other hand, CC part 3 [CC 2006c] describes the following security assurance classes with their families and components: Protection Profile (PP) evaluation, Security Target (ST) evaluation, development, guidance documents, life-cycle support, tests, vulnerability assessment, and composition. As previously described, components of the same family share the security objective of the family, but they differ in the level of rigour of the implementation of that objective. In other words, the level of rigour is essentially determined by the components which describe the level of confidence required for particular security issues.

According to the CC, the level of confidence required in the TOE is determined by the Evaluation Assurance Level (EAL). The CC defines seven evaluation assurance levels. These are [CC 2006c]:

- EAL1: functionally tested
- EAL2: structurally tested
- EAL3: methodically tested and checked
- EAL4: methodically designed, tested and reviewed
- EAL5: semiformally designed and tested
- EAL6: semiformally verified design and tested
- EAL7: formally verified design and tested

Each evaluation assurance level requires certain components of particular assurance families to be implemented. The correspondence between the evaluation assurance level and the components of assurance families in a class appears in the following table as given in [CC 2006c]. The "assurance class" column lists the security assurance classes as defined in CC part three (CC part 3: Security Assurance Requirements [CC 2006c]). Each of these classes has security assurance families which share the security concern with other families in the same class. The assurance families of each class are listed under the "assurance family" column along the assurance class row. For instance, the "Development" assurance class (ADV) has ADV_ARC, ADV_FSP, ADV_IMP, ADV_INT, ADV_SPM and ADV_TDS as its assurance families. It is the CC convention for any family to start with the class symbol (ADV for

Assurance class	Assurance Family	Assurance Components by Evaluation Assurance Level						
		EAL1	EAL2	EAL3	EAL4	EAL5	EAL6	EAL7
Development	ADV_ARC		**1**	1	1	1	1	1
	ADV_FSP	**1**	**2**	**3**	**4**	**5**	5	**6**
	ADV_IMP				**1**	1	**2**	2
	ADV_INT					**2**	**3**	3
	ADV_SPM						**1**	1
	ADV_TDS		**1**	**2**	**3**	**4**	**5**	**6**
Guidance documents	AGD_OPE	**1**	1	1	1	1	1	1
	AGD_PRE	**1**	1	1	1	1	1	1
Life-cycle support	ALC_CMC	**1**	**2**	**3**	**4**	4	**5**	5
	ALC_CMS	**1**	**2**	**3**	**4**	**5**	5	5
	ALC_DEL		**1**	1	1	1	1	1
	ALC_DVS			**1**	1	1	**2**	2
	ALC_FLR							
	ALC_LCD			**1**	1	1	1	**2**
	ALC_TAT				**1**	**2**	**3**	3
Security Target evaluation	ASE_CCL	**1**	1	1	1	1	1	1
	ASE_ECD	**1**	1	1	1	1	1	1
	ASE_INT	**1**	1	1	1	1	1	1
	ASE_OBJ	**1**	**2**	2	2	2	2	2
	ASE_REQ	**1**	**2**	2	2	2	2	2
	ASE_SPD		**1**	1	1	1	1	1
	ASE_TSS	**1**	1	1	1	1	1	1
Tests	ATE_COV		**1**	**2**	2	2	**3**	3
	ATE_DPT			**1**	**2**	**3**	3	**4**
	ATE_FUN		**1**	1	1	1	**2**	2
	ATE_IND	**1**	**2**	2	2	2	2	**3**
Vulnerability assessment	AVA_VAN	**1**	**2**	2	**3**	**4**	**5**	5

Figure 1 1. Evaluation Assurance Level Summary [CC 2006c]

DeVelopment Class) followed by an underscore (_) and then a family symbol (ARC for security ARChitecture family, FSP for Functional SPecification family, IMP for IMPlementation representation family, INT for tsf INTernals family, SPM for Security Policy Modelling family and TDS for Toe DeSign family). The development class symbol (ADV) and all other assurances classes' symbols (AGD for Guidance Documents, ALC for Life Cycle, ASE for Security Target evaluation, ATE for TEsts and AVA for Vulnerability Assessment) start with the letter "A" in order to differentiate them from functional classes.

As described in the "evaluation assurance level summary" table, each family has a set of components that share the same security problem but differ in the level of rigour of the solution. For example, ADV_FSP family of the assurance class "development" has six components. As the table dictates, the first component (1) is necessary for evaluation assurance level one (EAL1). Whereas the second component (2) of the same family is necessary for evaluation assurance level two (EAL2). In the same manner, EAL5 and EAL6 both require the implementation of the fifth component (5) of this family, and so on. Finally, some cells in the table are left out without any component. That means that it is not required to implement any component from the given family in order to achieve that evaluation assurance level.

Security functional and assurance requirements are specified in [CC 2006b] and [CC 2006c], respectively. They are specified in a generic way which allows customization. To be more concrete, the CC defines some operations which enable changing these requirements. There operations are: assignment, selection, refinement and iteration. For instance, a functional requirement which is specified in [CC 2006b] using the assignment operation enables the requirements developer to assign some specific values which may differ from one organisation to another and thus restricts the requirement defined in [CC 2006b]. In a sense, the assignment operation enables requirements developers to adjust the requirements specified in CC part 2 [CC 2006b] and CC part 3 [CC 2006c] to fit their purpose.

Developing the security target starts with defining the level of confidence required in the product (software product in this case) as per the evaluation assurance levels. Having the level of confidence determined, the evaluation assurance level summary table imposes the specification and implementation of the security components in the ST and the TOE respectively. The requirements written in the security target, as taken from [CC 2006b] and [CC 2006c], can then be customized to reflect some restrictions, such as organisation-specific or product-specific issues. Developing the security target using only the security functional requirements of [CC 2006b] results in a CC part 2 conformant product. However, adding extra requirements to the security target, which are demonstrated to be needed by the ST developer, results in a CC part 2 extended product. The same concept applies to CC part 3 security assurance requirements.

The security assurance requirements are organized into action elements for the developer, action elements for the content and presentation of the submitted deliverable, and action elements for the evaluator. Each evaluator action element in [CC 2006c] corresponds to an action which is detailed into work units in the Common Evaluation Methodology [CC 2006d], a companion document to the CC documents which describes the way in which a product

specified using the CC requirements is evaluated. The work units describe the steps which are to be undertaken by the testing laboratory in evaluating the ST, TOE and all other intermediate products. If these products passed the evaluation of a testing laboratory authorized by the CC, they would be submitted to the national scheme in that country to be certified.

In conclusion, the CC supports specification and development of security requirements using [CC 2006b] and [CC 2006c]. Thus, such a specification (ST), its implementation (TOE) and all other intermediate products are subject to evaluation as per the CEM [CC 2006d].

1.4 Product/process (P/p) modelling

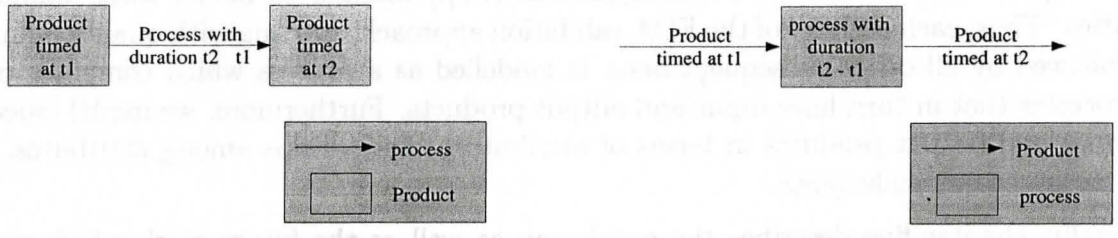
The "Product/process (P/p) methodology" has been introduced by Myers and Kaposi in their book "A first Systems Book" [Myers and Kaposi 2004]. This methodology takes a general systems approach to handle any problematic situation. In essence, it represents any engineering problem as a system (S) with a pair of elements (E) and relationships (R) among these elements (S = (E,R)). Such a representation for the problem (system) helps in establishing a common understanding between all participants about the problem and forms the basis for solution. The Product/process methodology makes a clear distinction between the world of realities and the world of models. An *entity* is defined as an object which exists in the real world. A *referent* is the entity of our concern. In this context, both the entity and the referent belong to the world of realities. Whereas, the system, which is defined as 'a representation of a referent as a set of interrelated elements' [Myers and Kaposi 2004], belongs to the world of models. Similarly, an attribute is defined as 'a representation of a characteristic of the referent'. An attribute can be measured in terms of values belonging to a measurement scale type. Thus, to model a referent in the real world is to provide a 'purposefully simplified representation of the referent, where all attributes can be given as measures' [Myers and Kaposi 2004].

From a construction perspective, the referent can be modelled as a whole or as structures (parts). When modelled as a whole, the referent is represented as a black box system. A black box system is autonomous. It stands on its own. Attributes that characterize the system along with the relationships among them are sufficient to specify (describe) the referent. To be more precise, a black box system has the attributes which characterize it as its elements (E), and the relationships among these attributes as the relationships (R) among the system's elements. Thus, the generic formal definition of a system S = (E,R) can be rewritten as S_B=(E_B,R_B), where E_B represents the set of all attributes that describe the referent and R_B represents the set of interrelationships over elements of E_B. An interrelationship can be thought of as a relationship which exists between the attributes of the same system as well as attributes of other systems. The finite set of relationships R_B has at least two mandatory relationships over attributes (elements of the E_B set). These are the co-attribute relationship (r_c), and the time-stamp relationship (r_t). In more detail, the co-attribute relationship (r_c) is the relationship which asserts that attributes (elements) in the set of all attributes belong

to the same referent. Whereas, the time-stamp relationship (r_t) is the relationship which specifies the time at which these attributes were measured and thus assigned values. Only attributes that help in providing a simplified representation of the referent in its problematic environment are our concern. Common attribute measurement scale types are: absolute, ratio, ordinal, interval, nominal and pieces (bits) of text. These measurement scales specify the types of values which will be given to attributes. In this context, measuring attributes, as per the defined measurement scales, can be accomplished directly or indirectly. Indirect measurements are derived from direct measurements.

On the other hand, a structural system cannot be described simply by its attributes. In other words, and because of the structure of the referent, attributes do not sufficiently describe the referent. This is due to the fact that the referent relies on its components. These components are represented as black box systems and are subject to modelling in their own sight. To be more precise, a structural system has the components which constitute it as its elements (E), and the rules, which specify the way in which these components are glued together to form the system as a whole, as the relationships (R) between the system elements. Thus, the generic formal definition of a system S = (E,R) can be rewritten as $S_S = (E_S,R_S)$, where E_S represents the set of all components, each as a part of the referent, and R_S represents the set of 'relationships among the components as elements of E_S' [Myers and Kaposi 2004]

From a temporal perspective, referents can be classified into either products or processes. A product is 'a representation of the referent at a time instant' [Myers and Kaposi 2004] In essence, a product has no time duration. It is characterized by its attributes at a given instant. In contrast, a process is a 'representation of the referent as an activity which transforms an input product into an output product over a period of time' In essence, a process must have a non-zero duration. It is characterized by transforming an input product at the time of its arrival to the process ($t1$) to an output product produced at another time instant ($t2$) In this example, ($t2$ $t1$) is the duration of the process. As described in the following figure, this transformation can also be graphically described in terms of Product/Process graph notations as in [Myers and Kaposi 2004]



(a) Product/process graph type a

(b) Product/process graph type b

Figure 1.2. Product/process graphs [Myers and Kaposi 2004]

Finally, it is important to notice that the modelled system specified in the definition S = (E,R) represents a blueprint for that kind of referent, and not an instance of the blueprint. In other words, a referent class is described, when modelled, as black-box system using the

P/p method is described in terms of attributes types (attributes with defined measurement scale types) and relationships. However, attributes' values are given to a particular instance of that blueprint and not to the blueprint itself. This justifies having no room for attributes' values in the definition of the system description. Furthermore, it is also imperative to realize that processes do not consume their input products. For the sake of simplicity, Myers and Kaposi considered a process as a transformation of a *single input product* into a *single output product.* Such a definition eliminates the need to worry about product aggregation and the timing of their reception or production.

1.5 Overview

In the second chapter of this thesis, a detailed description of the FDA validation approach is given. Such a description appears in the form of evaluating the FDA validation approach. In essence, *quality planning* is detailed as the first activity in the FDA approach. This activity, as well as other activities, is defined in terms of some typical tasks that are recommended to be undertaken by software producers. In this sense, the FDA believes that conducting these typical tasks supports the conclusion that the software is validated. While considering quality planning as described in [FDA 2002], we describe the corresponding effort to deal with these typical tasks in the CC. The comparison between the FDA approach, in handling the quality planning activity, and the CC aims to highlight both the inconsistency and the vagueness of the FDA evaluation function. These deficiencies, as they appear in the quality planning activity, are the characteristics of the FDA approach for all other subsequent activities. Thus, the comparison is only restricted to this activity.

After highlighting the drawbacks of the FDA validation approach as they appear in the quality planning activity, modelling is introduced in the third chapter. In chapter three, we describe the way in which the validation activities described in the FDA guidance document [FDA 2002] will be modelled (specified) in chapter four.

In chapter four, we use the Product/process (P/p) method to model these validation activities. Thus, each activity of the FDA validation approach, starting with quality planning and followed by all other subsequent ones, is modelled as a process which composes other subprocesses that in turn have input and output products. Furthermore, we model (specify) the input and output products in terms of attributes, relationships among attributes, and their measurement scale types.

Finally, chapter five describes the conclusion as well as the future work which results from this thesis.

Chapter 2

FDA: Between Process & Product Evaluation

Quality Planning is the first activity in the FDA software development and validation approach. It is concerned with the preparation of the plans which will organize the software development and validation process. The FDA guidance requires the plans to 'identify the necessary tasks, procedures for anomaly reporting and resolution, necessary resources and management review requirements, including formal design reviews' [FDA 2002]. The FDA in [FDA 2002] describes this activity (process) in terms of typical tasks (subprocesses) to be undertaken throughout the process.

Section 2.2 starts by bringing to the reader's attention the difference between product evaluation and process evaluation. The description of product and process helps the reader distinguish between process evaluation and product evaluation. Such a distinction is crucial and our evaluation of the FDA guidance will be based on it.

In this context, new definitions of products and processes, other than the ones introduced in section 1.4, are given in section 2.2. These definitions significantly support the goal of distinguishing between process and product evaluation. However, we will stick to the former definitions of products and processes, as they were introduced in section 1.4, in chapter four and five as part of the Product/process (P/p) modelling.

In section 2.3, we consider *Quality Planning* as the first activity addressed in the FDA approach. While considering Quality Planning as the activity (process) of concern herein in this chapter, we exhibit the corresponding effort for its typical tasks in another evaluation methodology, the Common Criteria (CC) for Information Technology Security Evaluation. For each typical task defined in the FDA approach, the FDA standpoint is compared to the CC one. This comparison clearly shows how the CC has put much more effort in specifying and describing its evaluation approach than the FDA has done. It also highlights both the

inconsistency and the vagueness of the FDA evaluation function. Since these drawbacks are the characteristics of the FDA validation approach for all other subsequent activities, we restrict the comparison with the CC to the quality planning activity. Finally, we summarize this chapter in section 2.4.

2.1 Introduction

The Food and Drug Administration (FDA) is a public agency concerned with the validation of medical device software or software that is used to design, develop or produce medical devices. The FDA describes its recommendations about medical software validation through guidance documents. In these documents, the FDA recommends certain activities to be undertaken and certain deliverables to be prepared during the development of the medical software. This is believed by the FDA to achieve higher confidence in the software quality and, accordingly, better validation. Through its guidance documents, the FDA proposes that they have considered the least burdensome approach in medical software development. Thus this will yield efficient evaluation and certification for the developed software. In contrast, we see no explicit criterion for the FDA evaluation function. In other words, the FDA guidance documents do not explicitly state what are the entities to be measured, what are the attributes to be measured in these objects (evaluation evidences), what are the accepted values for these attributes or even how to measure them. This leaves much room for subjective judgment.

In section 2.2, we highlight the difference between products and processes. Such a difference results in a clear distinction between process evaluation and product evaluation. Bearing in mind the difference between process evaluation and product evaluation is important since our evaluation of the FDA guidance will be based on it. This criterion in software measurement, product versus process evaluation, has proven to be effective in the evaluation industry for its clarity, preciseness and support for the achievement of various goals. Several public and private institutions adopted this criterion. Among these are: NASA-Goddard Space Flight Center, Hewlett-Packard and Siemens. Case studies about their software measurement programs are described in [Fenton and Pfleeger 1997]. Having the reader understand this difference will help the reader to see the shortcomings in the FDA approach.

After that, we consider "quality Planning" activity since it is the first activity which is addressed in the FDA approach. For each typical task, which is recommended to be undertaken in the quality planning activity, we exhibit the corresponding effort in the Common Criteria (CC) for Information Technology Security Evaluation. This approach (CC) was developed through 'a collaboration among national security and standards organisations within Canada, France, Germany, the Netherlands, the United Kingdom and the United States of America, as a common standard to replace their existing security evaluation criteria' [CC 1999]. Despite its focus on evaluating software security capabilities, the systematic methodology, which the CC follows, clearly defines what to measure and how to measure.

Comparing the FDA approach with the CC highlights both the inconsistency and the

vagueness of the FDA evaluation function. For inconsistency, we show how the FDA approach fluctuates between process-oriented evaluation and product-oriented evaluation. For vagueness, we show how the FDA approach neither defines the attributes of the evidence to be measured, nor does it define the acceptable values. Furthermore, the comparison between the FDA and CC points out the issues that would have to be reconsidered by the FDA as steps of enhancement.

As described in the chapter one, the CC categorizes security requirements into Security Functional Requirements (SFRs) [CC 2006b] and Security Assurance Requirements (SARs) [CC 2006c]. For each category, the CC groups the requirements into classes, families and components. Members of the same class share a common focus whereas members of the same family under a specific class share a common objective, but differ in the level of potential assurance. This taxonomy will be clarified later in this chapter. In addition to the benefit of grouping related requirements, the efficient ability to trace requirements is another key advantage of this taxonomy.

2.2 Process versus Product Evaluation

Fenton and Pfleeger defined *processes* in [Fenton and Pfleeger 1997] as 'collections of software-related activities'. Hence the process usually has a time factor, i.e., it has a clear beginning and end. Attributes of the process are the inherent characteristics of the process. These attributes are meaningful descriptions over the process life time. Fenton and Pfleeger in [Fenton and Pfleeger 1997] use the term "internal process attributes" to describe the attributes which can be measured directly by examining the process on its own. We will refer to these attributes as static attributes. Having identified these attributes, the evaluation function should start considering the criteria to measure each of these attributes. The evaluation may take place at predefined time checkpoints or even continue over a time interval.

In contrast, dynamic process attributes are those attributes that can only be measured with regard to the way the process relates to its environment, i.e., the behaviour of the process is the main concern of the measurement activity rather than the process in itself. The values of these process attributes are not meaningful outside their operating environment. Examples of such attributes are: quality and stability. The values of dynamic process attributes (external attributes in [Fenton and Pfleeger 1997]) may depend on some values of the static process attributes (internal attributes in [Fenton and Pfleeger 1997]).

On the other hand, Fenton and Pfleeger defined products in [Fenton and Pfleeger 1997] as 'any artifacts, deliverables or documents that result from a process activity'. We will refer to a product as a deliverable of a process. In a sense, we will overload the word deliverable to mean any outcome of the process. Examples of deliverables (products) are: Software Requirements Specification (SRS) document, Software Design Specification document, software source code or even any intermediate outcome of the development process. Products are atemporal in the sense that product attributes can be measured at any time instant (though

the measured values may differ from one instant to another). The same notion of static and dynamic attributes is applicable to the deliverables (products). To be more concrete, the version number (as a static attribute) of the software (as a product) is related to the product itself. Whereas, reliability (as a dynamic attribute) of the software (as a product) is relative to the way in which the product (software) behaves in its operating environment.

To achieve objective judgment for all of the evaluation evidence (deliverables), attributes have to be specified. Once the deliverable's attributes are specified, a measurement criterion for each attribute has to be established. Such a criterion will annotate the desired attribute value. Having both the measurement criteria and the desired values documented at the outset of any "deliverable (product) development" will facilitate the development, interim validation and the formal evaluation of these attributes. In the same manner, tools and machines which may be used in the software production may have to be evaluated. Tools are considered as entities with attributes, exactly as for products, and hence their attributes have to be specified and tested.

For both processes and products, and in order to ensure successful validation, we have to identify the entity which we want to validate. Having recognized the entity to be validated as a process or a product, we can then consider identifying the entity's (process/product) attributes which have to be measured. After that, we have to consider the applicable and feasible criterion/criteria for measuring each attribute. As a result, we will have reached a common understanding about what to measure, how to measure it and what are the acceptable values for the measured attributes. Having such a measurement framework defined will decrease the level of subjectivity in the evaluation activity. The FDA approach lacks such a measurement framework. The FDA approach has no explicit definition of what entities (processes or products) are to be measured. Hence the developers of the medical software and the evaluators from the FDA side share no common understanding about what evidence will be inspected, what attributes in this evidence will be measured, what values are acceptable and what values are not. To show the feasibility of this approach (identifying attributes of entities) in measuring software, we describe an effective approach to help the project participants identify their attributes of interest along with their associated measurement criteria. This approach has been widely implemented in the evaluation industry.

As processes and products are the key objects of measurement, Basili, Caldiera and Rombach in [Basili, Caldiera, Rombach 1994] developed the Goal-Question-Metric (GQM). GQM is an engineering approach effective for the specification and the assessment of entities. The GQM model is hierarchical in the sense that it is layered into three main levels; conceptual level (goals), operational level (questions) and quantitative level (metrics). At the conceptual level, goals specify the objectives to be achieved by measuring some objects (products or processes) . At the operational level, questions (what, who and how) should be derived from each goal. Answers to these questions will determine whether the goals are being met or not. Finally, the quantitative level describes what type of data has to be collected for every question in addition to the measurement mechanism to ensure a quantitative answer for the assessment. A detailed example for the GQM is described in figure 4 in [Basili, Caldiera, Rombach 1994]. The key advantage of the GQM is that it enables us to identify the key attributes along with their measurement scales and procedures. These at-

tributes are the ones which are identified as being important for achieving the objectives and goals. Fenton and Pfleeger also considered the process maturity model to be used hand in hand with the GQM model. As the GQM helps to understand why we measure an attribute in a process, the process maturity model suggests whether or not we are capable of measuring the process in a practical way. Thus this supports the applicability of the GQM model. The main reason for considering this maturity model is that not all processes are at the same level of maturity, i.e., processes vary in terms of the visibility of input products and output products. The model is described in detail in [Fenton and Pfleeger 1997]. In this context, we want to emphasize that processes with clear input and output are our main concern, i.e., clearly-defined processes in terms of inputs and outputs. Otherwise, any process has to be refined (detailed or decomposed into subprocesses with clear input and output products).

2.3 Quality Planning: FDA versus CC

The FDA in [FDA 2002] describes a general software lifecycle model. Since the FDA does not favour any specific software development model, the FDA guidance [FDA 2002] describes its model in terms of activities (processes) which are common in any software development lifecycle model. For each of these activities, the FDA in [FDA 2002] lists a number of typical tasks to be undertaken. These activities are:

- Quality Planning
- Requirements
- Design
- Construction or Coding
- Testing by the Software Developer
- User Site Testing
- Maintenance and Software Changes

In contrast, the CC takes a product-oriented approach. In the CC, the Protection Profile (PP), Security Target (ST) and the Target Of Evaluation (TOE) are the key products (deliverables) for the evaluation activity. A description of these products will come later. The CC also requires the submission of other evaluation evidences (products). The assurance level describes the level of confidence required in the developed products. Despite the fact that these deliverables are not explicitly required by the CC in a single document, they are listed as input products for some of the evaluation action elements (the tasks to be undertaken during final product evaluation).

Following are the typical tasks of the quality planning activity (process), the tasks which are concerned with the preparation of the following plans as the FDA guidance recommends

in [FDA 2002]. After describing the FDA standpoint for each task, we will show the CC standpoint, the way in which the CC handles these issues. This comparison helps the reader to recognize the virtues of the CC systematic approach over the FDA approach.

2.3.1 Risk (Hazard) Management Plan

The preparation of a risk (hazard) management plan is a typical task to be undertaken in the FDA approach. Therefore, we discuss the FDA standpoint on this, followed by the CC standpoint.

FDA standpoint

The FDA guidance in [FDA 2002] does not provide an explicit definition for the term risk. Instead, it uses the word "risk" interchangeably with the word "hazard". The FDA in [FDA 2005] considers risk as 'the product of severity of injury and the probability of its occurrence'. As software failures are systematic, probability of occurrence is not considered. Hence, the FDA in its guidance document recommends to 'base risk estimation for a software device on the severity of the hazard resulting from failure, assuming that the failure will occur' [FDA 2005].

In more detail, the FDA in [FDA 2002] requires the preparation of a plan, where the development tasks are identified along with their estimates, the resources needed, dependencies (if any) and the role which the risk management will play. In other words, the risk management plan has to identify the risks during the whole development process, the effect these risks may have and the actions to be undertaken as per these risks. The FDA in [FDA 2005] recommends the presentation of the risk analysis information in a 'tabular form with a line item for each identified hazard, where each line item should include' [FDA 2005]:

- identification of the hazardous event
- severity of the hazard
- cause(s) of the hazard
- method of control (e.g., alarm, hardware design)
- corrective measures taken, including an explanation of the aspects of the device design/requirements, that eliminate, reduce, or warn of a hazardous event, and
- verification that the method of control was implemented correctly

It is obvious that the FDA has described the quality planning process in terms of some other subactivities (the preparation of a risk management plan in this case). Nevertheless, the FDA neither describes the properties which this process (preparing the risk management plan)

should have, nor describes the way in which this process should be evaluated. Instead, the FDA switches to a product-oriented perspective to consider the product (risk management plan) of this activity (the preparation of a risk management plan). In spite of this switch, the FDA guidance document roughly describes some properties of the deliverable (the structure of the deliverable) without defining the method that will be used to evaluate this deliverable. As a result, the developer and the evaluator may argue about the quality of this deliverable since there is no reference to base it on. This switch between a process-oriented perspective and a product-oriented perspective is the common behaviour of the FDA guidance document that we will be seeing throughout this chapter.

CC standpoint

In this context, the CC defines vulnerability as 'a weakness in the Target Of Evaluation (TOE) that can be used to violate the Security Functional Requirements (SFRs) in some environment' [CC 2006a]. By TOE, the CC means 'a set of software, firmware and/or hardware possibly accompanied by guidance' [CC 2006c] (corresponding to the medical software in the FDA approach). Furthermore, the CC requires the identification of exploitable vulnerabilities in the operational environment, potential vulnerabilities and residual vulnerabilities. The taxonomy of the CC describes Security Assurance Requirements (SARs) in terms of action elements for the developer, action elements for the "content & presentation" of the submitted evaluation evidence and as action elements for the evaluator. In this context, the CC requires that the developer shall [CC] provide the TOE for testing. The TOE shall [CC] be suitable for testing and the evaluator shall [CC] examine sources of information to support the identification of potential vulnerabilities in the TOE. After that the evaluator shall [CC] conduct penetration testing to 'confirm that the potential vulnerabilities cannot be exploited in the operational environment for the TOE' [CC 2006c]. The CC uses the auxiliary verb *"shall"* to refer to mandatory work which has to be undertaken to ensure the correctness of evaluation and, accordingly, the verdicts assigned to products. Whereas, the CC uses the auxiliary verb *"should"* to mean strongly preferred.

In more detail, the vulnerability assessment is influenced by the CC *Vulnerability Assessment* class (AVA). This class addresses the possibility of exploitable vulnerabilities introduced in the development or the operational environment of the TOE which 'could allow attackers to violate the security functional requirements' [CC 2006c]. The components in the Vulnerability Analysis family (AVA_VAN) are categorized based on 'the increasing rigour of vulnerability analysis by the evaluator, and increased levels of attack potential required by an attacker to identify and exploit the potential vulnerabilities' [CC 2006c] as follows:

- Vulnerability survey (AVA_VAN.1): 'a vulnerability survey of information available in the public domain is performed by the evaluator to ascertain potential vulnerabilities that may be easily found by an attacker' [CC 2006c].
- Vulnerability analysis (AVA_VAN.2): 'a vulnerability analysis is performed by the evaluator to ascertain the presence of potential vulnerabilities' [CC 2006c].

Penetration testing is to be performed by the evaluator to 'confirm that the potential vulnerabilities cannot be exploited in the operational environment for the TOE' [CC 2006c], but with differences in the assumption of the level of attack potential. The following three components are similar to the previous one but with assuming an increased attack potential.

- Focused vulnerability analysis (AVA_VAN.3)
- Methodical vulnerability analysis (AVA_VAN.4)
- Advanced methodical vulnerability analysis (AVA_VAN.5)

The Evaluation Assurance Level (EAL) of the TOE is the determinant of which vulnerability assessment component is to be undertaken. For example, a vulnerability survey is sufficient for a TOE with an EAL1. Whereas, a TOE with EAL2 requires vulnerability analysis.

2.3.2 Configuration Management Plan

The preparation of a configuration management plan is a typical task to be undertaken in the FDA approach. Therefore, we discuss the FDA standpoint on this, followed by the CC standpoint.

FDA standpoint

The FDA guidance in [FDA 2002] requires the preparation of a configuration management plan. Such a plan has to be developed in order to 'guide and control multiple parallel development activities and ensure proper communication and documentation as controls are necessary to ensure positive and correct correspondence among all approved versions of the specifications documents, source code, object code, and test suites that comprise a software system. The controls also should ensure accurate identification of, and access to, the currently approved versions' [FDA 2002]. The FDA guidance in [FDA 2005] states that the usage of the verb *"should"* is to mean 'something that is suggested or recommended, but not required' [FDA 2005].

In this case, not only does the FDA guidance not describe the attributes of the process (preparing a configuration management plan), but also does not describe the attributes of the product (configuration management plan).

CC standpoint

In this context, the CC requires the submission of the following two pieces of evaluation evidence (products):

- Configuration Management (CM) documentation
  CM documentation provides a description of the configuration management system. Moreover, the document describes the method which the CM system uses to 'uniquely identify the configuration items' [CC 2006d]. The capabilities, as characteristics (attributes), of the CM system have to be specified. Thus these capabilities should [CC] guarantee that only authorized access takes place to the configuration items.

- Configuration List documentation
  The Configuration List documentation describes the objects to be identified as configuration items. The configuration items may include: 'the TOE and the evaluation evidence required by the SARs; parts of the TOE; the implementation representation; security flaws; and development tools and related information' [CC 2006c].

In more detail, the aforementioned deliverables (products) are influenced by the *CM Capabilities* family (ALC_CMC) and the *CM Scope* family (ALC_CMS), correspondingly. These families are elements of the *Life Cycle Support* class (ALC). The *CM Capabilities* family provides a 'more detailed description to the management of the configuration items' [CC 2006c]. The *CM Scope* family requires 'a minimum set of configuration items to be managed in the defined way' [CC 2006c]. More detail is provided in the following two sections.

CM Capabilities (ALC_CMC)

CM establishes assurance by 'requiring discipline and control in the processes of refinement and modification of the TOE and the related information' [CC 2006c]. Therefore, the CM Capabilities (ALC_CMC) family describes the capabilities (characteristics) of the configuration management system. The components of this family are made to ensure that [CC 2006c]:

- the TOE is correct and complete before it is sent to the consumer
- no configuration items are missed during evaluation
- no unauthorized modification, addition, or deletion of TOE configuration items takes place

The components of this family are categorized based on the properties of the CM: 'the CM system capabilities, the scope of the CM documentation and the evidence provided by the developer' [CC 2006c]. These are:

- Labeling of the TOE (ALC_CMC.1)
- Use of a CM system (ALC_CMC.2)
- Authorization controls (ALC_CMC.3)
- Production support, acceptance procedures and automation (ALC_CMC.4)

- Advanced support (ALC_CMC.5)

The EAL of the TOE is also the determinant of which CM capability is to be undertaken. For example, the *Authorization Controls* component has to be implemented for EAL3, but not for EAL1 or EAL2.

CM Scope (ALC_CMS)

The *CM Scope* family identifies what items are to be 'included as configuration items and hence to be placed under the CM requirements of CM capabilities' [CC 2006c]. The components in this family are categorized on the basis of which of the following are required to be included as configuration items' [CC 2006c]:

- TOE and the evaluation evidence required by the software assurance requirements in the ST be included in the configuration list (ALC_CMS.1: TOE CM coverage)
- Parts of the TOE (ALC_CMS.2: Parts of the TOE CM coverage)
- The implementation representation (ALC_CMS.3: Implementation representation CM coverage)
- Security flaws (ALC_CMS.4: Problem Tracking CM coverage)
- Development tools and related information (ALC_CMS.5: Development tools CM coverage)

Again, the EAL of the TOE is the determinant of which CM scope is to be undertaken. For example, the *Implementation Representation* component has to be implemented for EAL3, but not for EAL1 or EAL2.

The CC has a more systematic approach in the sense that it starts by identifying what pieces of evidence are required, their importance and the characteristics of these pieces of evidence. Furthermore, the evaluation methodology specifies the evaluation effort according to the EAL required for the final software product. The effort is divided into action elements to be initially undertaken by the developer, along with others to be undertaken by the evaluator. This framework facilitates the product development as well as the product evaluation.

2.3.3 Software Quality Assurance Plan

The preparation of a software quality assurance plan is a typical task to be undertaken in the FDA approach. It includes the preparation of other documents. Therefore, we discuss the FDA standpoint on this, followed by the CC standpoint.

Software Verification and Validation Plan

The preparation of a software verification and validation plan is a subactivity of the preparation of the software quality assurance plan. It also includes other subactivities. These are:

- Verification and Validation Tasks, and Acceptance Criteria
- Schedule and Resource Allocation (for verification and validation tasks)
- Reporting Requirements

FDA standpoint

The FDA guidance in [FDA 2002] defines *software verification* as 'providing objective evidence that the design outputs of a particular phase of the software development lifecycle meet all of the specified requirements for that phase. Software verification looks for consistency, completeness, and correctness of the software and its supporting documentation, as it is being developed, and provides support for a subsequent conclusion that software is validated' [FDA 2002]. The FDA guidance in [FDA 2002] considers *software validation* to be 'confirmation by examination and provision of objective evidence that software specifications conform to user needs and intended uses, and that the particular requirements implemented through software can be consistently fulfilled' [FDA 2002]. According to the FDA guidance in [FDA 2002], procedures should [FDA] be created to describe the way in which anomalies are reported, fixed and approved. Moreover, the management should [FDA] identify the documents used for reporting, describing their structure and format. The notation "[FDA]" is used to emphasize that the auxiliary verb (should) is introduced by the FDA in its guidance (similarly for the CC), and not by the author of this thesis.

The FDA guidance documents do not describe the process of preparing the quality assurance plan. Accordingly, evaluating this process is not tangible. Instead, the guidance documents describe the product (software assurance plan) by describing the structure of this product. For instance, the FDA guidance requires the developer to describe the verification and validation effort made during the development, but the guidance says nothing about the way in which the FDA staff are going to be assessing this document. Moreover, the FDA guidance document asks for objective evidence for the validation activity without defining what objective evidence is accepted by the FDA evaluation staff. Also, the guidance documents may seem to describe some attributes but the guidance documents give no definition of these attributes. For example, the FDA guidance documents specify completeness and consistency as desired attributes which the verification and validation document should exhibit, but the guidance documents do not even give an informal definition for completeness and consistency, i.e., what are the measurement scales for completeness & consistency and what are the criteria which the FDA staff will be following to determine whether this document is complete, consistent or not. This attitude of the FDA guidance documents introduces vagueness in addition to the previous shortcoming which lead to subjective judgment.

CC standpoint

In this context, the CC defines evaluation as 'an assessment of a Protection Profile (PP), a Security Target (ST) or a TOE, against defined criteria' [CC 2006a]. A PP is 'an implementation-independent statement of security needs for a TOE type' [CC 2006a], while a ST is an 'an implementation-dependent statement of security needs for a specific identified TOE' [CC 2006a]. For comparison purposes with the FDA, the evaluation of the TOE is our main concern. Evaluation in the CC is 'carried out by an accredited evaluation facilities (testing laboratories)' [CC 1999]. Such evaluation has three phases:

- Phase 1: Preparation
- Phase 2: Conduct of evaluation
- Phase 3: Conclusion

At the very beginning (preparation phase), when an evaluation seems to be feasible, 'a list of the required evaluation deliverables, a plan for each delivery and a project plan for the evaluation process are established' [CC 1999].

In more detail and as described before, the software assurance requirements are specified in terms of action elements for the developer, for the "content and presentation" of the submitted evaluation evidence and for the evaluator. Such a description clarifies the evaluation tasks along with their acceptance criteria. In essence, the CC Evaluation Methodology (CEM) [CC 2006d], a companion document to the CC, has been prepared in order to develop an agreed methodology for conducting evaluations which apply the CC. The evaluator's action elements in [CC 2006c] correspond to action in [CC 2006d] which are broken down into more detailed work-units (evaluation activities). In other words, the CEM defines what are the products to be measured along with how they are going to be measured. This description is explicit and accessible to the developer. Therefore, this description will be the reference which the evaluation activities will be based on.

For the reporting requirements, the CEM specifies minimum characteristics for evaluation outcomes. More specifically, the CEM specifies minimum characteristics for the Observation Reports (ORs) and the Evaluation Technical Reports (ETRs). The CEM defines the OR as 'a report written by the evaluator requesting a clarification or identifying a problem during the evaluation' [CC 2006d]. For the ETR, the CEM in [CC 2006d] defines it as 'a report that documents the overall verdict and its justification, produced by the evaluator and submitted to an overseer'. The characteristics of these two products are explicitly defined by the CC.

Formal Design Review Requirements

This subactivity has been listed in the FDA guidance [FDA 2002] under the preparation of a software quality assurance plan. Therefore, we discuss the FDA standpoint on this, followed by the CC standpoint.

FDA standpoint

The FDA guidance in [FDA 2002] does not give an explicit definition for a formal design review. Indeed, the FDA guidance refers to a formal review as an independent review which includes participation from other parties, outside the development team (third-party). The FDA provides no specification to indicate what to evaluate, how to evaluate it and even what the range of the acceptable results might be.

CC standpoint

In the CC, the formal design review is performed by the evaluation facility (testing laboratory) which evaluates the products. The evaluation tasks along with their acceptance criteria are explicitly specified in [CC 2006d]. Again, this leaves no room for subjective judgment. This evaluation takes effect after several interim evaluations by the developer, and before the TOE is submitted to the certification authorities (national schemes).

Other Technical Review Requirements

The FDA guidance in [FDA 2002] does not define any of the "other technical review requirements". In fact, other technical reviews may be organisational or product-specific. Nevertheless, no general evaluation model is described for such deliverables.

2.3.4 Problem Reporting and Resolution Procedures

The preparation of problem reporting and resolution procedures is a typical task to be undertaken in the FDA approach. Therefore, we discuss the FDA standpoint on this, followed by the CC standpoint.

FDA standpoint

The FDA guidance requires having in place problem reporting and resolution procedures during the development of the medical software. This is believed by the FDA to result in software with better quality for evaluation. Again, the properties of "preparing a problem reporting and resolution procedures" process are not described. Moreover, the properties of the "problem reporting and resolution procedures" product (document) are also not described.

CC standpoint

In this context, the CC requires the preparation of the following deliverables (products):

- Flaw Remediation Procedures Documentation

  This document provides a description about the way of discovering flaws and processing them by the developer. The policies and procedures the developer used to correct flaws have to be documented. These are all subject to assessment by the evaluator. The evaluation activities to be undertaken to evaluate the deliverables (products) are specified as detailed work-units to be performed by the evaluator in [CC 2006d].

- Flaw Remediation Guidance Documentation

  This document acts as a guideline for discovering flaws, reporting them and then fixing them. Again, the evaluation activities to judge this document are also specified as detailed work-units to be performed by the evaluator.

The CC requires "flaw remediation procedures documentation" and "flaw remediation guidance documentation" to be prepared. These documents represent the deliverables (products) which will be subject to the CC evaluation as described by the Common Evaluation Methodology in [CC 2006d].

2.3.5 Other Support Activities

The FDA guidance in [FDA 2002] mentions this task with no further description. Indeed, other support activities may be organisational or product-specific. However, they should be described, in the same manner, in terms of their attributes and their measures.

In addition to the above typical tasks addressed in [FDA 2002], the FDA in [FDA 2005] recommends the inclusion of the following documents in the premarket submissions for software devices. Therefore, we consider these documents in this section as they are related to the *Quality Planning* process.

2.3.6 Statement about the level of concern

The FDA guidance document [FDA 2005] recommends the submission of a statement about the level of concern. Therefore, we discuss the FDA standpoint on this, followed by the CC standpoint.

FDA standpoint

The FDA in [FDA 2005] recommends the preparation of level of concern documentation. Such a documentation has to classify the level of concern of the software into major, moderate or minor concern. The level of concern is determined by an estimation of the severity of the injury, before hazard mitigation, which the software may cause to the patient or even the operator of the medical device. The FDA guidance helps the software development team in determining the level of concern through questionnaires as detailed in [FDA 2005].

Only for this product, the FDA guidance identifies an attribute (level of concern), discusses its possible values (major, moderate or minor) and elaborates on the measurement procedure which has to be followed to determine these values.

CC standpoint

Although the FDA guidance requires the preparation of "level of concern" documentation, it does not relate this level of concern afterwards to the effort of evaluating the product. In contrast, the CC provides seven assurance packages, on 'a rising scale of assurance' [CC 2006b], commonly known as the Evaluation Assurance Levels (EALs). These are:

- Functionally tested (EAL1)
- Structurally tested (EAL2)
- Methodically tested and checked (EAL3)
- Methodically designed, tested and checked (EAL4)
- Semiformally tested and checked (EAL5)
- Semiformally verified design and tested (EAL6)
- Formally verified design and tested (EAL7)

As described earlier, the EAL determines which components of the families to be implemented for each class and accordingly it determines the level of evaluation.

2.3.7 Software Description

The FDA guidance document [FDA 2005] recommends the submission of a description about the software. Therefore, we discuss the FDA standpoint on this, followed by the CC standpoint.

FDA standpoint

The FDA guidance as described in [FDA 2005] recommends the preparation of a 'comprehensive overview of the device features that are controlled by the software and describe the intended operational environment'. Such documentation should [FDA] include information about [FDA 2005]:

- programming language
- hardware platform
- operating system (if applicable)
- use of Off-the-Shelf software (if applicable).

The FDA guidance recommends the preparation of this deliverable. Nevertheless, it does not describe the process of preparing this deliverable (product) or even the attributes of the deliverable itself. In other words, it provides no detail about the way in which this deliverable will be evaluated.

CC standpoint

In the CC, each of the Security Target (ST) and Protection Profile (PP) has a section which provides a description of the TOE (TOE summary specification). Moreover, the operational environment of the TOE is described in the "security objectives for the operational environment" section in both the ST and PT. Furthermore, the assumptions and the constraints of the TOE environment have a section for their documentation.

The CC has two assurance classes dedicated to evaluate the ST and the PT. These classes are: "security target evaluation" and "protection profile evaluation". In these assurance classes, the evaluation effort is expressed in terms of components organized into: developer action elements, content and presentation action elements and evaluator action elements.

2.4 Summary

In this chapter, we questioned the clarity and feasibility of the FDA approach for evaluating medical software. We did this by examining the guidance documents which were published by the FDA, for both the medical software industry and FDA staff, to express the FDA's current thinking about the validation of medical software. In a more concrete sense, we analysed the FDA guidance documents based on considering the product versus process evaluation criterion. This criterion in software measurement, product versus process evaluation, has proven to be effective in the evaluation industry for its clarity, preciseness and support for the achievement of various goals. It has also been adopted by several public and private

institutions in their projects, like NASA-Goddard Space Flight Center, Hewlett-Packard and Siemens.

We believe that the FDA validation approach leads to confusion in the minds of both software producers and evaluators and it may well lead to lower quality software than desired being certified. In essence, the developer and the evaluator may argue about the quality of a deliverable since they have no reference to base it on. To demonstrate these observations, we presented the "Quality Planning" activity as the first activity described in the FDA guidance document [FDA 2002]. We described the effort which the FDA recommends to be performed. On the other hand, we exhibited the corresponding effort for this activity in another product-oriented evaluation approach, the Common Criteria(CC) for Information Technology Security Evaluation. This approach (CC) was developed through 'a collaboration among national security and standards organisations within Canada, France, Germany, the Netherlands, the United Kingdom and the United States of America, as a common standard to replace their existing security evaluation criteria' [CC 1999]. Despite its focus on evaluating software security capabilities, the systematic methodology, which the CC follows, clearly defines what to measure and how to measure (these are exactly the aspects which the FDA approach lacks).

As its main objective, the comparison between the FDA approach and the CC aims to highlight both the inconsistency and the vagueness of the FDA evaluation function, which may result in subjective judgment, as the acceptance criteria are not defined in terms of well defined measures of relevant attributes. For evidence of inconsistency, we showed how the FDA approach fluctuated between product and process validation. Furthermore, we showed how the FDA guidance document had no consistent approach to tackle the validation of deliverables (products). For evidence of vagueness, we described how the FDA guidance documents neither defined the attributes of the evidence to be measured, nor did they define their measures or even their acceptable values.

Chapter 3

Modelling Structure

This chapter is concerned with describing the structure of the modelling mechanism. Such a structure will be followed in the next chapter as a way to propose a simplified representation for the FDA validation approach. The chapter starts with reminding the reader about the FDA validation approach. Similarly, it recaps the Product/process (P/p) methodology which has already been described in section 1.4. As a blueprint for the modelled products (referents), each product is described in terms of attributes (characteristics of interest) which characterize it. Then, each of these attributes along with its measurement scale type is defined, and the relationships among attributes are annotated. Such a specification corresponds to the P/p methodology in the sense that it describes each product as a black box system (S_B=(E_B,R_B)) which is characterized by attributes as elements of the E_B set, whose measures are defined. Furthermore, the relationships between these attributes as elements of the R_B set are also described.

The measurement procedures (the ways in which these attributes are measured and given values) are not specified. Leaving out the measurement procedures to be specified by software producers as per their measurement methodology preserves flexibility as a characteristic which the FDA aims to have. However, some suggestions and comments on the measurement procedures are given in this thesis. Nevertheless, the auditing process which has to be undertaken by the FDA staff over the products which have been already validated by software producers should be specified by the FDA. Having such an evaluation (auditing) process specified with clear pass/fail verdicts would ensure a systematic and objective certification approach. Indeed, this auditing process is outside the scope of this thesis. However, we will touch on it in the future work (section 5.2).

3.1 Introduction

The Food and Drug Administration (FDA), as described in chapter two, describes a general software lifecycle approach for software development. During software development, some validation activities are recommended to be conducted and some deliverables are also recommended to be prepared. These activities and deliverables support to help in reaching a conclusion that the developed software is validated. Since the FDA does not favour any specific software development approach, the FDA guidance [FDA 2002] describes this approach (lifecycle model) in terms of activities (processes) which are common in any software development approach. For each of these activities, the FDA in [FDA 2002] lists a number of typical tasks that are recommended to be undertaken. In other words, the FDA in its guidance believes that performing these typical tasks of each activity will constitute by construction a good evidence for validation of the software. These activities, as previously described, are: *Quality Planning, Requirements, Design, Construction or Coding, Testing by the Software Developer, User Site Testing, and Maintenance & Software Changes.*

The FDA guidance documents use natural language in specifying recommendations because of the wide audience to which they are addressed. However, the specification is not explicit and precise enough to be able to impose a contract (obligation) between the two parties (software producers and evaluators). Furthermore, we see no explicit criterion for the FDA evaluation function. In other words, the FDA guidance documents do not explicitly state what are the entities to be measured (processes or products of these processes), what attributes that characterize the products/processes are to be measured, what kinds of scale are needed to measure these attributes and even how will evaluators (certifiers) after that measure these attributes. The inconsistency and vagueness of the FDA approach have been highlighted in the second chapter, as they appear throughout the FDA guidance [FDA 2002]. As this inconsistent and vague specification is the characteristic of the whole FDA guidance [FDA 2002], we restricted highlighting these pitfalls to the first activity in the FDA approach (quality planning). The drawbacks of the FDA approach became obvious when we compared it with the Common Criteria. Hence, we consider modelling in this chapter as an effective way for representing the products of the FDA activities, not only the ones of the quality planning activity, in a simpler and more explicit way.

Product/process (P/p) modelling takes a systematic approach in the way it handles any engineering problem. It proposes to represent any engineering problem in terms of a system of components with relationships among them. These components can be products or processes. Myers and Kaposi define a product as 'a representation of a referent at a time instant' [Myers and Kaposi 2004], where the referent is the object of our interest. On the other hand, they define a process as 'a representation of the referent as an activity which transforms an input product into an output product over a period of time' [Myers and Kaposi 2004].

3.2 Conceptual Framework

In this section, we describe the general framework of the modelling approach. The framework describes the purpose of modelling, objects which are subject to evaluation, definitions as given in the FDA guidance documents and the attributes which we define to describe these objects (products).

3.2.1 Purpose of Modelling

The main purpose of modelling is to provide a simplified representation for the products of the FDA processes. These products are recommended to be prepared by the FDA in [FDA 2002] and hence are subject to evaluation. The simplified representation facilitates the clarification of attributes that characterize the products. Moreover, it addresses their measurement scales and the relationships among them as they were not defined in the FDA guidance documents before.

3.2.2 Format of Activity Description

The products which the FDA guidance [FDA 2002] recommends to be prepared represent our referents (entities of concern). These products are either considered as black box systems or as structural systems [Myers and Kaposi 2004]. A product is considered as a black box system when it stands on its own and it is not composed of any other sub-products. Thus, the attributes which characterize the product along with their associated measures suffice to model (specify) the product. On the other hand, a product is considered as a structured system when it is defined or structured in terms of other sub-products that can be modelled themselves as black-box systems. Thus and after decomposition, each product, as a black box system, is represented in terms of interrelated attributes. These attributes can be primitive or composite. A composite attribute's measure is defined in terms of other attributes. However, only the attributes that contribute to the purpose of modelling are considered, i.e., an attribute may characterize a product, nevertheless it may add no value to the purpose of modelling. Consequently, it will not be considered in our modelling process.

3.2.3 Definitions

For each typical task (process) of the FDA activities, the "definitions" section aims to introduce any term related to the process as given in the FDA guidance documents. Sometimes, the lack of definitions, need for definitions or even the introduction of some new definitions will be pointed out.

3.2.4 Attributes

This section describes the attributes that characterize the products. Attributes represent the elements of the E_B set as per the formal definition of a black-box system (S_B=(E_B,R_B)). In this context, products that take the form of documents share some common attributes. These attributes are:

- **Size**; describes, in an absolute (counting) scale, the number of pages with a specified font type and size for paragraphs and headers.
- **Length**; describes, in an absolute (counting) scale, the words count of the document. Automated scripts can be used to automatically calculate the number of words in a document.
- **Clarity**; describes, in a relative (ratio) scale, the complement to the percent of undefined terms in the document relative to its length.
- **Syntactic correctness**; describes, in an absolute (counting) scale, the number of syntactic errors in the document. For grammatical issues, a particular lexicon has to be approved (referenced) by the certifier (FDA). Similarly, orthographical issues can be resolved by referring to a language standard, for example, requiring American English to be used in writing documents.
- **Ambiguity**; describes, in an absolute (counting) scale, the number of statements which are obscure. A suggested procedure to discover ambiguous statement would be whether or not the statement is subject to different interpretations that are not considered correct by a consensus of document readers.

The previously described attributes represent the ones that are common among document products. However, other ones may be project or organisation-specific. Thus, and for the products recommended to be prepared by the FDA, product-specific attributes are considered per product in the next chapter.

3.2.5 Relationships

In addition to products' attributes, relationships among these attributes represent the elements of the R_B set as per the formal definition of a black-box system (S_B=(E_B,R_B)). The set R_B has at least two mandatory relationships among the attributes of any product. These are:

- **The co-attribute relationship** (r_c); asserts that attributes (elements) in the set of all attributes belong to the same referent (product of concern).
- **The time-stamp relationship** (r_t); specifies the time instant at which the attributes were measured and thus assigned their specified values.

In brief, modelling (specifying) any product in terms of attributes and relationships corresponds to the P/p methodology in the sense that it describes the product as a black box system (S_B=(E_B,R_B)) which is characterized by attributes as elements of the E_B set, whose measures are defined. Furthermore, the relationships among these attributes, as elements of the R_B set, are also specified.

Chapter 4

Contract by Systems Modelling

The second chapter focused on highlighting the drawbacks of the specification of the quality planning activity as they appear in [FDA 2002]. In this chapter, we consider modelling as a way for representing the products of the FDA activities in a simpler, yet more explicit way. Modelling is proposed to specify the products as per the modelling structure described in the third chapter. Thus, and through sections 4.1 to 4.7, the activities defined in the FDA approach are considered each in turn. For each activity, as the process of interest, we consider the deliverables which are recommended to be prepared during the typical tasks (subprocesses) of each activity (process) as our products of concern. Each product is then modelled. The modelling activity, as its main goal, does not aim to impose a definitive proposal on the FDA. Instead, it aims to illustrate the existing problems and propose a way in which these matters could be handled. Reconsideration of these issues by the FDA, in addition to defining an explicit certification criteria for assessing (auditing) the effort undertaken by software producers to develop and validate these products, will lead to a more objective evaluation function. Indeed, an objective evaluation function, with clear evaluation process from the FDA side, guarantees the same conclusion (result) based on a particular input at a specific instant regardless of the person performing the audit.

4.1 Quality Planning

The quality planning activity is the first activity (process) which is recommended to be undertaken in [FDA 2002]. This process is described in terms of other typical tasks (subprocesses). These subprocesses are concerned with the preparation of the following products:

- Risk (Hazard) Management Plan
- Configuration Management Plan

- Software Quality Assurance Plan
  - Software Verification and Validation Plan
  - Formal Design Review Requirements
  - Other Technical Review Requirements
- Problem Reporting and Resolution Procedures
- Other Support Activities

Thus, each process is described in terms of its input and output products. After that, the products of these processes are modelled each in turn.

4.1.1 Risk Management Plan

The preparation of a risk management plan is a typical task of the quality planning activity. The products of this process are our concern.

Definitions

The FDA in [FDA 2002] does not give an explicit definition for the term "risk". Instead, it uses the term risk interchangeably with the term "hazard". However, the FDA in [FDA 2005] defines risk as 'the product of severity of injury and the probability of its occurrence'. As software failures are systematic, the probability of occurrence is not considered. Hence, the FDA in its guidance document recommends basing risk estimation for a software device on the 'severity of the hazard resulting from failure, assuming that the failure will occur' [FDA 2005]. The FDA in [FDA 2005] recommends the presentation of hazard (risk) analysis information in a 'tabular form with a line item for each identified hazard, where each line item should include: identification of the hazardous event; severity of the hazard; cause(s) of the hazard; method of control (e.g., alarm, hardware design); corrective measures taken, including an explanation of the aspects of the device design/requirements, that eliminate, reduce, or warn of a hazardous event; and verification that the method of control was implemented correctly' [FDA 2005]. In this context, risk analysis should not be restricted to the risks introduced only by the medical software itself. As detailed in [FDA 1999], risk analysis should [FDA] also include the risks which may be presented by the OTS software.

Attributes

The following figure describes the "preparation of the risk management plan" process. This process has the "System Description" as its input product and the "Risk Management Plan" as its output product. Product/process (P/p) graph type (a) is used herein to describe this

process and it will be used to describe atomic processes throughout this chapter Whereas, P/p graph type (b) will be used to illustrate the big picture of the high level activities (processes)

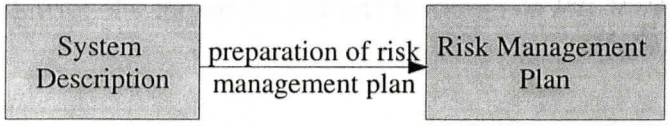


Figure 4.1 Preparation of risk management plan

In this context, the risk management plan, as the output product of this process, has to be modelled in terms of attributes and their associated measurement scales. However, system description as the input product for this process is a document which describes the medical software to be developed. In addition to the document-related attributes which this product has, system description can be modelled (specified) in terms of the following attributes:

- **System name**; describes, in a nominal scale, the name of the system (medical software) to be developed.
- **Purpose (goal) of the software**; describes the set of goals, each described as pieces of text, to be achieved by developing the software. The text itself can be seen as a product with its own characteristics (attributes) However, we do not go into that level of detail in our modelling process.
- **Operating environment description**, a textual description of the physical environment in which the software operates.
- **List of Assumptions description**, the set of all assumptions, each described as pieces of text, which form the basis for the software to operate.
- **List of Constraints description**, the set of all constraints, each described as pieces of text, which limit the software functionalities.
- **Underlying hardware description**, the set of all hardware, each listed as pieces of text, needed for the software to function. Each hardware description as an element in the set of all hardware may be considered as a sub-product of the system description and, accordingly, it can be specified by its key attributes. However, hardware systems are usually clearly and precisely specified (by their producers) through their specification. Therefore, the specification of these products have to be referenced herein.
- **Partner Applications description**, the set of all software applications, each listed as pieces of text, on which the medical software depends to operate properly

Similarly to hardware products, partner applications (software products) with which the medical software interacts are products by themselves. They can be described in terms of attributes as per the specification given by their producers.

These attributes represent the elements of the E_B set as per the formal definition of a black-box system (S_B=(E_B,R_B)).

Relationships

In addition to co-attribute relationship and time-stamp relationship as the key two mandatory relationships of the R_B set, the following relationships exist among the aforementioned attributes:

- **Operating environment–hardware consistency**; the operating environment must meet the requirements of the underlying hardware to operate. For instance, room temperature of the operating environment must go with the requirements of the underlying hardware.
- **Partner Application–hardware consistency**; the underlying hardware must meet the minimum requirements of partner applications, especially in case they have to be operating on top of the same hardware of the medical software.

The information specified in the system description as the input product for this process forms the basis for the development of the output product. In other words, the risk of the medical software significantly depends on the operating environment of the software, assumptions & constraints for the software to operate, and all other cooperating products, both hardware and software, which the software needs in order to operate properly. In this sense, and in addition to the common document attributes (size, clarity, syntactic correctness and ambiguity), the set of all identified risks represents the main attribute which characterizes the output product (risk management plan document) of this process (the preparation of a risk management plan). Thus, it is modelled as per the P/p modelling methodology.

Identified set of risks

The identified set of risks represents the main attribute which characterizes the risk management plan document. However, risk itself is a product which can be characterized by several attributes. In other words, risk description is a sub-product of the risk management plan document. In this sense, risk description is characterized by the following attributes:

- **Risk Id**; a unique number (or any combination of numbers and letters) which identifies a particular risk. Such an identifier is used to reference and track the risk.

- **Risk category**; describes, in a nominal scale, the category (type) into which the risk can be classified. For instance, a risk can be a hardware risk, software risk, operating risk, etc.

- **Risk severity**; the level of damage which the risk may cause to the patient or even to the medical device operator. Only for this attribute, the FDA guidance [FDA 2005] helps identify the severity level through a number of questionnaires. Eventually, the risk severity is classified, on an ordinal scale, into minor, moderate or major.

- **Trigger point**; the event which indicates the occurrence of the risky situation. This attribute tends to be more software (product) specific. The FDA guidance does not provide measurement procedures for this attribute. Thus, historical data or even the literature could be a significant source of information in defining these measurement procedures.

- **Effort**; the planned effort documented to be undertaken in order to resolve the risk. Such an effort is described in terms of a list of activities. Effort documentation itself is the sub-product of the risk description which can be described in terms of other attributes [Dobson 2003]:

  - *Prevention effort*; describes the list of activities, each described as pieces of text, which have to be undertaken in advance to prevent the occurrence of the injury.
  - *Mitigation effort*; describes the list of activities, each described as pieces of text, which have to be undertaken to mitigate the effect of the injury.
  - *Handling effort*; describes the list of activities, each described as pieces of text, which have to be undertaken to deal with the effect of the injury, considering its occurrence.

  The set of all identified risks also includes the ones that are presented by any OTS software on which the medical software depends to operate. Each of the prevention/mitigation/handling effort describes the activities, each as pieces of text, to be undertaken to handle the risk. However, these activities are measured in the number of resources needed over a period of time (man-days). These activities tend to be more software (product) specific. Historical data and even the literature could also be significant sources of information.

These attributes represent the elements of the E_B set as per the formal definition of a black-box system (S_B=(E_B,R_B)).

Relationships

In addition to co-attribute relationship and time-stamp relationship as the key two mandatory relationships of the R_B set, the following relationship exists among the aforementioned attributes:

- **Risk severity–effort**; the effort which has to be conducted to prevent, mitigate and/or handle the injury is proportional to the risk severity. In other words, the more severe is the injury the more effort has to be planned to resolve it.

The risk management plan document, a product of concern, has been modelled in terms of attributes which characterize the product. These attributes are accompanied with measurement scales. Furthermore, the relationships among attributes of this product are discussed.

4.1.2 Configuration Management Plan

The development of a configuration management plan is a typical task of the quality planning activity as described in the FDA guidance [FDA 2002]. The products of this process are our concern.

Definitions

The FDA in [FDA 2002] requires the preparation of a Configuration Management (CM) plan. A plan should [FDA] be developed that 'will guide and control multiple parallel development activities and ensure proper communications and documentation. Controls are necessary to ensure positive and correct correspondence among all approved versions of the specification documents, source code, object code, and test suites that comprise a software system. The controls also should ensure accurate identification of, and access to, the currently approved versions' [FDA 2002]. In this chapter also, the notation "[FDA]" is used to emphasize that the auxiliary verb (should) is introduced by the FDA in its guidance, and not by the author of this thesis himself.

Attributes

The following figure describes the "development of configuration management plan" process. Such a process has the "Configuration Items History Descriptions" as its input product. Whereas, it has the "Configuration Management Plan" as its output product. In essence, our goal herein is not to model the process as a transformation of the input product into the output product, but to model the products themselves.

Configuration management is concerned with ensuring the correct labeling of different items, each identified by its version number, and controlling the access to this labeled build (list of items). Thus, versioning control is an essential part on which configuration management depends to ensure the delivery of items with the right version numbers. In this context, the input product is specified as the set of all items under versioning control. Each item (object) is characterized by the following attributes:

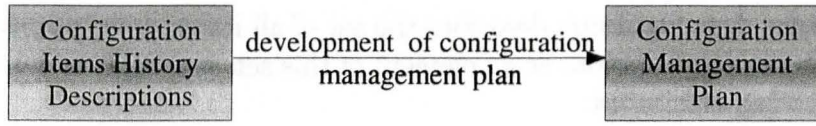


Figure 4.2. Development of configuration management plan

- **Item name**; describes, in a nominal scale, the name which meaningfully identifies the item.
- **Item type**; describes, in a nominal scale, the type to which the item (object) belongs. Examples of values of this attribute, as given in the FDA guidance, are: specification document, source code, object code and test suites.
- **Versions history**; describes, in an ordinal scale, the sequence of versions through which the item (object) passed during its development. In this sense, each version description as an element of this sequence is a sub-product which is characterized by·
  - *Version identifier*; describes the unique number (or any combination of numbers and characters) which identifies a particular version.

    Owner; describes, nominally, the person who performed the change.

    Date, the date & time of the change.
  - *Comments*; description to the reasons for performing the change.

    Informed list, the set of persons, nominally listed, to be informed as a result of a change to this item.

These attributes represent the elements of the E_B set as per the formal definition of a black-box system (S_B=(E_B,R_B))

Relationships

In addition to co-attribute relationship and time-stamp relationship as the key two mandatory relationships of the R_B set, the following relationship exists among the aforementioned attributes:

- **Owner–informed list**, the owner of a version must be in the list of all people whom should be informed whenever a change to that item takes place.

On the other hand, the configuration management plan is the output product of this process. It is characterized by the following attributes:

- **Build Id**, describes, in a nominal scale, the Id which uniquely identifies a particular build (collection of items descriptions) from any other build.

- **List of items descriptions**; describes the set of all items descriptions included in the build. Each item description, as an element of this set, is a tuple which is characterized by the following attributes:
  - *Item name*; describes, in an nominal scale, the name of the item included in the build.
  - *Version description instance*; describes a specific version description of the item. The value of the version identifier attribute identifies which version of the item to be included in the build.
  - *Access list*; describes the list of users grouped on an ordinal scale in accord with the access privilege (read, write, execute) they are granted to the item. In other words, this is a tuple of three sets (readers, writers, executors).

These are the attributes of the E_B set as per the formal definition of a black-box system (S_B=(E_B,R_B)).

Relationships

The co-attribute and time-stamp relationships are the only two relationships among elements of the E_B set.

The configuration management plan document is the output product of this process. The document, as a product of concern, has the build description as its sub-product. Each build description is characterized by a collection of configuration items, where each configuration item has its access control defined. Therefore, this supports the delivery of a product (with a unique build Id) where versions of all of its constituent items, as well as the access level to these items, are defined.

4.1.3 Software Quality Assurance Plan

The preparation of a software quality assurance plan is an activity (process) to be undertaken in the FDA approach. However, this process is complicated enough to be considered as it is. The FDA guidance document describes this process in terms of other subprocesses.

Software Verification and Validation Plan

The preparation of a software verification and validation plan is a subprocess of the preparation of the software quality assurance plan. The FDA in [FDA 2002] requires the identification of: Verification & Validation (V&V) tasks along with acceptance criteria, schedule & resource allocation (for software V&V activities), and reporting requirements.

Definitions

The FDA guidance in [FDA 2002] defines software verification as 'providing objective evidence that the design outputs of a particular phase of the software development lifecycle meet all of the specified requirements for that phase. Software verification looks for consistency, completeness, and correctness of the software and its supporting documentation, as it is being developed, and provides support for a subsequent conclusion that software is validated' [FDA 2002] The FDA guidance considers software validation to be 'confirmation by examination and provision of objective evidence that software specifications conform to user needs and intended uses, and that the particular requirements implemented through software can be consistently fulfilled' [FDA 2002]

Attributes

The following figure describes the "preparation of verification and validation plan" process. The input product for this process takes the form of a document which references all design output products & specification documents as well as the requirements specification document which describes the purpose of the software and user needs. These products which are referenced in the input product are modelled on their own later on in the requirements development and design activities. Thus, "Verification & Validation Plan" as the output product of this process is our concern in this section.

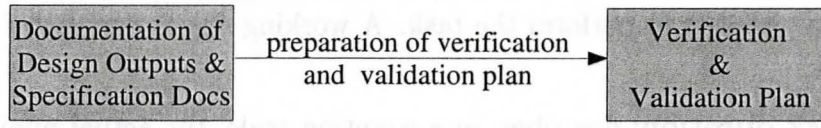


Figure 4.3. Preparation of software verification & validation plan

The input product, with all other products being referenced in it, significantly affects the production of the output product (verification and validation plan) In other words, the verification and validation tasks, personnel/equipment performing these tasks, venues and their schedule significantly depend on these products.

Since the output product (verification and validation plan) takes the form of a document, it has the same document-related attributes which were already described. The definition of validation and verification as given in [FDA 2002] dictate that the verification process is concerned with ensuring that the "design outputs" conform to the "specified requirements for that phase" Whereas, the validation process is concerned with ensuring that "software specification" conforms to "user needs and intended uses" In this sense, the software verification and validation plan (product) is described as a set of all verification/validation tasks. Indeed,

each verification/validation task description is a sub-product of the verification/validation plan. It can be described in terms of the following attributes:

- **Task name**; identifies, in a nominal scale, the task which is to be undertaken as part of the verification/validation process.
- **Personnel descriptions**; the set of all "personnel description" required to perform the validation/verification task. Each personnel description element is a tuple of three valued attributes (areas-of-expertise, level-of-expertise, number). The values of areas-of-expertise and level-of-expertise are given from the following set of all possible values:
  - *Areas of Expertise*; describes the set of all areas, listed nominally, in which the V&V resource may need to be experienced.
  - *Level of Expertise*; describes the set of all levels of expertise, described in an ordinal scale, that any of these areas of expertise (listed above) may require.
- **Equipment descriptions**; the set of all equipments, each named as pieces of text, which are needed to perform the validation/verification task. As previously described, each equipment description is a product in itself, which have to be specified in terms of attributes that characterize it.
- **Time**; describes the point(s) in time at which the verification/validation task should start.
- **Estimated task duration**; describes, in a counting scale, the estimated number of working days needed to perform the task. A working day is equivalent to eight hours of working.
- **Actual task duration**; describes, in a counting scale, the actual number of working days that were actually needed to finish the task.
- **Location**; describes, in a nominal scale, the venue where verification/validation task would take place. Sample values of this attribute are: development environment, staging environment, production environment, etc.
- **Task Complexity**; describes, in an ordinal scale, the complexity (difficulty) level of the verification/validation task to be accomplished.
- **Description of the acceptance criteria**; describes the acceptance criteria, as pieces of text, associated with the validation/verification task.
- **Effort**; describes the amount of effort, described in man-days, needed to perform the V/V task.

These attributes are related with each other in the sense that they, all together, suffice the description of the verification/validation task, the key sub-product of the verification/validation plan. Thus, these attributes represent the elements of the E_B set as per the formal definition of a black-box system (S_B=(E_B,R_B)).

Relationships

In addition to co-attribute relationship and time-stamp relationship as the key two mandatory relationships of the R_B set, the following two relationships exist among the aforementioned attributes:

- **Resources-duration–effort conformance**; effort described in man-days must equal to the number of resources needed to accomplish the task times the task duration.
- **Task difficulty–level of expertise**; the level of expertise of the personnel appointed to perform the validation/verification task, or even to monitor it if the task is automated, is proportional to the task complexity. In other words, a task with the highest level of complexity must not be undertaken by the least experienced member of staff.

The FDA requires the submission of an evidence as a confirmation to the implementation of the verification and validation plan. However, the guidance documents do not describe what evidence is accepted by the FDA. Indeed, these attributes which characterize verification and validation tasks when evaluated and accompanied with real values represent a well-defined structure for a validation/verification report.

Formal Design Review Requirements

The preparation of the requirements for formal design review is another subprocess of the main process (preparation of software quality assurance plan). Therefore, this process results in a document which describes the requirements for formal design review.

Definitions

The FDA guidance document does not provide definitions for this process. By formal review, it just recommends to have an independent third-party reviewer(s).

Attributes

The following figure describes the "preparation of formal design review requirements" process. This process has the "Verification & Validation Plan", which was outputted from the previous process, as its input product. It has the "Formal Design Review Requirements" documentation as its output product. In this context, and since the input product was already modelled in terms of its attributes in the previous process, only the output product is modelled in section.

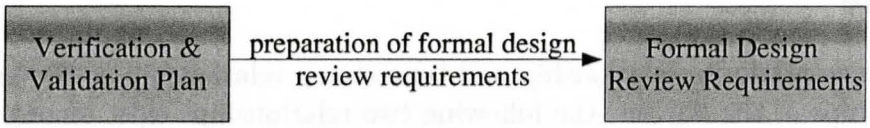


Figure 4.4. Preparation of formal design review requirements

The output product takes the form of a document. Thus, it shares with other documents the same attributes which were already described in chapter three. Furthermore, it adds the following attributes:

- **Goals of review**; describes the objectives of the whole review process.
- **Time**; describes, in a nominal scale, the date and time of the review process.
- **Duration**, describes, in a counting scale, the number of business days needed for the review process.
- **Place**; describes the venue where the formal review takes place.
- **Input product(s)**, sub-product(s) of the formal design review requirements document (product) Each can be described in terms of other attributes like product name and entry properties. Entry properties, a composite attribute as described in [Wiegers 2003], can also be considered as a sub-product which is characterized by attributes of the input product as well as the values for these attributes that are needed as prerequisites to start the review process. This aims to ensure that an input product to the review process is at a certain quality. These attributes are product-specific. However, they should be specified.
- **Independence of review**; describes, in a percentage scale, the relative number of people from outside the software development party to the total number of people participating in the review process.
- **Reviewers description**, a sub-product which can be described as the set of all reviewer-types required. Similarly to the "personnel description", each reviewer-type is described as a tuple of (areas-of-expertise, level-of-expertise and number-of-reviewers)
- **Required exit properties**; a sub-product which can be described in terms of attributes along with some values that are required to indicate the end of the review In more detail, these attributes with particular values are sufficient to base confidence in the quality of the product. Although these attributes may be product-specific, but they should be specified before the review process starts.
- **Actual exit properties**; similarly to required exit properties, however this sub-product is specified in terms of the same attributes of the "required exit properties" product, but with the actual values of these attributes.

These attributes represent the elements of the E_B set as per the formal definition of a black-box system (S_B=(E_B,R_B)).

Relationships

In addition to co-attribute relationship and time-stamp relationship as the key two mandatory relationships of the R_B set, the following relationship exists among the aforementioned attributes:

- **Required exit properties–actual exit properties conformance**; the actual values of the attributes of the exit properties for a particular product under review should be in the acceptance bound of the required values attributes of the exit properties.

The formal design review requirements is a sub-product of the documentation of quality assurance plan. This product has been described in terms of attributes and other sub-products. Each sub-product has been considered as a black-box system which stands on its own. Having described each product in terms of attributes which characterize it along with the measurement scales for these attributes will help software producers to focus on measuring these attributes, which will be subject to evaluation afterwards by the FDA staff.

Other Technical Review Requirements

The preparation of requirements for any technical review is another subprocess of the main process (preparation of software quality assurance plan). Therefore, this process should result in a document which describes the requirements for any technical review.

Definitions

The FDA guidance in [FDA 2002] does not define any other technical review requirements. Indeed, other technical reviews may be organisational or product-specific.

Attributes

The Formal Design Review Requirements product is described in a generic way which can be considered as review requirements of any type of material, not specifically a design or technical product. However, the values of input products, input properties and exit properties differ. Also, the areas-of-expertise may tend to be more technical.

Relationships

The relationships among attributes of this product are similar to the ones of the previous product (formal design review requirements)

4.1.4 Problem Reporting and Resolution Procedures

The preparation of a problem reporting and resolution procedures is a typical task (process) in the FDA approach. As described in the following figure, this process has the "Technical (Development) Team Hierarchy Description" as its input product and the the "Problem Reporting and Resolution Procedures Documentation (PRRPD)" as its output product.

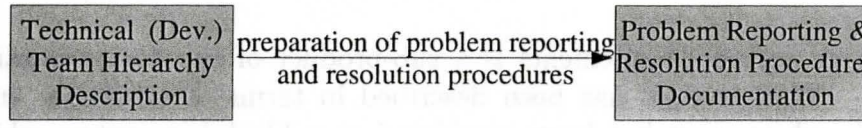


Figure 4.5. Preparation of problem reporting & resolution procedures

Definitions

The FDA guidance requires having problem reporting and resolution procedures during the development of the medical software. According to the FDA guidance in [FDA 2002], procedures should [FDA] be created to describe the way in which anomalies are reported, fixed and approved.

Attributes

The input product is concerned with providing a description to the hierarchy of the technical team. In other words, it describes the main roles in the development team along with the level of responsibility of each role. Such a description is necessary for establishing the reporting and approval mechanisms for discovered anomalies. In more detail, the responsibility hierarchy defines who should report the ticket, to whom it should be assigned, who should verify it after being fixed & tested, and eventually who should close it. Thus, this document describes the set of all roles in the software development environment. Each role can be characterized by the following two attributes:

- **Role name**; describes, in a nominal scale, the name which uniquely identifies a certain role in the software development team.

- **Supervisor role name**; describes, in a nominal scale, the name of the role which is responsible to verify the job of the supervised role.

The aforementioned attributes represent the elements of the E_B set as per the formal definition of a black-box system (S_B=(E_B,R_B)).

This document can be replaced with a chart that clarifies each role in the development team along with its supervisory party. This product may seem trivial since software testers, developers and technical team leads are the usual roles in a software development project. However, the lifecycle of the ticket (software bug) from its issuance till it is resolved may vary depending on the available roles in the software project, especially when one of these roles is missing or even not represented by a separate entity.

Relationships

The co-attribute relationship and time-stamp relationship are the only two relationships of the R_B set. This is the set of all relationships among attributes of the product.

For the output product (problem reporting and resolution procedures documentation), and as it takes the form of a document, it has the document-related attributes. Furthermore, the problem reporting and resolution procedures (procedures for anomaly reporting and resolution) describes the definition of the lifecycle of a ticket (software bug) from the time of issuance till resolving it. Therefore, this product is considered next.

Ticket (software bug) definition

The ticket definition is a sub-product of the problem reporting & resolution procedures documentation. It describes the ticket in terms of its attributes. A good ticket definition should specify the key attributes of the ticket. These are:

- **Ticket Id**; uniquely identifies the ticket.
- **Urgency**; describes, in an ordinal scale, the importance of resolving the ticket.
- **Ticket (bug) status**; describes, in an ordinal scale, the status of the bug. Values can be: active, fixed & tested and verified.
- **Ticket summary**; describes, as pieces of text, the reasons behind reporting the ticket. In other words, it is a description of the problem that the ticket addresses.
  Having defined the values of the "ticket status" attribute, a good ticket definition is sufficient to describe the ticket at any instant in terms of its status, the time at which it is given this status and the persons concerned with it. In this sense, three concrete ticket instances, are given below.

- **Reported ticket**; a sub-product by itself which can be characterized by the following attributes:
  - *Reported by*; the person who reported the ticket.
  - *Time*; the date and time of reporting the ticket.
  - *Assigned to*; the person whom the ticket is assigned to.
  - *Bug status*; takes the value of "Active" herein.
- **Resolved ticket**; a sub-product that can be specified in terms of other attributes. These are:
  - *Resolved by*; the person who fixed and tested the ticket.
  - *Time*; the date and time of fixing the ticket.
  - *Assigned to*; the person who is supposed to verify the ticket.
  - *Bug status*; takes the value of "Fixed & tested" herein.
- **Verified ticket**; a sub-product by itself which can be specified in terms of other attributes. These are:
  - *Verified by*; the person who verified the ticket.
  - *Time*; the date and time of verifying the ticket.
  - *Bug status*; takes the value of "Verified" herein.
- **Duration of ticket closure**; describes, in a counting scale, the number of business days which were needed to resolve the ticket.

These attributes represent the elements of the E_B set as per the formal definition of a black-box system (S_B=(E_B,R_B)).

Relationships

In addition to co-attribute and time-stamp relationships as the key two mandatory relationships of the R_B set, the following relationships exist:

- **Ticket status sequence**; any ticket has to be pass through the defined ticket statuses in sequence. For example, given the ticket statuses as active, fixed & tested and verified, a ticket cannot be assigned the status "verified" unless it was previously assigned the status of "fixed & tested".
- **Verified–resolved by**; the person who verifies a ticket must belong to the supervisor role of the one who fixes and testes that ticket.

- **Reporting–resolving–verifying time**; the time on which the ticket is resolved must be beyond the reporting time of the ticket. Similarly, the time on which the ticket is verified must be beyond the reporting and resolving time of the ticket
- **Duration of ticket closure–urgency**; the duration of a ticket closure is proportional to the urgency of the ticket. Such a duration can be constrained to a certain amount of days as per the ticket urgency.

The problem reporting and resolution procedures documentation, as the product of concern, gives a specification to the ticket definition as its key sub-product. This specification illustrates the key attributes which characterize a good ticket definition. The ticket definition is concerned with providing a sufficient description to the ticket from the moment of issuance till it is resolved. In conclusion, a simplified representation (model) describes the attributes of the problem reporting and resolution procedures documentation, its sub-product (bug definition), relationships among attributes, and their associated measurement scales. Therefore, this will support the preparation of this product by software producers and its evaluation by certification afterwards.

4.1.5 Other Support activities

This section does not explicitly describe any other support activity. Therefore and as no further definitions are given by the FDA, no products are modelled herein. However, modelling can take the same form. The process is identified by its products. These products are described in terms of attributes that characterize them. Relationships among these attributes are described and a measure is given to each attribute.

4.1.6 Quality Planning, the big picture

The typical tasks (subprocesses) of quality planning activity (process) have been described as processes with input and output products. These products were modelled (specified) as black-box systems characterized by attributes whose measures are defined. The relationships among these attributes were pointed out as well. After having described the atomic subprocesses that constitute the quality planning process, an overall description to the quality planning process is important.

The following figure describes the "quality planning" process with the "Quality Planning Requirements Document" as the input product of this process, and the "Quality Planning Documentation" as the output product.

The "Quality Planning Requirements Document" takes the form of a document. Such a document references the input products of the subprocesses of the quality planning process. In other words, the "Quality Planning Requirements Document" references the "System Description", "Configuration Items History Descriptions", "Documentation of Design Outputs

Figure 4.6. Quality Planning activity (process)

& Specification Docs", "Verification & Validation Plan" and "Technical (Dev) Team Hierarchy Description" In the same manner, the "Quality Planning Documentation" references the output products of the subprocesses of the quality planning process. In other words, the "Quality Planning Documentation" references the "Risk Management Plan", "Configuration Management Plan", "Verification & Validation Plan", "Formal Design Review Requirements" and "Problem Reporting & Resolution Procedures Documentation" Such a description for the input and output products of the quality planning activity conforms with the P/p methodology in the sense that the process has only a single input product and produces a single output product.

From a marshalling perspective, the "preparation of a risk management plan", "development of configuration management plan", "preparation of verification and validation plan" and "preparation of problem reporting and resolution procedures", as subprocesses of the quality planning activity, can work in parallel in producing their output products once the "Quality Planning Requirements Document" is received by the "quality planning" activity. However, the "preparation of formal design review requirements" process must wait for the "preparation of verification and validation plan" process to produce the "Verification & Validation Plan" in order to be able to start producing the "Formal Design Review Requirements" product.

In conclusion, we used the Product/process (P/p) method to model the quality planning activity of the FDA validation approach. The P/p modelling approach tends to be systematic and generic. It models any engineering problem as a set of products and processes. These products can then be characterized by attributes whose measurement scale types are defined. In other words, P/p modelling clearly specifies the products of concern, attributes (characteristics of products) to be measured along with their measurement scales. Therefore, such a model (specification) facilitates products evaluation afterwards.

4.2 Requirements

Requirements development is the second activity defined in the FDA approach (software lifecycle model) The FDA guidance [FDA 2002] requires the requirements development process to 'include the identification, analysis, and documentation of information about the device and its intended use' The level of concern of the medical device which was described

in chapter two determines the structure of the requirements documentation. In more detail and as described in [FDA 2005], a summary of the functional requirements will serve as a requirements documentation for a software with a minor level of concern. Whereas a complete Software Requirements Specification (SRS) document is required for a device with a moderate or major level of concern.

The FDA guidance [FDA 2002] describes the "requirements development" activity (process) in terms of other typical tasks (subprocesses). The FDA guidance [FDA 2002] recommends undertaking these typical tasks. In this sense, each subprocess is concerned with the preparation of a product (most likely a document) which represents the output product of this subprocess. These typical tasks (subprocesses) as described in [FDA 2002] are:

- Preliminary Risk Analysis
- Traceability Analysis
  - Software Requirements to System Requirements (and vice versa)
  - Software Requirements to Risk Analysis
- Description of User Characteristics
- Listing of Characteristics and Limitations of Primary and Secondary Memory
- Software User Interface Requirements Analysis
- Software Requirements Evaluation
- Ambiguity Review or Analysis
- System Test Plan Generation
- Acceptance Test Plan Generation

These subprocesses will be considered each in turn.

4.2.1 Preliminary Risk Analysis

The FDA guidance [FDA 2002] recommends conducting a preliminary risk analysis. The guidance document describes it as a way to 'identify the most appropriate measures necessary to prevent harm' [FDA 2002]. However, the guidance document does not add any further detail to what was described in the previous process (preparation of a risk (hazard) management plan). The risk management plan documentation is the result of analyzing potential risks and documenting them. Thus, having the risk management plan describe the analysis of potential software risks, as previously modelled, will serve as the output product of this process. Hence, this product is not remodelled in this section.

4.2.2 Traceability Analysis

The FDA guidance [FDA 2002] recommends conducting a traceability analysis. A software requirements traceability analysis should [FDA] be carried to 'trace software requirements to (and from) system requirements and accordingly to (and from) risk analysis results' [FDA 2002] Generally, traceability analysis aims to ensure that every statement, as given by a potential software user or imposed by any regulation or standard, is translated to a requirement specified in the SRS. In the same manner, it aims to ensure that every software requirement specified in the SRS has an identified origin (source)

Software Requirements to System Requirements (and vice versa)

System requirements focus on describing the whole system of which the software could be part. Therefore, they describe the overall system in terms of its major components where the software is considered one of these components. System requirements tend to be at a higher level of the detailed software requirements.

Definitions

The FDA guidance [FDA 2002] recommends allocating system functions to 'hardware/software, operating conditions, user characteristics, potential hazards, and anticipated tasks' [FDA 2002] Despite its rough description to the traceability analysis process, the guidance neither describes the process (traceability analysis) nor does it describe the product (traceability matrix or traceability documentation)

Attributes

As the FDA guidance document [FDA 2002] does not have an explicit specification to this process or even to its products, we specify this process in terms of its input and output products. In more detail, the following figure describes the "traceability analysis" as the process of preparing a traceability matrix. This process has the "System Description" as its input product and the "Traceability Matrix" as its output product.

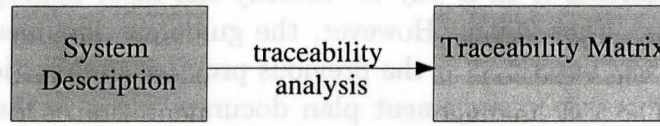


Figure 4.7 Traceability analysis

Since the input product (system description) of this process was modelled in the previous process (quality planning), the output product is our concern. Thus, the product is modelled by providing a simplified description (specification) to it. This specification describes the product in terms of attributes which characterize it, relationships among attributes as well as measurement scales of these attributes. As this product takes the form of a document, it has the document-related attributes which were detailed in chapter three. The traceability documentation (matrix) is described as a set of requirements that are traced to their origin. Thus, the traceability documentation (matrix) as the product of concern has the requirement description as its key sub-product. Any requirement described in the traceability matrix can be characterized by the following attributes:

- **Requirement Id**; a number (or any combination of numbers and letters) which identifies a particular requirement. Such an identifier is used to reference the requirement in the traceability matrix.
- **Requirement type**; describes, in a nominal scale, the category to which a requirement belongs. Examples of requirement types as described in [FDA 2002] are: hardware requirements, software requirements, precision requirements, legal requirements, safety requirements, etc.
- **Requirement origin**; describes, in a nominal scale, the source of the requirement. A system requirements may have a potential user as its source. A low-level software requirement may have a high-level system requirement as its source, especially if a complex requirement is decomposed into more simplified ones for better verifiability.
- **Requirement dependency**; describes, in a nominal scale, the set of all requirements on which this particular requirement depends.

These attributes represent the elements of the E_B set as per the formal definition of a black-box system (S_B=(E_B,R_B)).

Relationships

In addition to co-attribute relationship and time-stamp relationship as the key two mandatory relationships of the R_B set, the following relationships exist among the aforementioned attributes:

- **Requirement Id–Uniqueness**; each requirement must be identified by a unique Id which distinguishes a requirement from any other ones.
- **Requirements dependency–referential integrity**; a requirement on which other requirements depend cannot be simply removed since requirements depend on it.
- **Requirements origin–Required**; each requirement must have a source (origin), i.e., this cell cannot be left out without being filled.

The requirements traceability matrix, product of concern, has the requirement definition as its key sub-product. The requirement definition is modelled in terms of attributes which characterize the product along with their measurement scales. Such a specification forms the basis for evaluating this product.

Software Requirements to Risk Analysis

The FDA guidance requires that software safety requirements should [FDA] be originated from 'a technical risk management process that is closely integrated with the system requirements development process' [FDA 2002]. Furthermore, it requires any software safety requirement specified in the SRS to refer to the risk to which the requirement aims to handle. In this sense and as every risk is identified by a unique Id, every requirement of type "safety" in the traceability matrix will have a particular risk Id as its requirement origin (source). Therefore, the risk management plan document is the input product of this process. Whereas, a traceability matrix is the output product of this process. The requirement definition, a sub-product of the traceability matrix which has been specified in the previous section, is modelled in a generic manner to serve this purpose.

4.2.3 Description of User Characteristics

The preparation of a description of user characteristics is the process concerned with describing the characteristics of the software's potential users. Such a description appears in the form of a document which represents the output product of this process.

Definitions

The process of providing a description of user characteristics was mentioned as it is in the FDA guidance document [FDA 2002]. This typical task was mentioned without any further details about the specification that is sufficient to describe the characteristics of software users.

Attributes

The following figure describes the "preparation of a description of user characteristics" process. This process has the "System Description" as its input product and the "Description of User Characteristics Documentation" as its output product.

Since the input product (system description) of this process was modelled in the previous process (quality planning), the output product is our concern. The following section proposes a list of attributes that characterize software users. A user description is a sub-product of

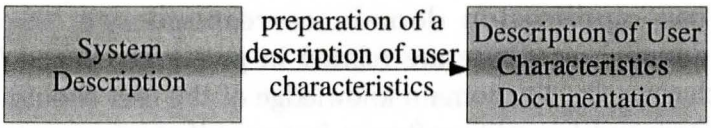


Figure 4.8. Preparation of a user characteristics description

the description of user characteristics documentation. Any user can be described in terms of the following attributes where the values of these attribute participate in grouping software users with common characteristics [Wiegers 2003]

- **User Id**, a unique string (any combination of numbers and letters) which identifies a particular user
- **Frequency of software usage**; describes, in a counting scale, the average number of hours of daily software usage.
- **Features in use**; describes the set of features that a particular user may be using. This set represents a subset of the set of all "software features" specified in the system description which is detailed in the software requirements specification.
- **Application domain**, describes the set of medical knowledge, each described as pieces of text, which may be required as prerequisites to the usage of the medical software.
- **Geographic location**, describes, in a nominal scale, the location of a user with regard to the used software. In other words, the value of this attribute tells whether the user is a local software user or a remote one.
- **Group name**; a unique string (any combination of numbers and letters) which identifies a particular group to which some users belong.

These attributes are sufficient to describe any software user Thus, the "description of user characteristics" document, the product of concern, should describe software users in terms of these attributes. These attributes represent the elements of the E_B set as per the formal definition of a black-box system (S_B=(E_B,R_B))

Relationships

In addition to co-attribute relationship and time-stamp relationship as the key two mandatory relationships of the R_B set, the following relationship exists among the aforementioned attributes:

- **Feature in use–application domain correspondence;** the values given for the "application domain" attribute are related in particular to the "features in use" attribute. In other words, the domain knowledge of the user should sufficiently serve the necessary background for using software features, if any.

The description of user characteristics documentation, product of concern, has the user description as its key sub-product. Therefore, the user description is specified in terms of attributes which characterize it, relationships over these attributes and attributes' measures. Such a specification helps in measuring the values of these attributes by software producers and forms the basis for specifying the auditing (evaluation) procedures by the FDA.

4.2.4 Listing of Characteristics and Limitations of Primary and Secondary Memory

This process was listed as it is in the FDA guidance [FDA 2002] as a typical task in the requirements development activity, without any further details. However, this process can be generalized to describe the assumptions and limitations of all hardware, not memory in specific, upon which the medical software operates. Since these hardware parts are delivered as black boxes by their producers, medical software producers do not usually ask for attributes other than the ones that are given in the specification by hardware producers, but they ask for specific values of these attributes. Indeed, they seek to get the needed (appropriate) hardware as described by its producer. In essence, their job is restricted to referencing the specs of the hardware. Therefore, it is sufficient to describe these hardware parts in terms of attributes as given in the specification by hardware producers. For instance, a primary memory is described in terms of its type or technology (SDRAM, DDR, RDRAM, etc.), its capacity size, etc. In the same manner, a secondary memory is described in terms of disk-size, buffer-size, the number of Revolutions Per Minute (RPM) at which the disk spins, etc.

In conclusion, all hardware parts, including both primary and secondary memory, have to be described in terms of their specification as given by their producers. These specifications are well-defined in the computing literature.

4.2.5 Software User Interface Requirements Analysis

Software user interface requirements analysis is a typical task (subprocess) of the requirements development activity. The FDA guidance document [FDA 2002] recommends the preparation of a software user interface requirements analysis documentation. Such a document describes the way in which the user interacts with the system in terms of its available interfaces.

Definitions

In this context, the FDA guidance requires the specification of 'how users will interact with the system' [FDA 2002] along with the definition of user interfaces.

Attributes

As described in the following figure, "software user interface requirements analysis" has the "System Description" as its input product and the "Software User Interface Specification Document" as its output product.

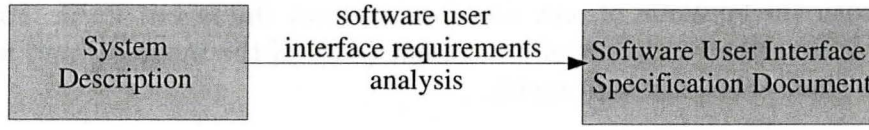


Figure 4.9. Software "user interface requirements" analysis

The software user interface specification, as the output product of this process, takes the form of a document and has all the document-related attributes. Moreover, it adds its own attributes which characterize the product. In essence, the user interface (screen) description as a sub-product of the key product should be described in terms of attributes which characterize it. These attributes are mainly related to the description of the data as key elements of any user interface (screen) Thus, the user interface description is characterized by the following attributes:

- **Screen name**; describes, in a nominal scale, the name of the screen which meaningfully describes its purpose. The name uniquely identifies a particular screen from any other ones.
- **Screen type**; describes, in a nominal scale, the type of screen which the document describes. Values may include filling form, display message or even an interactive message.
- **Data description**, the set of all data elements which appear in the screen. Data elements are tuples of data names and their types. In other words, the data name and type represent the signature of any data element in the set of all screen data elements.
- **Description of functions & reactions including system messages**; the set of all tuples of functions available to a software user, system reaction and system messages per each of these functions. System messages include error messages that are associated with each function. In other words, each element of this set is a tuple (function name, system reaction, system messages) which are all described as pieces of text.

These attributes are sufficient to describe any software user interface. They represent the elements of the E_B set as per the formal definition of a black-box system (S_B=(E_B,R_B)).

Relationships

The co-attribute relationship and time-stamp relationship are the only two relationships that exist among attributes of this product. In other words, the co-attribute and time-stamp relationships are the only two elements of the R_B set.

Having described the attributes of the user interface (screen) description will help evaluating the software user interface specification document. In its main goal, the description is concerned with the contents of user interface and not the layout itself. For instance, an error message is described in terms of its content (data of the message) and not in terms of its appearance (font color, size and style).

For a software with a moderate or major level of concern, a Software Requirements Specification (SRS) document must be prepared as stated in [FDA 2005].

4.2.6 Software Requirements Specification

The software requirements specification document, the product of concern in this section, is modelled in terms of attributes that characterize it.

Definitions

The FDA requires the software requirements specification document to include 'functional, performance, interface, design, development and any other requirements for the software' [FDA 2005]. Furthermore, the FDA guidance requires that the SRS should [FDA] explicitly state the intended use of the software. Moreover, the functionalities which the software system provides should be explicitly specified.

Attributes

The following figure describes the "software requirements specification" process in terms of its input and output products. The "software requirements specification" process has the "System Description" as its input product, whereas it has the "Software Requirements Specification (SRS) Document" as its output product.

In addition to the document-related attributes which every document has, the SRS document has other attributes which characterize it. These are:

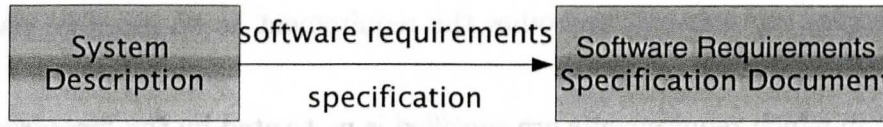


Figure 4.10. Software requirements specification

- **Description of the purpose of the software**; describes the set of all reasons that justify the need for the software.
- **Description of the operating environment**, describes the operating environment in which the medical software is intended to operate.
- **Description of all constraints**; describes the set of all constraints, each described as pieces of text, that restrict the software functionalities.
- **Description of all assumptions**; describes the set of all assumptions, each described as pieces of text, on which the software base.
- **Description of all underlying hardware and software**; describes the set of all hardware and software, each described in terms of its specification as given by its provider (producer), on which the hardware depends to properly operate.
- **Listing of software features**; a composite attribute which is described in terms of in-scope-features and out-of-scope-features. In-scope-features is the set of features that are provided by the software. Whereas, out-of-scope-features is the set of features which are not provided by this software, at a time they may be provided by a rival software. However, they are linked to the medical area to which the software is related. Each feature is itself a product. Indeed, each in-scope-feature can be further characterized by a *set of requirements that are related to it.* In this sense, requirements are instances of the requirement definition which is considered in itself as a sub-product. It can be described in terms of the following attributes:
  - *Requirement Id*, a number (or any combination of numbers and letters) which identifies a particular requirement. Such an identifier is also used to reference the requirement in the traceability matrix.

    Requirement type, describes, in a nominal scale, the category to which a requirement belongs. Examples of requirement types as discussed in [FDA 2002] are: performance requirements, interface requirements, hardware requirements, precision requirements, legal requirements, security requirements, safety requirements, etc.
  - *Requirement specification type*; describes, in a nominal scale, the way in which the requirement is specified. Examples of values of this attribute are: natural language, mathematics, diagrammatical model.

- *Specified requirement*; embodies the requirement as an essential part of a given in-scope feature.

The way in which requirements are specified is motivated by the "requirements by features" approach adopted by the IEEE requirements specification template as detailed in [Wiegers 2003].

The aforementioned attributes represent the elements of the E_B set as per the formal definition of a black-box system (S_B=(E_B,R_B)).

Relationships

In addition to co-attribute relationship and time-stamp relationship as the key two mandatory relationships of the R_B set, the following two relationships exist:

- **System Description Assumptions & Constraints (A&C)–SRS A&C**; the assumptions and constraints specified in the SRS document have to go with the assumptions and constraints already specified in the system description.
- **Requirement Id–Uniqueness**; each requirement must be identified by a unique Id which distinguishes the requirement from any other ones.

In addition to the previous typical tasks (subprocesses) of the requirements development activity (process), the FDA guidance document [FDA 2002] considers "software requirements evaluation" and "ambiguity review or analysis" as two tasks to be undertaken. Since requirements evaluation includes ambiguity review, these two subprocesses will be described together in the following section (software requirements evaluation).

4.2.7 Software Requirements Evaluation

Software requirements evaluation is a typical task (subprocess) of the "requirements development" activity. It also includes ambiguity review or analysis.

Definitions

In this context, the FDA guidance [FDA 2002] states that the Quality System (QS) regulations require a mechanism for addressing incomplete, ambiguous or conflicting requirements, especially as requirements are the inputs of the design activity and, accordingly, the quality of design depends on their quality.

Attributes

An inspection process, as described in [Wiegers 2003], serves the purpose of requirements evaluation and ambiguity review This process (inspection) results in an evaluation report as the output product of this process. The inspection process itself includes participants from different expertise who examine the requirements. The ultimate goal of this process is to end up with requirements that are complete, consistent and unambiguous (verifiable and technically feasible) Therefore, participants in this process include the requirement analyst who specified the requirements, potential software designer, developer and quality engineer (tester) A possible mechanism for addressing completeness, consistency, ambiguity and feasibility is described herein as it is introduced by Karl E. Wiegers in [Wiegers 2003] The requirement analyst ensures requirements consistency by looking for conflicts between system requirements (high-level requirements) and software requirements (low-level requirements) and even between software requirements themselves. In the same sense, a potential software designer tests the completeness of the requirements by assessing whether or not the requirements are sufficient to base the design on them. Similarly, a potential software developer tests the technical feasibility of the documented requirements by questioning their technical applicability. At last, a potential quality engineer is concerned with examining the requirements testability by checking the ability to derive test cases from the specified requirements and thus ensuring their ability to form a basis of testing for the final product (medical software)

The following figure specifies the "software requirements evaluation" process in terms of its input and output products. The process has the "Software Requirements Specification (SRS) Document" as its input product, whereas the "Software Requirements Evaluation Report" represents its output product.

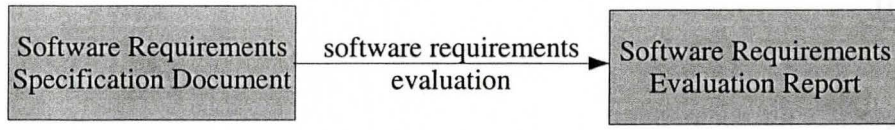


Figure 4.11. Software requirements evaluation

The input product (Software Requirements Specification (SRS) document) has been modelled as per the P/p methodology Thus, the output product is the product of our concern herein in this section. Software requirements evaluation report takes the form of a document which contains the results of the inspection activity. The description of "inspection meeting minutes" is the key sub-product of the software requirements evaluation report. Each instance of the inspection meeting minutes can be characterized by the following attributes:

- **Participants list**, describes, in a nominal scale, the persons participating in the inspection process, each with his/her role.

- **Documented venue**; describes, in a nominal scale, the place where the meeting takes place.
- **Documented time**; describes the date and time of the meeting.
- **Ticket (Requirement defect) definition**; a sub-product of the "inspection meeting minutes" description. It describes a raised requirement-issue (ticket) from the time of its issuance till the time it is resolved. The ticket definition is described in terms of the following attributes (similarly to the way in which the ticket (software bug) definition was specified):
  - *Ticket Id*; uniquely identifies the ticket.
  - *Urgency*; describes, in an ordinal scale, the importance of resolving the ticket.
  - *Ticket status*; describes, in an ordinal scale, the status of the raised requirement-issue. Examples of values for this attribute are: active, fixed & tested and verified.
  - *Ticket summary*; describes, as pieces of text, the reasons for reporting the ticket (problem description).

  Any ticket raised by a member of the meeting participants is assigned to the requirement analysts at the time of that meeting, and will be verified in the next meeting(s) as per its importance. Instances of a requirement-related ticket are similar to those instances of the software defect ticket which were concretely described in the quality planning activity.

These attributes are sufficient to describe the software requirements evaluation report. They represent the elements of the E_B set as per the formal definition of a black-box system (S_B=(E_B,R_B)).

Relationship

In addition to co-attribute relationship and time-stamp relationship as the key two mandatory relationships of the R_B set, the following relationships exist:

- **Ticket–requirement Id**; the Id which is given to the ticket is actually the one which was originally given to the requirement. In other words, the requirement Id perfectly serves as a ticket Id since a ticket is usually issued per a poorly-specified requirement. In the case where more than one ticket have to be issued for the same requirement, the ticket Id can be constituted from the requirement Id followed by a hyphen (–) and followed by a sequence number of the ticket with regard to that particular requirement.
- **Participants–roles**; the roles to which the participants in the inspection process belong must include requirements analyst, developer, software designer and quality assurance engineer. In other words, representative from the requirements, design, development and testing activities must participate in the inspection process.

The requirements evaluation process is described in terms of its input and output products. These products are modelled (specified) in terms of attributes which characterize the products and measures of these attributes. The values of these attributes have to be specified by software producers, especially as they will be subject to evaluation by the FDA afterwards.

4.2.8 System Test Plan Generation & Acceptance Test Plan Generation

The "system test plan generation" and "acceptance test plan generation" are two typical tasks that are recommended to be undertaken in [FDA 2002] The plans generated from these two tasks represent the products of our concern.

Definitions

In this context, no details are given about these two subprocesses or even the properties of the products outputted from them.

Attributes

The following figure specifies the processes of concern in terms of their input and output products.

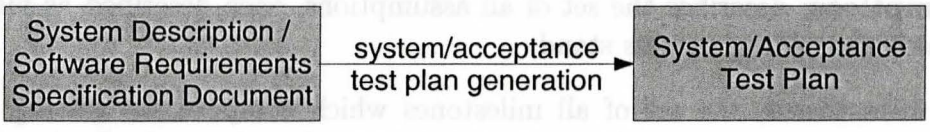


Figure 4.12. System/Acceptance test plan generation

The "system test plan generation" process has the "Software Description" as its input product, whereas it has the "System Test Plan" as its output product. Similarly, the "acceptance test plan generation" process has the "Software Requirements Specification (SRS) Document" as its input, whereas it has the "Acceptance Test Plan" as its output product.

Since the input products (SRS document and system description) have been already modelled, the system test plan and the acceptance test plan are the products of our concern. As the FDA guidance [FDA 2002] does not provide any description to these plans, we propose a general specification for these test plans. Indeed, system test plan and acceptance test plan share the common attributes of any test plan. However, they differ in the methodology (strategy) which is conducted in the testing process. Therefore, we model (specify) them

in terms of attributes which characterize any test plan. These attributes are the common ones that are sufficient to describe both system test plans and acceptance test plans. These attributes as motivated by the test plans of William Perry in [Perry 2000] are:

- **Software System**; describes, in a nominal scale, the name which identities the system (medical software) that is subject to testing.
- **Purpose (goal) of testing**; describes the set of goals, each described as pieces of text, to be achieved by the end of testing.
- **Pretesting background**; describes the set of all prerequisites, each described as pieces of text, that are needed as pretesting background to perform testing.
- **Description of the testing environment**; describes the environment in which testing process will take place.
- **Tools required (if any)**; describes the set of all tools that are required to perform testing. A testing tool is a product in itself. It can be described in terms of the attributes given in its specification by the tool producer.
- **Personnel**; describes the set of all personnel description needed to perform testing. Personnel description has been thoroughly modelled in the quality planning activity. In the same manner, it is described herein as a tuple of field-of-expertise, level-of-expertise, pretesting background and number of resources.
- **Constraints**; describes the set of all constraints, each described as pieces of text, that restrict the testing process.
- **Assumptions**; describes the set of all assumptions, each described as pieces of text, on which the testing process stand.
- **Key milestones**; the set of all milestones which compose the testing process. A milestone description is a sub-product of the test plan. It can be further described in terms of the following attributes:
  - *Milestone name*; describes the name that uniquely identifies a particular milestone.
  - *Projected time*; describes the projected date and time at which this stub (milestone) is anticipated to be reached.
  - *Actual time*; describes the actual date and time at which this sub (milestone) has been reached.
  - *Output products & their acceptance criteria*; describes the set of all products that participate in the decision of passing the milestone *and* their acceptance criteria. Each element in this set is a tuple of the name of an output product, and the acceptance criteria of that named product.

- **Methodology (Strategy) of testing**; describes the strategy which the testing process follow. These strategies are well-defined in the software engineering literature for different types of testing. *In essence, the value of this particular attribute, and accordingly the focus of test cases, differentiates system testing from acceptance testing.*
- **Test cases**; describes the set of all test cases that are to be carried. A test case description is a sub-product of the test plan. It can be further described in terms of the following attributes:
  - *Test case Id*; describes the Id (any combination of characters and numbers) which uniquely identifies the test case.
  - *Test case summary*; a textual description which meaningfully describes the case to be tested.
  - *Input parameters*; the set of all input parameters, each described as a tuple of parameter name, type and value.
  - *Accepted values*; the set of all output parameters, each described as a tuple of parameter name, type and *range of accepted value.*
  - *Actual values*; the set of all output parameters, each described as a tuple of parameter name, type and actual value.
  - *Result*; describes, in a nominal scale, whether the test case passes the test or not with respect to the accepted/actual value. Values of this attribute are either pass or fail.
  - *Justification for disapproval*; a textual description which clarifies the reasons behind not accepting the test case.

These attributes are related with each other in the sense that they, all together, suffice to specify any test plan. Therefore, they represent the elements of the E_B set as per the formal definition of a black-box system (S_B=(E_B,R_B)).

Relationships

In addition to the co-attribute relationship and time-stamp relationship as the key two mandatory relationships of the R_B set, the following relationships exist:

- **Pretesting background–field-of-expertise**; the field-of-expertise of the person who conducts the testing process must cover the pretesting background which is required to perform the test.
- **Test plan Assumptions & Constraints (A&C)–System Desc./SRS A&C**; the assumptions and constraints specified in the test plan have to go with the assumptions and constraints specified in the system description and the SRS document.

- **Test plan, description of the testing environment–System Desc./SRS, description of the operating environment**; the description of testing environment in the test plan must go with the description of the environment in which the software would be running. In other words, testing must take place in the same environment (even if simulated) in which the software would operate.
- **Result–justification for disapproval**; if the result of a test case takes the value of "fail", then the justification for disapproval attribute must be filled. In other words, there should not be any test case with a fail result where the justification is left out.

System and acceptance test plan generation are typical tasks that are recommended to be conducted in the FDA guidance [FDA 2002] These tasks are subprocesses of the requirements development activity (process) They have the test plan as their key output product. System test plan and acceptance test plan share some common attributes which characterize any test plan. However, they differ in the strategies and, accordingly, the test cases to be undertaken during the testing process. Therefore, these two products are modelled similarly In other words, they are modelled in a generic way which is appropriate to specify any test plan regardless of its testing trend (focus)

4.2.9 Requirements, the big picture

The typical tasks (subprocesses) of the requirements development activity (process) have been described as processes with input and output products. These products were modelled (specified) as black-box systems characterized by attributes. The relationships among these attributes were pointed out as well. After having described the atomic subprocesses that constitute the requirements development process, an overall description to the process itself is needed.

The following figure describes the "requirements development" process with the "Software Description" as the input product of this process, and the "Software Requirements Documentation" as the output product.

Figure 4.13. Requirements Development activity (process)

The "Software Description" takes the form of a document. Such a document references the input products of the subprocesses of the requirements development process. In other words, the "Software Description" document references both the "System Description"

and "Software Requirements Specification Document". On the other hand, the "Software Requirements Documentation" references the output products of the subprocesses of the requirements development process. In more detail, the "Software Requirements Documentation" references the "Traceability Matrix", "Description of User Characteristics Documentation", "Software User Interface Specification Document", "Software Requirements Specification Document", "Software Requirements Evaluation Report" and the "System/Acceptance Test Plan". The description of this activity (process) conforms with the P/p methodology in the sense that this process has only a single input product and produces a single output product.

From a marshalling perspective, "traceability analysis", "preparation of a description of user characteristics", "software user interface requirements analysis" and "software requirements specification" as subprocesses of the requirements development activity can work in parallel in producing their output products once the "Software Description" product is received by the "requirements development" activity. However, the "generation of system/acceptance test plan" and "software requirements evaluation" processes have to wait for the "software requirements specification" process to produce the "Software Requirements Specification Document" in order to be able to start producing their own output products.

In conclusion, we used the Product/process (P/p) method to model the requirements development activity (process). The FDA guidance describes this activity in terms of other typical tasks (subprocess). Thus, we modelled (specified) the products of these subprocesses in terms of attributes with defined measurement scale types. The relationships among these attributes are also elaborated. Such a modelling (specification) facilitates the measurements of the products' attributes and, accordingly, the assessment of the products themselves.

4.3 Design

Software design is the third activity defined in the FDA's software lifecycle model. It is the process of 'translating the software requirements specification into a logical and physical implementation of the software to be implemented' [FDA 2002]. In this sense, the FDA guidance describes this activity (process) in terms of other typical tasks (subprocesses). Therefore, the FDA in [FDA 2002] recommends undertaking the following typical tasks:

- Updated Software Risk Analysis
- Traceability Analysis - Design Specification to Software Requirements (and vice versa)
- Software Design Evaluation
- Design Communication Link Analysis
- Module Test Plan Generation
- Integration Test Plan Generation

- Test Design Generation (module, integration, system, and acceptance)

Thus, these processes will be considered each in turn.

4.3.1 Updated Software Risk Analysis

As a typical task (subprocess) of the software design process, "updated software risk analysis" has an updated risk management plan documentation as its main output product. The product (risk management plan) has been already modelled (specified) in terms of attributes which characterize it along with their measurement scales. Attributes of the "Updated Risk Management Plan" and the "Risk Management Plan" are the same. However, the measured values for those attributes may be changed (updated). In more detail and as previously described, risk description, a sub-product of the risk management plan, is characterized by the following attributes: risk Id, category, severity, trigger point and effort. These attributes remain intact as characteristics of risk, but the values of these attributes may be changed. For instance, new risks may be added to the set of all risks or even some attributes' values may be changed as the project proceeds and the risk descriptions becomes more clear. Thus, more accurate values are assigned to these attributes.

In conclusion, this product needs not to be remodelled. Indeed, it needs to be revisited to update the values of the product's attributes.

4.3.2 Design Communication Link Analysis

Design communication link analysis is a typical task that is recommended to be undertaken in [FDA 2002]. Thus, this process is described as given in the FDA guidance [FDA 2002]. Furthermore, it is described in terms of its products.

Definitions

Communication links are defined in the FDA guidance document [FDA 2002] as 'links among internal modules of the software, links with the supporting software, links with the hardware, and links with the user'. The FDA guidance document states that an analysis of communication links should [FDA] be carried to 'evaluate the proposed design with respect to hardware, user, and related software requirements' [FDA 2002].

Attributes

The following figure describes the "design communication link analysis" process with the "Software Description" as the input product of this process, and the "Design Communication Link Documentation" as the output product.

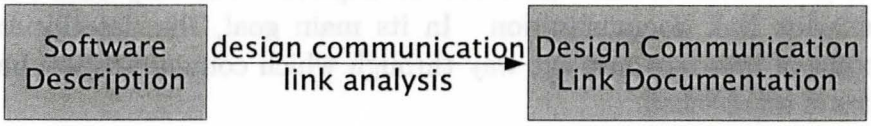


Figure 4.14. Design communication link analysis

The "Software Description" has been already modelled as the input product of the "requirements development" activity Thus, the "Design Communication Link Documentation" is the product of our concern in this section. In essence, this product describes the set of all interfaces between any two objects. Interface description, a sub-product of the main product, is specified in terms of attributes which characterize it. These are:

- **Interface Id**, a number (or any combination of numbers and letters) which uniquely identifies the interface.
- **Interface name**; describes, in a nominal scale, the name which meaningfully describes the purpose of the interface.
- **Objects of communication**, describes, in a nominal scale, the two components, each identified by its component Id, that are involved in the communication.
- **Interface type**; describes, in a nominal scale, the type of object with which the software should interface. Values for this attribute may include hardware devices, supporting software devices or even a software user
- **Data description**, the set of all data elements which are subject to interchange through the interface. Data elements are tuples of data names and their types. In other words, the data name and type represent the signature of any data element.
- **Communication mechanism**, textual description to the approach, implemented or even approved to be used, to establish the communication between the two entities. At the time the requirements development process is only considered with the contents of the interface, the design process describes the content and further focuses on the way in which the communication is established. In other words, the design process has a more concrete and detailed focus in describing such an interface.

These attributes represent the elements of the E_B set as per the formal definition of a black-box system (S_B=(E_B,R_B))

Relationships

For this product, the co-attribute relationship and time-stamp relationship, detailed in chapter three, are the only two relationships in the R_B set.

Describing the attributes of the "interface description" will help in evaluating the design communication link documentation. In its main goal, the description is concerned with the contents of interface and the way through which communication between the two involved parties is established.

4.3.3 Traceability Analysis - Design Specification to Software Specification (and vice versa)

The traceability analysis is a typical task to be undertaken as recommended in [FDA 2002] Therefore, this process is described as given in the FDA guidance document. After that, it is described in terms of its input and output products which are characterized by their attributes.

Definitions

The FDA guidance [FDA 2002] recommends conducting traceability analysis. Such analysis should [FDA] be undertaken to 'verify that the software design implements all of the software requirements' [FDA 2002] (forward traceability) Moreover, the guidance document also considers background traceability as a way 'to verify that all aspects of the design are traceable to software requirements' [FDA 2002]

Attributes

The following figure describes the "traceability analysis (design)" process with the "Software Description" as its input product, and the "Design Traceability Matrix" as its output product.

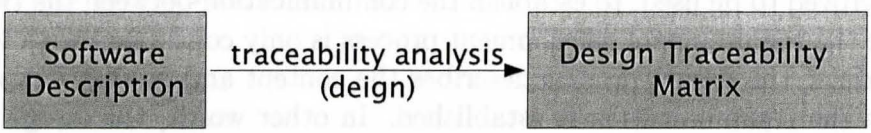


Figure 4.15. Traceability analysis (design)

Software description has already been specified as the input product of the requirements development activity. Hence, it is not remodelled in this section. However, the design traceability matrix is the product of concern as the output product of this process. It has the "traceable element description" as its key sub-product. Similarly to the traceable element specified in the requirements activity, this element is modelled. However, the traceable element in the design activity can be a requirement element or a design element (as traceability takes place between requirements and design). Therefore, this traceable element description as a sub-product of the traceability matrix is characterized by the following attributes:

- **Element Id**; a number (or any combination of numbers and letters) which identifies a particular element. Such an identifier is used to reference the element in the traceability matrix. In more detail, this element will be a requirement Id in the forward traceability matrix. Whereas it will be a design element Id in the backward traceability matrix.
- **Element type**; describes, in a nominal scale, the category to which the element belongs. Examples of element types are: legal component/requirement, safety component/requirement, etc.
- **"Destination (forward)/Source (backward)" Element Id**; describes, in a nominal scale, the source/destination Id of the element. A requirement element has design element(s) as the destination element(s). Whereas the design element has the requirement element(s) as the source element(s). An element may be decomposed into more simplified elements (modules/requirements) for better verifiability.
- **Element dependency**; describes, in a nominal scale, the set of all elements (other requirements or design modules) on which this element depends.

These attributes represent the elements of the E_B set as per the formal definition of a black-box system (S_B=(E_B,R_B)).

Relationships

In addition to co-attribute relationship and time-stamp relationship as the key two mandatory relationships of the R_B set, the following relationships exist among the aforementioned attributes:

- **Element Id−Uniqueness**; each element must be identified by a unique Id which distinguishes it from any other ones.
- **Elements dependency−referential integrity**; an element on which others depend cannot be simply removed since other elements depend on it.
- **Elements source/destination−Required**; each element must have a source (origin) and a destination, i.e., these cells cannot be left out without being filled.

As previously described, attributes of the traceable element sub-product do not change in essence over the attributes of the requirement traceable element. These attributes are the sufficient ones to specify the traceable element description as the key sub-product of traceability analysis documentation (traceability matrix) Having described the traceability matrix by software producer in terms of these attributes which characterize it represents the basis for the product evaluation by the FDA afterwards.

4.3.4 Software Design Evaluation

Software design evaluation is a typical task (subprocess) in the design activity (process) This process is concerned with the evaluation of software design. As described in the following figure, this process has "Software Design Specification (SDS) Document" as its input product and "Software Design Evaluation Report" as its output product.

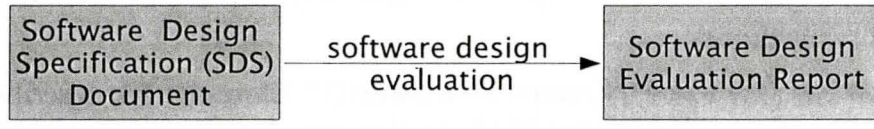


Figure 4.16. Software design evaluation

In this context, software design evaluation report, as the output product of this process, has to be modelled (specified) in terms of attributes and their associated measures. However, software design specification document as the input product of this process has also to be specified. Hence, it is considered first, then the output product is considered.

Definitions

The FDA guidance [FDA 2002] defines the Software Design Specification (SDS) document, as 'a description of what the software should do and how it should do it' The FDA guidance recommends that the developed SDS provides a high-level summary in addition to the detailed design information. This enables people of different level of technical responsibilities understand the SDS. Furthermore, the FDA guidance [FDA 2002] requires the SDS to 'constrain the programmer/coder to stay within the intent of the agreed upon requirements and design, i.e., a complete SDS will relieve the programmer from the need to make ad-hoc design decisions' [FDA 2002]

The FDA guidance does not provide a precise description to the SDS document. However, the FDA guidance [FDA 2002] indicates that the SDS should include the following elements:

- Software requirements specification, including predetermined criteria for acceptance of the software;

- Software risk analysis;
- Development procedures and coding guidelines (or other programming procedures);
- Systems documentation (e.g., a narrative or a context diagram) that describes the systems context in which the program is intended to function, including the relationship of hardware, software, and the physical environment;
- Hardware to be used;
- Parameters to be measured or recorded;
- Logical structure (including control logic) and logical processing steps (e.g., algorithms);
- Data structures and data flow diagrams;
- Definitions of variables (control and data) and description of where they are used;
- Error, alarm, and warning messages;
- Supporting software (e.g., operating systems, drivers, other application software);.
- Communication links (links among internal modules of the software, links with the supporting software, links with the hardware, and links with the user);
- Security measures (both physical and logical security); and
- Any additional constraints not identified in the above elements.

The FDA guidance states that 'if some of the above elements are not included in the software, it may be helpful to future reviewers and maintainers of the software if that is clearly stated' [FDA 2002]. In this context, the FDA guidance document states that the 'first four of the elements noted above usually are separate pre-existing documents that are included by reference in the software design specification' [FDA 2002].

Software requirements specification document, software risk analysis documentation, system documentation (system description) have been already modelled (specified) in the requirement development activity. Other elements listed above, as recommended in the FDA guidance, are included in other products. For instance, hardware to be used, supporting software and other constraints are contained in the "System Description" product. Similarly, communication links are also contained in the "Design Communication Link Documentation". Thus, only "modules specification" and "development procedures & coding guidelines" are considered herein as sub-products of the software design specification document. Therefore, they are modelled in the same manner as any other product.

Development Procedures & Coding Guidelines

The development procedures and coding guidelines is a sub-product of the software design specification document. Thus, it is the product of concern in this section.

Definitions

The FDA guidance document [FDA 2002] recommends the preparation of a development procedures & coding guidelines since the development procedures 'serve as the guide to the organization' [FDA 2002] and the programming procedures 'serve as a guide to individual programmers' [FDA 2002].

Attributes

The development procedures and coding guidelines documentation has the document-related attributes which any document has. Furthermore, the development procedures is characterized by the "description of the development approach" which is approved and followed by the organisation. The development approach (lifecycle model) description can be characterized by the following attributes:

- **Number of phases**; describes, in a counting scale, the number of phases of the software development approach.
- **Description of phases**; describes, in a nominal scale, all phases of the development approach as well as a brief textual description to each phase.
- **Model type**; describes, in a nominal scale, the type of the software development approach. Examples of types of software development approaches are: waterfall, iterative, spiral, literate, etc.

On the other hand, coding guidelines description has a coding convention (standard) as its sub-product. The standard is usually well-known in the literature. For example, if the software implementation has to be in Java, the convention for programming in Java is clearly defined and well-known in the software literature. The convention covers both code style and code documentation. Hence, the coding convention has only to be referenced in this document. However, if there is no defined coding convention for the programming language to be used in the implementation, software producers have to consider developing their own convention to keep the code developed by different programmers consistent. In this sense, software producers could benefit from the conventions defined in the literature to develop their own convention.

Relationships

For this product, co-attribute relationship and time-stamp relationship are the only two relationships in the R_B set.

Modules Specification

Modules specification is a sub-product of the software design specification document. It describes software modules and their internal functionalities. Thus, it is the product of concern in this section.

Definitions

The FDA guidance document describes modules as objects whose specification are 'translated into a programming language' [FDA 2002]. However, no further detail is given to this product.

Attributes

Modules specification has the document-related attributes which any document has. Furthermore, it is characterized by the following attributes:

- **Module Id**; describes the Id (any combination of characters and numbers) which uniquely identifies the module. This Id will be used afterwards to reference the module in traceability matrices .
- **Module name**; describes, in a nominal scale, the name of the module which meaningfully tells about the area to which modules' functions are related.
- **Module description**; a textual description which explains the purpose of the module.
- **Module functionalities**; describes, the set of functions that are required in order to accomplish the purpose described in the module description. Each function description can be considered as a sub-product which is characterized by the following attributes:
  - *Function Id*; describes the Id (any combination of characters and numbers) which uniquely identifies the function.
  - *Function goals*; describes the goals, as pieces of text, which the function aims to achieve.
  - *Function description*; the set of all steps, each described as pieces of text, that describe the way in which function's goals have to be accomplished.
  - *Input parameters*; the set of all input parameters, each described as a tuple of parameter name and type.
  - *Return values*; the set of all parameters outputted from the function, each described as a tuple of parameter name, parameter type and accepted returned value.

These attributes represent the elements of the E_B set as per the formal definition of a black-box system (S_B=(E_B,R_B)).

Relationships

In addition to co-attribute relationship and time-stamp relationship as the key two mandatory relationships of the R_B set, the following property exists:

- **Module/Function Id**; both module and function Id have to be unique. A function Id must start with the Id of the module which contains it followed by a hyphen, followed by its Id. This property ensures a meaningful Id for each function. Such an convention tells about the module in which the function is contained.

After having the input product modelled, the output product (software design evaluation report) is considered next.

Definitions

The FDA guidance [FDA 2002] recommends conducting a software design evaluation. A software design evaluation is carried to 'determine if the design is complete, correct, consistent, unambiguous, feasible, and maintainable' [FDA 2002]. This typical task (subprocess) of the design activity (process) ends up with a "Software Design Evaluation Report" as its main output product. The software design evaluation report consists of a set of the meeting-minutes for each meeting conducted to evaluate the Software Design Specification (SDS) document. Hence, the meeting-minutes report is the key sub-product of the main product.

Attributes

Attributes of meeting-minutes of each software design evaluation session are identical to the attributes of the meeting-minutes of the requirements inspection (review) process. In other words, the meeting minutes report is characterized by the following attributes which were previously detailed: *participants lists, documented venue, documented time, tickets (design defects).* However, the mechanisms to address a design defect differ from the ones to address the requirement one. In more detail, a software developer is the primary participant responsible to tell whether the documented (specified) design, developed by another designer or team, is sufficient to be the source for implementation (coding) or not. In the same manner, the software developer is the one to assure the unambiguity and the technical feasibility for the specified design. The ability of the developer to pursue coding without making any decision by his/her own is the determinant whether the design is complete, unambiguous and technically feasible or not. This mechanism in identifying incompleteness, ambiguity and feasibility results in a high-quality SDS as required in [FDA 2002], where it 'constrains the programmer/coder to stay within the intent of the agreed upon requirements and design and relieves the programmer from the need to make ad hoc design decisions' [FDA 2002]. In the same sense, a software designer, other than the one who delivered the SDS document, works on identifying design conflicts to ensure consistency.

In brief, software design evaluation report is described as a set of evaluation meeting-minutes. Each evaluation meeting-minutes description as a sub-product of the software evaluation report is specified in terms of attributes which characterize it. After that, the mechanisms to assess the software design specification (the input product) and document the findings of this assessment are described. People involved in the evaluation process are representatives of subsequent activities, especially coding and testing. Although reporting tickets (design defects) seems to be subjective since it is performed by humans, any reported ticket must obtain the consensus of the concerned party. This aims to reduce subjectivity in the evaluation of the software design conducted by the software development team.

In addition to all of the above described typical tasks (subprocesses) of the design activity (process), the FDA guidance lists the following typical tasks:

- Test Plan Generation
- Integration Test Plan Generation
- Test Design Generation (module, integration, system and acceptance)

Each of the above typical tasks (subprocesses) of the design activity (process) results in a test plan as the output product of the process. The way in which the test plan was modelled (specified) in the requirements development activity serves the specification of any test plan, regardless of its testing orientation (focus). In other words, the modelled test plan shares the same attributes that also characterize these plans. However, values of "methodology (strategy) of testing" attribute and, accordingly, the defined set of test cases are different.

4.3.5 Software Design, the big picture

The typical tasks (subprocesses) of software design activity (process) have been described as processes with input and output products. These products were modelled (specified) as black-box systems characterized by attributes whose measurement scale types are defined. The relationships among these attributes were pointed out as well. After describing the atomic subprocesses that constitute the design process, we provide an overall description to whole design process.

The following figure describes the "software design" process with the "Software Design Prerequisites Document" as the input product of this process, and the "Software Design Documentation" as the output product.

The "Software Design Prerequisites Document" takes the form of a document. Such a document references the input products of the subprocesses of the software design activity (process). In other words, the "Software Design Prerequisites Document" references the "Software Description" and the "Software Design Specification Document". In the same manner, the "Software Design Documentation" references the output products of the sub-processes of the software design activity (process). In other words, the "Software Design

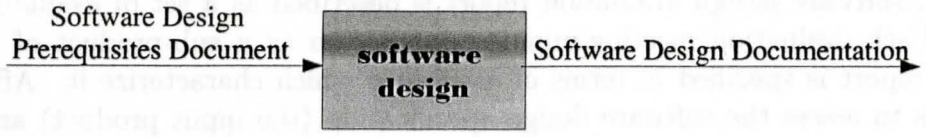


Figure 4.17 Software Design activity (process)

Documentation" references the "Updated Risk Management Plan", "Design Communication Link Documentation", "Design Traceability Matrix" and the "Software Design Evaluation Report" This description of input and output products of the design activity, as of the description of previous activities, conforms with the P/p methodology in the sense that the design process has only a single input product and produces a single output product.

From a marshalling perspective, "design communication link analysis" and "traceability analysis" as subprocesses of the software design activity can work in parallel in producing their output products once "Software Description" is received by these processes. Meanwhile, "updated software risk analysis" can also work in parallel with them once the process receives the "Risk Management Plan" as its input product. However, the "software design evaluation" has to wait for the "design communication link analysis" process to terminate since its output product (Design Communication Link Documentation) is referenced by the "Software Design Specification Document", the input product of this process.

In conclusion to this section, the Product/process (P/p) method is used to model the "Design" activity (process) The FDA guidance describes this activity in terms of other typical tasks (subprocesses) For each typical task, the task is described as given in the FDA guidance [FDA 2002] Furthermore, it is modelled (specified) as a process with input and output products. Each product is then specified in terms of attributes which characterize it. Attributes' measures are given. Relationships among attributes are also annotated. Such a specification clarifies these processes along with their products. Thus, it helps in measuring the products' attributes by software producers and, accordingly, evaluating them afterwards by the FDA staff.

4.4 Construction or Coding

Software construction or coding is the fourth activity defined in the FDA's software lifecycle model. It is the process where 'the detailed design specification is implemented as source code' [FDA 2002] Thus, the FDA guidance describes this activity (process) in terms of other typical tasks (subprocesses) The FDA guidance recommends the following typical tasks to be undertaken[FDA 2002]

- Traceability Analysis

  - Source Code to Design Specification (and vice versa)
  - Test Cases to Source Code and to Design Specification
- Source Code and Source Code Documentation Evaluation
- Source Code Interface Analysis
- Test Procedure and Test Case Generation (module, integration, system, and acceptance)

These typical tasks will be considered each in turn.

4.4.1 Traceability Analysis

Traceability analysis is the first typical task (subprocess) in the construction activity. It is decomposed into two subprocesses which are considered next.

Design Specification to Source Code (and vice versa)

Software code to design specification traceability (and vice versa) is concerned with ensuring that every design element has been implemented into some function in the source code (forward traceability). Furthermore, it is concerned with ensuring that the implemented source code has origins in the design specification (backward traceability).

Definitions

The FDA guidance states that 'a source code traceability analysis is an important tool to verify that all code is linked to established specifications and established test procedures' [FDA 2002]. For the source code, a source code traceability analysis should [FDA] be undertaken to verify that [FDA 2002]:

- Each element of the software design specification has been implemented in code; and
- Modules and functions implemented in code can be traced back to an element in the software design specification and to the risk analysis;

Attributes

The following figure describes the "traceability analysis (Design Specification to Source Code (DS-to-SC))" process with the "DS-to-SC Treatability Prerequisite" as its input product, and the "DS-to-SC Traceability Matrix" as its output product.

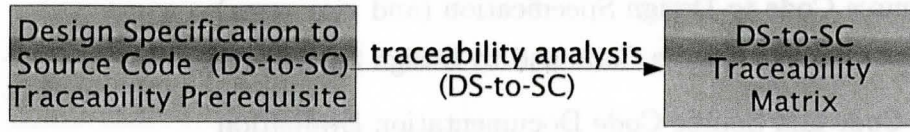


Figure 4.18. Traceability analysis (Design Specification to Source Code (DS-to-SC))

"DS-to-SC Traceability Prerequisite" is the input product of this process. It is a document which references the software design specification document, risk management plan and source code as its sub-products. However, the traceability analysis (design specification to source code (and vice versa)) results in a traceability analysis documentation (DS-to-SC traceability matrix) as the output product of this process.

The traceability analysis documentation (DS-to-SC traceability matrix) is the output product of this process. Similarly to the traceability matrices described in the requirements and design activities, the traceability matrix herein has the "traceable element description" as its key sub-product. In this context, the traceable element is described in terms of the following attributes: *element Id, element type, destination (forward)/Source (backward) element Id and element dependency.*

In the forward traceability, an element Id would be an Id for an element in the design where the destination element Id would be an Id for a module or function in the source code which implements that design element. In the same sense and for the backward traceability, an element Id would be an Id for a module or a function where a source code exists, and the source element Id would be an Id for a design element in the design specification or an Id for a risk in the risk management plan. Therefore and as the product's attributes do not change over the attributes of other traceability matrices, there is no need to remodel this product. In other words, these attributes are the sufficient ones to describe the traceable element description as the key sub-product of traceability analysis documentation (traceability matrix).

Relationships

No further relationships exist among attributes of the traceable element description other than the ones defined in requirements and design activities.

Test Cases to Source Code and to Design Specification

The specification of "test cases to source code and to design specification" is the other subprocess of the traceability analysis process. It is described similarly to the previous subprocess.

Definitions

In this context, a source code traceability analysis should [FDA] be undertaken to verify that [FDA 2002]

- Tests for modules and functions can be traced back to an element in the software design specification and to the risk analysis; and
- Tests for modules and functions can be traced to source code for the same modules and functions.

Attributes

The following figure describes the "traceability analysis (Test Cases to Source Code & Design Specification (TC-to-SC&DS))" process with the "TC-to-SC&DS Treatability Prerequisite" as its input product, and the "TC-to-SC&DS Traceability Matrix" as its output product.

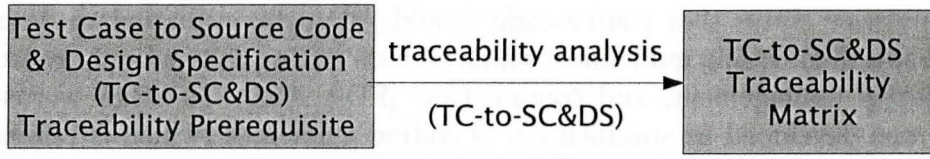


Figure 4.19. Traceability analysis (Test Cases to Source Code & Design Specification (TC-to-SC&DS))

"TC-to-SC&DS Traceability Prerequisite" is the input product of this process. It references the acceptance test plan, source code and software design specification document. However, the traceability analysis (test cases to source code and design specification) results in a traceability analysis documentation (TC-to-SC&DS traceability matrix) as the output product of this process.

As described in the previous process and similarly to the traceability matrices described in requirements, design and coding activities, the traceability matrix herein has the "traceable element description" as its key sub-product. The traceable element description is defined in a generic way to serve as a requirement element, design element, source code element or even a test case element. It is described in terms of the following attributes: *element Id, element type, "destination (forward)/Source (backward)" element Id and element dependency.* These attributes are sufficient to describe any traceable element as the key sub-product of traceability analysis documentation (traceability matrix)

Relationships

Herein also, no further relationships exit among attributes of the traceable element description other than the ones defined in the requirements and design activities.

4.4.2 Source Code and Source Code Documentation Evaluation

The evaluation of source code and source code documentation is the second typical task (subprocess) of the construction activity (process) which is recommended to be undertaken in [FDA 2002] In the following two sections, the process is described as given in [FDA 2002] Then, it is described in terms of input and output products, with each of these products modelled in terms of attributes that characterize it.

Definitions

The FDA guidance states that source code should [FDA] be evaluated to 'verify its compliance with specified coding guidelines, which include coding conventions regarding clarity, style, complexity management, and commenting' [FDA 2002] In other words, the document which was developed as specification of coding guidelines (which is referenced in the software design specification document) is the reference for this evaluation. The FDA guidance requires code comments to provide 'useful and descriptive information for a module, including expected inputs and outputs, variables referenced, expected data types, and operations to be performed' [FDA 2002] Moreover, the guidance requires that source code should [FDA] also 'be evaluated to verify its compliance with the corresponding detailed design specification' [FDA 2002]

Attributes

The following figure describes the "source code and source code documentation evaluation" process. This process has the "Source Code-to-Design Compliance Material" as its input product and the "Source Code (SC)/ SC Documentation Evaluation Report" as its output product.

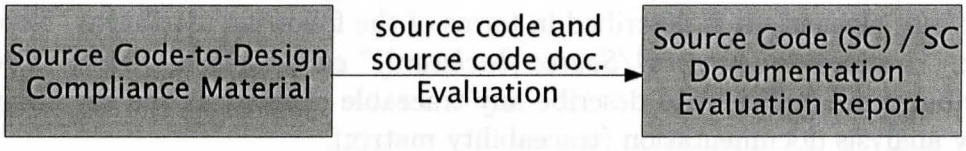


Figure 4.20. Source code and source code documentation evaluation

The "source code and source code documentation evaluation" process is concerned with evaluating the source code as well as its documentation. The input product of this process (Source Code-to-Design Compliance Material) references both source code and software design specification document to which the code is required to conform. On the other hand, the process produces the "Source Code (SC) / SC Documentation Evaluation Report" which is used to determine the conformance of the source code to the coding guidelines specification document. The evaluation report, the output product of this process, assures the conformance by describing the sequence of resolving the "source code/source code documentation" defects that were discovered per each module throughout a peer-review process.

In this context, a ticket (source code/source code documentation bug) definition as a sub-product of the evaluation report describes the the lifecycle of the ticket. The definition of the "source code/source code documentation" defect is similar to any defect definition previously described. It is described in terms of the following attributes: *ticket Id, violation type, urgency, ticket summary, reported by, time, assigned to and ticket (bug) status.* For this product, the violation type attribute describes the type of violation discovered. Examples of values for this attributes are: violation to coding conventions regarding style, violation to coding conventions regarding clarity, violation to coding conventions regarding complexity management and violation to commenting. The attributes that are used to model (specify) any defect (bug) definition are similar. In other words, these attributes, as specified in previous sections along with their measurement scales and relationships among them are sufficient to model the defect definition, the key sub-product of the source code/source code documentation evaluation report.

Relationships

No further relationships over attributes of the source code/source code documentation defect definition, other than the ones defined for defect definition in the requirements and design activities, exist.

4.4.3 Source Code Interface Analysis

The analysis of source code interface is a typical task (subprocess) of the construction activity (process) which is recommended to be undertaken by the FDA. Thus, this process is described as given in [FDA 2002]. After that, the process is described as it receives an input product and produces an output product. Input and output products are modelled (specified) in terms of attributes which characterize them, relationships among attributes and attributes measures.

Definitions

In this context, the FDA guidance states that 'source code evaluations should be extended to verification of internal linkages between modules and layers (horizontal and vertical interfaces), and compliance with their design specifications' [FDA 2002]

Attributes

The source code interface analysis is the process that is concerned with confirming the conformance of source code modules to their specification as detailed in the software design specification document.

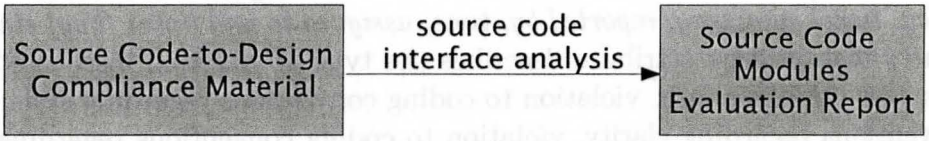


Figure 4.21. Source code interface analysis

As described in the above figure, this process has the "Source Code-to-Design Compliance Material" as its input product. This product references both software source code and software design specification document. As an output product, the process has the "Source Code Modules Evaluation Report"

The source code interface analysis appears as a peer-review for the software source code. The review aims to ensure the conformance between modules as documented in the modules description section of the software design specification document and their actual implementation in the software source code. Similarly, and as in the previous process, any mismatch between the specification and the implementation must be reported as a defect (ticket) and documented in the source code modules evaluation report. Resolving all the tickets, reported by independent reviewers, in the source code modules evaluation report is the proof of the conformance between the specified software modules' interfaces and their implementation. The ticket definition as the key sub-product of the source code modules evaluation report has the same attributes as the ticket definition described in the previous processes. Hence, it is not remodelled in this section.

Relationships

Herein also, no further relationships exist over attributes of the defect definition. However, defects in this context are reported as a result of a mismatch between modules' interfaces as specified in the modules description of the software design specification document, and modules' interfaces as implemented in the software source code.

4.4.4 Test Procedure and Test Case Generation (module, integration, system, and acceptance)

As previously described, a test plan is the output product of any of these activities. The test plans outputted from these processes share the same attributes. However, each has its own values for the "methodology (strategy) of testing" attribute which describes the procedures used in the testing process. Accordingly, the set of all test cases are different from one test plan to another

4.4.5 Construction or Coding, the big picture

The typical tasks (subprocesses) of the construction or coding activity (process) have been described as processes with input and output products. These products were modelled (specified) as black-box systems characterized by attributes whose measures are defined. The relationships over these attributes were specified as well. After describing the atomic subprocesses that constitute the coding process, we provide an overall description to the validation of the construction (coding) activity.

The following figure describes the validation of the "construction or coding" process with the "Source Code Assessment Material" as the input product of this process, and the "Source Code Assessment Report" as the output product.

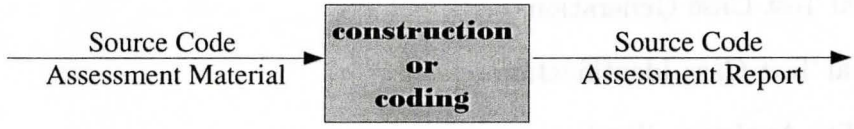


Figure 4.22. Construction or Coding activity (process)

The "Source Code Assessment Material" references the input products of subprocesses of the coding process. In more detail, the "Source Code Assessment Material" references "DS-to-SC Traceability Prerequisite", "TC-to-SC&DS Traceability Prerequisite" and the "Source Code-to-Design Compliance Material" Similarly, the "Source Code Assessment Report" references the output products of the subprocesses of the coding process. In other words, the "Source Code Assessment Report" references the "DS-to-SC Traceability Matrix", "TC-to-SC&DS Traceability Matrix", "Source Code (SC) / SC Documentation Evaluation Report" and the "Source Code Modules Evaluation Report" Such a description of input and output products of the validation tasks for the coding activity, similarly to the description of previous activities, conforms with the P/p methodology in the sense that the process has only a single input and produces a single output product.

From a marshalling perspective, all of the subprocesses of the coding process can work in parallel in producing their output products once their inputs are received. This property (parallelism) is achieved since no input product to any of the subprocesses of the coding process is outputted from a previous subprocess.

In conclusion, and similarly to any other activity (process) defined in [FDA 2002], the Product/process (P/p) method is used to model the "Construction or Coding" activity (process). The FDA guidance describes this activity in terms of typical tasks (subprocesses). For each typical task, we described it as given in the FDA guidance [FDA 2002]. Furthermore, we modelled (specified) each input and output product of these tasks (subprocesses) in terms of attributes which characterize it. This modelling (specification) explicitly describes these processes along with their products. Thus, it forms a basis for establishing an evaluation mechanism for these products as well as their attributes by the FDA.

4.5 Testing by the Software Developer

Testing by the software developer is an activity that is recommended to be undertaken in [FDA 2002]. Similarly to other activities (processes), this process is described in terms of other typical tasks (subprocesses). These are [FDA 2002]:

- Test Planning
- Structural Test Case Generation
- Functional Test Case Identification
- Traceability Analysis - Testing
  - Unit (Module) Tests to Detailed Design
  - Integration Tests to High Level Design
  - System Tests to Software Requirements
- Unit (Module) Test Execution
- Integration Test Execution
- Functional Test Execution
- System Test Execution
- Acceptance Test Execution
- Test Results Evaluation
- Error Evaluation/Resolution

- Final Test Report

Thus, these subprocesses will be considered each in turn in this section.

4.5.1 Test Planning

Test planning is the first typical task (subprocess) of the testing activity (process) performed by software developer. It is concerned with the preparation for testing. It aims to have the testing process more systematic, effective and organized. Thus, test planning is discussed as given in [FDA 2002]. After that, it is considered as per the modelling structure described in chapter three.

Definitions

The FDA guidance [FDA 2002] recommends that planning for the testing process starts early in order to have it efficient and effective. In more detail, the FDA guidance [FDA 2002] recommends that test plans should [FDA] be developed 'as early in the software development process as feasible. They should identify the schedules, environments, resources (personnel, tools, etc.), methodologies, cases (inputs, procedures, outputs, expected results), documentation, and reporting criteria' [FDA 2002]. Moreover, test plans should [FDA] identify the 'particular tasks to be conducted at each stage of development and include justification of the level of effort represented by their corresponding completion criteria' [FDA 2002]. In this context, the guidance states that the real effort of testing 'lies in the definition of what is to be tested rather than in the performance of the test' [FDA 2002]. Therefore, the guidance states that the amount of testing of the testing process 'can be linked to complexity, criticality, reliability, and/or safety issues' [FDA 2002].

For test cases, the guidance highlights the importance of specifying the expected result as an essential element of the software test case. Furthermore, it considers the expected result, when prepared before actual testing starts, as the key detail that permits objective evaluation of the actual test result.

Attributes

The following figure describes test planning process. Such a process has the "Software Requirements Specification Document" as its input product, and it has the "Test Plan" as its output product.

Since the software requirements specification document has been already modelled, the output product is our main concern. The test plan has to be modelled (specified) in terms of key attributes that characterize the product (test plan). Furthermore, measurement scales

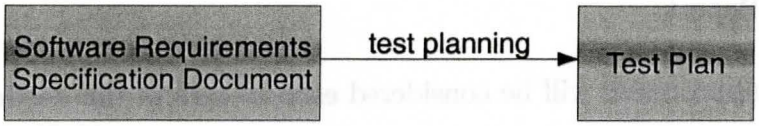


Figure 4.23. Test planning

for these attributes have to be defined. Relationships among attributes have also to be annotated.

In this sense, the test plan as detailed in the requirements development activity was modelled (specified) in a generic way that perfectly fits this purpose. It was described in terms of the following attributes: *software system, purpose (goal) of testing, pretesting background, testing environment, tools required (if any), personnel, constraints, assumptions, key milestones, methodology (strategy) of testing and the set of all test cases.* Adding "justification for the level of testing" to the aforementioned list of attributes as a new attribute enhances the test plan to better serve the FDA requirements. By now, these attributes are the common ones that characterize any test plan, regardless of its focus (field-orientation) The measurement scales of these attributes are previously defined and they do not change from a plan to another However, the values assigned to the "methodology (strategy) of testing" attribute which describe the procedures & strategies conducted in testing, may change from one test plan to another Consequently, the set of all test cases differ from one plan to another, depending on the testing focus. These attributes are related with each other in the sense that they, all together, suffice to specify any test plan. Therefore, they represent the elements of the E_B set as per the formal definition of a black-box system (S_B=(E_B,R_B))

Relationships

Similarly to the relationships among attributes of the test plan as defined in the quality planning section, the co-attribute relationship, time-stamp relationship, pretesting background−field-of-expertise relationship, test plan Assumptions & Constraints (A&C)−SRS A&C relationship and result−justification for disapproval relationship are the relationships among attributes. Since these relationships are previously specified, they are not described in this section.

4.5.2 Structural Test Case Generation

The generation of structural test cases is a typical task (subprocess) which is recommended to be undertaken in [FDA 2002], as part of the testing activity (process) performed by the software developer Structural test case generation is concerned with the preparation of

structural test cases. In this sense, structural test cases are the elements of the set of test cases which is part of the test plan.

Definitions

By structural testing, the FDA guidance [FDA 2002] means "white-box testing" or "code-based testing". In this context, the FDA guidance requires structural testing to 'identify test cases based on the knowledge obtained from source code, detailed design specification, and other development documents' [FDA 2002]. Test cases are developed (prepared) to challenge the control decision of the program and the data structures used in it. According to the FDA guidance, structural testing is accomplished at the unit testing level, but can also be extended to other levels of testing.

The percentage of the software structure which has been tested during structural testing is called "coverage" as in [FDA 2002]. The level of coverage is usually specified in the form of a matrix. These matrices are called coverage matrices. In this context, the guidance states that the 'use of the term "coverage" usually means 100% coverage. For example, if a testing program achieved "statement coverage", it means that 100% of the statements in the software has been executed at least once' [FDA 2002]. Therefore, the guidance describes the common structural coverage metrics as [FDA 2002]:

- Statement Coverage
- Decision (Branch) Coverage
- Condition Coverage
- Multi-Condition Coverage
- Loop Coverage
- Path Coverage, and
- Data Flow Coverage

Attributes

The following figure describes the structural test case identification process. Such a process is concerned with identifying the code-based test cases. These are the test cases that are derived from deep understanding of the detailed specification and implementation.

The process has the "Software Specification/Implementation" as its input product. This product references both software source code and software design specification document. On the other hand, this process has "Structural Test Cases" as its output product. As previously described, test cases (including structural ones) are characterized by test case

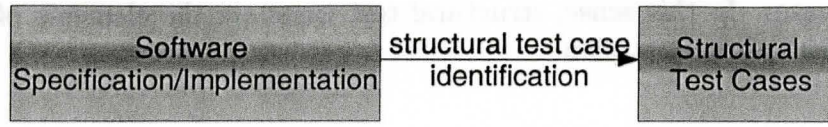


Figure 4.24. Structural test case identification

description, a sub-product of the test plan. Test case description is modelled in terms of the following attributes: *test case Id, test case summary, input parameters, accepted values and actual values.* The values for these attributes in addition to the values for the "pretesting background" and "methodology (strategy) of testing" as attributes of the test plan are sufficient to characterize structural test cases. Indeed, they distinguish structural test cases from other types of test cases. In this context and since structural test cases depend on the knowledge obtained from low-level understanding of the software and/or software design specification document, knowledge about these products may be annotated as pretesting background required for identifying test cases, and after that performing them. The aforementioned attributes represent the elements of the E_B set as per the formal definition of a black-box system (S_B=(E_B,R_B))

Relationships

As elements of the set of all relationships (R_B) among elements of the set of all attributes (E_B) as per the definition (S_B=(E_B,R_B)), the co-attribute relationship, time-stamp relationship and "result−justification for disapproval" relationship are the ones among attributes of this product.

4.5.3 Functional Test Case Identification

The generation of functional test cases is a typical task (subprocess) which is recommended to be undertaken in [FDA 2002], as part of the testing activity (process) performed by the software developer In the same manner, this process is concerned with the preparation of test cases, yet functional ones. These test cases are derived from the knowledge of the functionalities which the software product provides. In this case, functional test cases are the elements of the set of test cases, as part of the test plan, which focuses on functional-based testing.

Definitions

By functional testing, the FDA guidance [FDA 2002] means "definition-based/specification-based testing" or "black-box testing" In this context, the FDA guidance requires functional

testing to 'identify test cases based on the definition of what the software product (whether it be a unit (module) or a complete program) is intended to do' [FDA 2002] Test cases are developed (prepared) to challenge the intended use or functionality of the software along with its interfaces (both internal and external) According to the FDA guidance, functional testing can be applied at all levels of software testing from unit to system testing. Thus, the FDA guidance [FDA 2002] describes the following different types of functional testing (with increasing levels of effort)

- Normal case; concerned with testing usual inputs.
- Output forcing; concerned with ensuring that all software outputs have been generated.
- Robustness; concerned with demonstrating that the software behaves correctly even for invalid or unexpected inputs. In this context, the guidance recommends using some of the methods which are known for identifying a sufficient set of such test cases. Examples of these methods, as mentioned in the guidance, are: equivalence class partitioning, boundary value analysis and special case identification (error guessing) However, the guidance explicitly states that even the usage of these techniques does not guarantee that all the challenges to the software have been identified.
- Combinations of inputs; concerned with applying combinations of inputs which a software may encounter while it is running. In this context, the guidance recommends cause-effect graphing as 'one functional software testing technique that systematically identifies combinations of inputs to a software product for inclusion in test cases' [FDA 2002]

Attributes

The following figure describes functional test case identification process. Such a process is concerned with identifying black-box test cases. Black-box test cases are the ones that are derived from understanding the functionalities which the software is required to provide. Such functionalities embody the functions of the software modules as specified in the software design specification document.

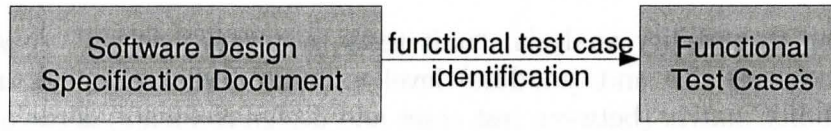


Figure 4.25. Functional test case identification

The process has the "Software Design Specification Document" as its input product. Whereas it has the "Functional Test Cases" as its output product. Likewise structural test

cases, functional test cases are characterized by the test case description, a sub-product of the test plan.

As in the previous process (structural test case identification), test case description is characterized by the following attributes: *test case Id, test case summary, input parameters, accepted values and actual values.* However, values given to the "methodology (strategy) of testing" attribute of the test plan as well as the focus of test cases distinguish functional test cases from structural test cases.

Relationships

For the relationships among attributes, no more relationships exist, other than the ones previously described over the attributes of the test case description.

4.5.4 Traceability Analysis - Testing

Traceability analysis at the testing level is another typical task (subprocess) of the testing activity (process) performed by the software developer. This task, as recommended to be undertaken in [FDA 2002], is concerned with ensuring that all test cases generated at the unit, integration and system levels have origins in the detailed design, high-level design and requirements, respectively. This task (subprocess) is further decomposed in [FDA 2002] into other traceability analysis activities but at different levels. These are:

- Unit (Module) Tests to Detailed Design
- Integration Tests to High Level Design
- System Tests to Software Requirements

Thus, these treatabilities at their different levels are considered next.

Unit (Module) Tests to Detailed Design

As part of testing traceability analysis, "unit tests to detailed design" is concerned with tracing all test cases at the unit (module) level to some design elements in the detailed design. A traceability matrix (between test cases and design elements) is the output product of this process.

Definitions

The FDA guidance [FDA 2002] describes unit testing (module or component testing) as an important process to ensure that 'functionality not visible at the system level is exam-

ined by testing' [FDA 2002] Moreover, the FDA guidance states that unit testing should ensure that 'quality software units are furnished for integration into the finished software product' [FDA 2002]

Attributes

The following figure describes "unit tests to detailed design traceability analysis" process in terms of its input and output products. This process has the "Software Design Tests Document" as its input product. This product references both the "Software Design Specification Document" and the "Unit Test Plan" Unit test plan is a test plan with all the attributes specified for a test plan. However, it contains unit test cases. On the other hand, the process has the "Unit Tests to Detailed Design Traceability Matrix" as its output product.

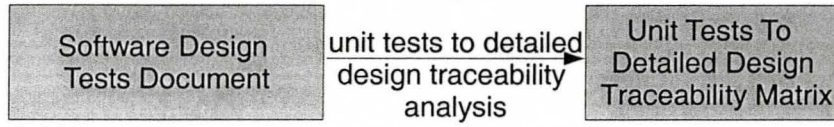


Figure 4.26. Unit tests to detailed design traceability analysis

In essence, the process is concerned with tracing unit test cases back to their origin in the detailed design (modules of the software design specification document) This product (traceability matrix) has the "traceable element description" as its key sub-product. The traceable element has been modelled in terms of the following attributes: *element Id, element type, source (backward) element Id and element dependency.* In this context, each traceable element will have the test case Id as the "element Id" mapped to the design element Id (module Id) as the "source element Id" These attributes are the sufficient ones that characterize the traceable element description, the key sub-product of the traceability matrix. Further, they represent the elements of the E_B set as per the formal definition of a black-box system (S_B=(E_B,R_B))

Relationships

No further relationships exist among attributes of the traceable element description, other than the ones defined in the requirements and design activities. In more detail, the co-attribute relationship, time-stamp relationship, element Id−uniqueness relationship, elements dependency−referential integrity relationship and elements source−required relationship are the only relationships among attributes of this product. In other words, these relationships are the only elements in the set of all relationships (R_B) as per the formal definition of a black-box system (S_B=(E_B,R_B))

Integration Tests to High Level Design

As another part of testing traceability analysis, "integration tests to high level design" is concerned with tracing all test cases at the integration level to some design elements in the high level design (design communication link documentation)

Definitions

The FDA guidance [FDA 2002] considers integration testing as an important testing process which 'focuses on the transfer of data and control across a program's internal and external interfaces' [FDA 2002] In this context, the guidance states that external interfaces are 'those with other software (including operating system software), system hardware, and the users can be described as communications links' [FDA 2002]

Attributes

As described in the following figure, the "integration tests to high level design traceability analysis" process has the "Design Communication Tests Document" as its input product. This document references both "Design Communication Link Documentation" and "Integration Test Plan" The integration test plan is characterized by the same attributes of the test plan. However, it contains integration test cases. Conversely, this process has the "Integration Tests to High Level Design Traceability Matrix" as its output product.

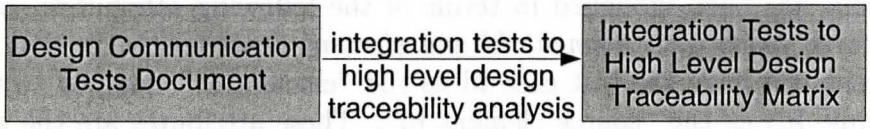


Figure 4.27 Integration tests to high level design traceability analysis

Integration tests to high level design traceability matrix traces integration test cases to high level design elements. This product (traceability matrix) has the "traceable element description" as its key sub-product. Attributes which are used in traceability matrices to model the traceable element are generic, yet sufficient to describe the traceable element herein also. However and in this context, each traceable element will have the integration test case Id as the value of "element Id" attribute, which is mapped to the high level design element Id (the Id of the interface through which the two components identified in the integration test case communicate) as the "source element Id"

Relationships

No more relationships among attributes of the traceable element description, other than the ones previously defined, exist.

System Tests to Software Requirements

As the last part of testing traceability analysis, "system tests to software requirements" is concerned with tracing all test cases at the system level to requirements which led to these test cases. Similarly to the previous two processes, a traceability matrix (between test cases at the system level and requirements) is the output product of this process.

Definitions

In the same manner, the FDA guidance [FDA 2002] considers system testing as an important process which 'demonstrates that all specified functionality exists and that the software product is trustworthy' [FDA 2002]. The FDA guidance [FDA 2002] requires this testing process to verify the functionalities implemented in the software, along with the non-functional behaviour of the software (like performance) on the production environment and as per the specified requirements. However, the guidance document considers that applying this kind of testing depends on the capability of the software developer/tester to produce the target operating environment. Thus, the guidance proposes simulation as an appropriate alternative to be utilized where the target environment is difficult to be provided. In this context, the guidance stresses on the importance of obtaining an Investigational Device Exemption (IDE) or Institutional Review Board (IRB) approval from the FDA, when the medical software is to be used on humans especially if an FDA clearance has not been already granted.

Attributes

As described in the following figure, the "system tests to software requirements traceability analysis" process has the "System Tests Document" as its input product. This product references both "Software Requirements Specification Document" and "System Test Plan". System test plan is characterized by the same attributes of any test plan. However, it contains system test cases. The strategies of testing also differ from unit to integration test plans. On the other hand, this process has the "System Tests to Software Requirements Traceability Matrix" as its output product.

Similarly to the previous two processes, a traceability matrix (system tests to software requirements traceability matrix in this case) is the output product of this process. This matrix traces system test cases to their origin in software requirements. This product (traceability matrix) has the "traceable element description" as its key sub-product. The attributes which were used to model this traceable element are sufficient to describe the traceable element

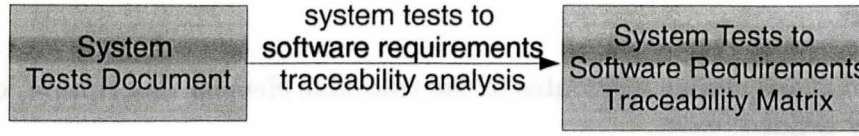


Figure 4.28. System tests to software requirements traceability analysis

herein also. However and in this context, each traceable element has the system test case Id as the "element Id" mapped to the requirement Id as "source element Id"

Relationships

Herein also, no further relationships exist among attributes of the traceable element description, other than the ones previously defined.

4.5.5 Unit (Module), Integration and Functional Test Execution

"{Unit (Module), integration and functional} test execution" are all concerned with executing the corresponding {unit, integration and functional} test cases as specified in their test plans. In other words, each of these tasks is concerned with executing the test plans with the set of all test cases, but on the developer's control environment. The output product of all these typical tasks (subprocesses) is the test plan itself with all actual results assigned to the "actual values" attribute of test cases. Again, the typical tasks (subprocesses) as mentioned in [FDA 2002] are:

- Unit (Module) Test Execution
- Integration Test Execution
- Functional Test Execution
- System Test Execution

With respect to software validation, these processes are concerned with assigning values to attributes of their products after performing testing as per the test plans. These products (test plans) are characterized by the same attributes (though the values assigned to these attributes are different) Since no further definitions are given by the FDA in [FDA 2002] with respect to the execution of these tests, these processes are not considered in terms of definitions, attributes and relationships sections, similarly to other processes. Nevertheless, some of the outstanding issues are pointed out.

At the system level, as per the FDA guidance, any software which is used to automate the medical device process, or any part of the software quality system, or even any partner

application on which the medical software depends to operate is subject to validation. However, the validation approach varies from one application to another. The medical software producers has the freedom of defining the validation approach to be used in order to validate the partner application. However, the producer holds the responsibility of any damage which may happen as a result of the usage of these applications. In this sense, the FDA guidance document clarifies the general activities by which validation is generally supported. These are [FDA 2002]:

- verifications of the outputs from each stage of that software development life cycle; and
- checking for proper operation of the finished software in the device manufacturer's intended use environment.

Any chosen validation approach should support these activities.

The same rules apply on commercial software that may be used as part of the quality system such as databases. In this way, only those functionalities, among all the ones which the application provides, on which the medical software depends are subject to validation. However, partially-validated applications should not [FDA] be running in the same operating environment with life-threatening medical software. The FDA proposes some risk mitigation techniques to overcome that. Examples of these techniques as specified in [FDA 2002] are memory partitioning or any other approaches to resource protection.

In the case of any software upgrade or new releases, regression test is an important way to reconfirm validation, especially as it tests the new portions and ensures that they have no negative impact on other existing parts of the software.

When a medical device relies on a software or an automated equipment, then the medical software producer (manufacturer) has to prepare a user requirements specification document, per each software or automated equipment, which defines [FDA 2002]:

- the "intended use" of the software or automated equipment; and
- the extent to which the device manufacturer is dependent upon that software or equipment for production of a quality medical device

For the software that is purchased Off-The-Shelf(OTS) to support a functionality in the medical software, the medical device producer is responsible for 'ensuring that the product development methodologies used by the OTS software developer are appropriate and sufficient for the device manufacturer's intended use of that OTS software' [FDA 2002]. The job of the medical software producer starts with acquiring validation documentation from the software's vendor. These pieces of documentation have to be sufficient to establish confidence that the software has been validated. In cases where these documents are not available or the vendor refuses to give them, the medical device producer should consider other options. Depending on the risk of this software and where possible, the medical device producer (manufacturer) should [FDA] 'consider auditing the vendor's design and development methodologies used in the construction of the OTS software' [FDA 2002]. If vendors

do not have a documented lifecycle process that support the validation of the OTS, black-box testing will be the next option. In this sense, black-box testing aims to ensure that the 'software meets user needs and intended uses' [FDA 2002].

'Depending upon the risk of the device produced, the role of the OTS software in the process, the ability to audit the vendor, and the sufficiency of vendor-supplied information, the use of OTS software or equipment **may or may not be appropriate**, *especially if there are suitable alternatives available.* The device manufacturer should also consider the implications (if any) for continued maintenance and support of the OTS software should the vendor terminate their support' [FDA 2002].

Lastly and for developed software products or some commercial tools where intensive black-box testing may seem to be impractical, the FDA guidance accepts the fact that their validation may be inferred by other means. For example, 'compilers are frequently certified by independent third-party testing, and commercial software products may have "bug lists", system requirements and other operational information available from the vendor that can be compared to the device manufacturer's intended use to help focus the "black-box" testing effort' [FDA 2002].

In conclusion, the medical device manufacturer has to submit to the FDA authorities a documentation which describes the validation effort being made and justifies the magnitude of effort with regard to the software risk. This documentation should [FDA] describe [FDA 2002]:

- defined user requirements;
- validation protocol used;
- acceptance criteria;
- test cases and results; and
- a validation summary

Validation protocol is defined in [FDA 1993] as 'a written plan stating how validation will be conducted, including test parameters, product characteristics, production equipment and decision points on what constitutes acceptable test results' [FDA 1993]. According to the FDA in [FDA 2002], such a documentation 'objectively confirms that the software is validated for its intended use' [FDA 2002]. In this sense, the test plan which is specified in terms of attributes, measures and relationships, sufficiently covers the validation protocol used, acceptance criteria and validation summary.

The following figure describes the "preparation of user requirements specification document" process. Such a process has the "System Description" of the medical software as its input product. The output product of this process is the "User Requirements Specification Document" of the partner application or automated device on which the medical software depends.

The input product (system description) has already been modelled. Thus, the output product is our concern. As previously described, user requirements specification document

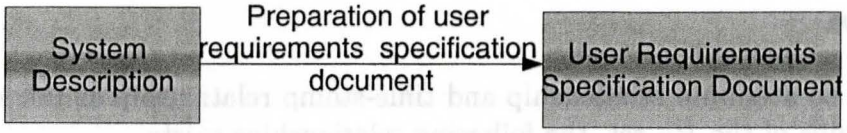


Figure 4.29. Preparation of user requirements specification document

should describe the intended use of the software/equipment and the level to which the medical software depends on this particular software or automated tool. In this sense, the user requirements specification document, a product of concern, is modelled in terms attributes which characterize it. These attributes are:

- **Application name**; describes, in a nominal scale, the name of the application on which the medical software depends to operate.
- **Application provider (vendor) name**; describes, in a nominal scale, the name of the firm which produces the application on which the medical software depends.
- **Application version**; describes, in a nominal scale, the version of the application which is used by the medical software.
- **Application purpose**; describes the set of all reasons, each described as pieces of text, that urge the usage of this application by the medical software.
- **Application assumptions**; describes the set of assumptions, each described as pieces of text, on which the application base to operate. This set only contains the assumptions that are related to the support the application gives to the medical software.
- **Application constraints**; describes the set of all constraints, each described as pieces of text, that restrict the application functionalities. In this context, this set also contains the application constraints that are only related to the operation of the medical software.
- **Application all-features**; describes the set of all features, each described as pieces of text, which the application provides.
- **Application supportive-features**; describes the set of features with which the application supports the medical software.

These attributes represent the elements of the E_B set as per the formal definition of a black-box system (S_B=(E_B,R_B)) Having them defined along with their measurement scales forms the basis for evaluating the application/equipment which this document describes.

Relationships

In addition to co-attribute relationship and time-stamp relationship as the key two mandatory relationships of the R_B set, the following relationships exists

- **Application all-features–supportive-features**; the set of supportive-features is a subset of the set of all features which the application has. Such a subset has only the features which the medical software uses.
- **Application constraints–medical software constraints consistency**; no conflict is accepted between the constraints of the application and the constraints which restrict the operation of the medical software.
- **Application assumptions–medical software assumptions consistency**; the assumptions of the application/equipment cannot break any of the assumptions on which the medical software depends to operate.

4.5.6 Acceptance Test Execution

Acceptance test execution is the process concerned with determining whether the product has achieved its acceptance criteria or not. Therefore, the result of executing the acceptance test plan helps in determining whether or not the product is ready to be released.

Definitions

No definitions are given with regard to this task in the FDA guidance document [FDA 2002].

Attributes

As given in the following figure, the acceptance test execution process has the test plan as its input product. The output product of this process is the same as the input product, yet the output product is accompanied with the actual values (results) of testing. In other words, the input product (test plan) only has the required values for acceptance which manifest as ranges of accepted values. However, test plan as the output product has the actual values that resulted from the testing process at each milestone.

In this context, and as described in the requirements development activity, the test plan has the "key milestones" attribute which describes the set of milestones that compose the testing process. A milestone description as a sub-product of the test plan is specified in terms of the following attributes: *milestone name, projected time, actual time and output products & their acceptance criteria.*

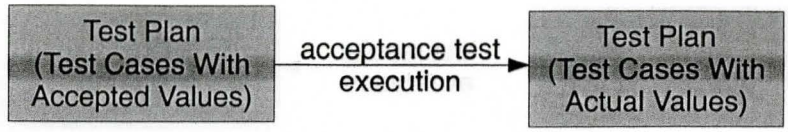


Figure 4.30. Acceptance test execution

Relationships

The relationships among attributes are the ones among attributes of the test plan, which were previously detailed.

4.5.7 Test Results Evaluation

Test results evaluation is a typical task in the testing activity performed by the developer The evaluation takes place by comparing the actual test results with the expected ones which form a basis afterwards for making the pass/fail verdict.

Definitions

The documentation of test results is a prerequisite to the evaluation of these results. Therefore, the FDA guidance document [FDA 2002] recommends that test procedures, test data, and test results should [FDA] be documented in a 'manner permitting objective pass/fail decisions to be reached. They should also be suitable for review and objective decision making subsequent to running the test, and they should be suitable for use in any subsequent regression testing' [FDA 2002] Indeed, the guidance recommends that test reports should [FDA] 'comply with the requirements of the corresponding test plans' [FDA 2002] Furthermore, the guidance recommends that proper documentation demonstrating the validation of tools used in software testing should [FDA] be maintained.

Attributes

The following figure describes the "test results evaluation" process. The "Test Plan" outputted from the acceptance test execution process represents the input product of this process. Such a plan contains the accepted and actual values of test cases.

The "Test Result Evaluation Report" is the output product of this process. This report is similar to the test plan in the sense that it also contains test cases. However, these test cases are the ones that failed. In other words, these are the test cases where variances exist between accepted results and actual results. In essence, the test case description is

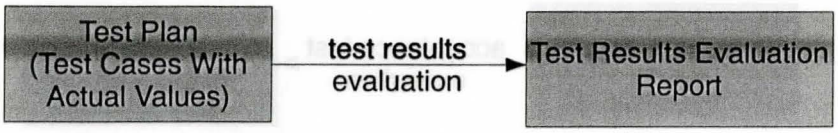


Figure 4.31 Test results evaluation

a sub-product of this report. It is characterized by the following attributes: *test case Id, test case summary, input parameters, expected values, actual values, result and justification for disapproval, if any.* Likewise any other product, attributes are the characteristics of the product. They represent the elements of the set E_B as per the definition of a black-box system (S_B=(E_B,R_B))

Relationships

The output product (test result evaluation report) has the same attributes of the test plan, since it contains the test cases (same elements of a test plan) that failed. Thus, the relationships among these attributes are the same relationships among attributes of the test plan. Hence, these relationships are not reconsidered in this section.

4.5.8 Error Evaluation/Resolution

Error evaluation/resolution is another typical task (subprocess) of the testing activity which is recommended to be undertaken in [FDA 2002] by the software developer Thus, it is considered herein in this section.

Definitions

For the errors detected during the testing process, the FDA guidance [FDA 2002] requires these errors to be reported, logged, fixed and verified. Thus, the flow of any reported error (bug) should follow the problem reporting and resolution procedures documentation.

Attributes

The following figure describes the "error evaluation/resolution" process. This process has the "Test Results Evaluation Report", was outputted from the test results evaluation process, as its input product. Also, the process has the "Tickets (Defects) Document" as its output product.

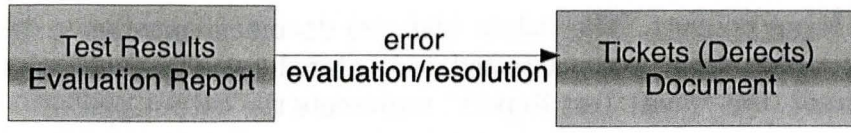


Figure 4.32. Error evaluation/resolution

The tickets (defects) document is a printout of the defects which resulted from the test results evaluation report. These defects are entered to the problem (defect) tracking system. Thus, each entered ticket (bug) represents an instance of the ticket definition as per the problem reporting and resolution procedures documentation. Such a definition describes the ticket (error) from the time of its issuance till it is resolved. As previously described, the ticket definition is characterized by the following attributes: *ticket Id, urgency, ticket summary, ticket (bug) status, reported by, assigned to, tested by, verified by, and the time at which the ticket status is changed.* These attributes with their measurement scales were thoroughly specified in the quality planning activity

Relationships

The relationships among attributes of the ticket definition were described in the quality planning section. These relationships, in addition to the co-attribute relationship and time-stamp relationship, are: *ticket status sequence, verified–resolved by, reporting–resolving–verifying date, and duration of ticket closure–urgency.*

4.5.9 Final Test Report

The preparation of a final test report is the last typical task (subprocess) of the testing activity which is recommended to be undertaken by the software developer

Definitions

No further definitions are given in [FDA 2002] with regard to this typical task, except that the final test report represents the output product of this typical task.

Attributes

The following figure describes the "preparation of the final test report" process. Such a process has the "Tickets (Defects) Document", which was outputted from the error resolution

process, as its input product. The tickets (defects) document provides a description to the life flow of the tickets which were detected and entered into the problem tracking system. On the other hand, the "Final Test Report" represents the output product of this process.

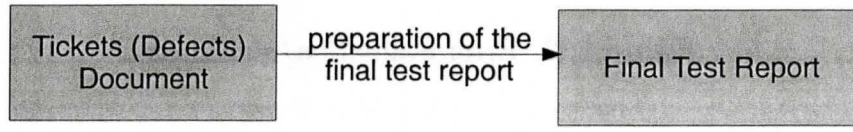


Figure 4.33. Preparation of the final test report

The final test report is the output product of this process. In essence, this product appears as a companion document to the test plan. In addition to the document-related attributes, it is modelled (specified) in terms of the following attributes that characterize it:

- **Date of issuance**; describes, in a nominal scale, the date at which the report is issued.
- **Decision makers**; describes the set of all members, each identified by his/her name and job description, who participated in making the decision.
- **Conclusions**; describes the set of all conclusions, each described as pieces of text, reached by the end of the validation process.
- **Overall validation result**, describes, in a nominal scale, the final verdict made on the overall validation process. The final pass/fail value is an accumulation to the results of all test cases.
- **Recommendations, if any**; describes the set of all suggestions, each described as pieces of text, made by the validation committee members in order to be considered by the software developers/testers.

These attributes represent the elements of the set E_B as per the definition S_B=(E_B,R_B) Furthermore, the final test report, characterized by these attributes, as well as the test plan filled with testing results both form an evidence that the software has been validated. Such an evidence has to be submitted to the FDA before actual user site testing starts.

Relationships

For this product, the co-attribute relationship and time-stamp relationship, described in chapter three, are the only two relationships in the R_B set.

4.5.10 Testing by the Software Developer, the big picture

The typical tasks (subprocesses) of testing activity (process) undertaken by the software developer have been described. Each subprocess has been specified in terms of input and output products. These products were modelled (specified) as black-box systems characterized by attributes whose measures are defined. The relationships among these attributes were also annotated. After having described the atomic subprocesses that constitute this kind of testing, the following figure gives an overall description to this testing process. The process of "testing by the software developer" has the "Requirements Of Developer's Testing" as its input product, and the "Developer's Testing Outcome Report" as its output product.

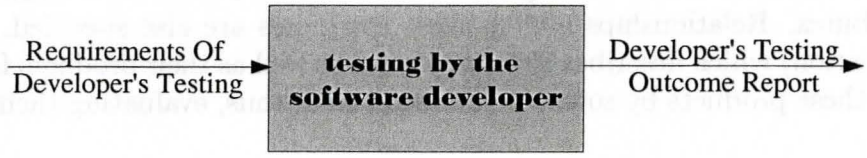


Figure 4.34. Testing by the Software Developer activity (process)

The "Requirements Of Developer's Testing" takes the form of a document. Such a document references the input products of the subprocesses of the testing by software developer process. In more detail, the "Requirements Of Developer's Testing" references the "Software Requirements Specification Document", "Software Sepcificatio/Implementation", "Software Design Specification Document", "Software Design Tests Document", "Design Communication Link Documentation", "System Tests Document", "System Description", "Test Plan (Test Cases With Accepted Values)", "Test Plan (Test Cases With Actual Values)", "Test Results Evaluation Report" and "Tickets (Defects) document" In the same manner, the "Developer's Testing Outcome Report" references the output products of the subprocesses of the testing by software developer activity (process) In more detail, the "Developer's Testing Outcome Report" references the "Test Plan", "Structural Test Cases", "Functional Test Cases", "Unit Tests To Detailed Design Traceability Matrix", "Integration Tests To High Level Design Traceability Matrix", "System Tests To Software Requirements Traceability Matrix", "User Requirements Specification Document", "Test Plan (Test Cases with Actual Values)", "Test Results Evaluation Report", "Tickets (Defects) Document" and the "Final Test Report" The description of the input and output products of this process (testing by the software developer) goes with the P/p methodology requirements in the sense that the process has only a single input product and it produces a single output product as well.

From a marshalling perspective, "acceptance test execution", "test results evaluation", "error evaluation/resolution" and "preparation of the final test report" are the only subprocesses which cannot work in parallel. In more detail, acceptance test execution has to wait for test planning process to produce the test plan. Likewise, test results evaluation has to wait for the acceptance test execution to finish. Correspondingly, error evaluation/resolution has to wait for test results evaluation to finish. Finally, preparation of the final test report

also has to wait for the error evaluation/resolution. These subprocesses cannot work in parallel since input products of some them are produced by other precedent subprocesses in the same activity (process). Whereas, all other subprocesses of the testing process can work in parallel.

Similarly to other activities (processes) in the FDA approach, the Product/process (P/p) method is used to model the "testing by software developer" activity (process) as given in [FDA 2002]. The FDA guidance describes this activity in terms of other typical tasks (subprocesses). Each typical task of this process is described as given by the FDA in [FDA 2002]. After that, each typical task (subprocess) is described in terms of its input and output products. Each product is then modelled by attributes which characterize it. Measures are given to these attributes. Relationships among these attributes are also specified. Such a modelling (specification) which describes these processes as well as their products forms the basis for preparing these products by software producers and, thus, evaluating them by the FDA.

4.6 User Site Testing

User site testing is the activity concerned with testing the medical software outside its development environment. It is considered by the FDA to be an essential part of software validation. As other activities (processes), this process is described in [FDA 2002] in terms of the following typical tasks (subprocesses):

- Acceptance Test Execution
- Test Results Evaluation
- Error Evaluation/Resolution
- Final Test Report

Therefore, these subprocesses will be considered each in turn.

4.6.1 Acceptance Test Execution

The execution of acceptance test is concerned with determining whether the product has achieved its acceptance criteria or not. This typical task, as recommended to be undertaken in [FDA 2002], is similar to the acceptance test execution task of the "testing by software developer" process. However, it takes place at the user's site with actual hardware and software.

Definitions

The FDA guidance uses the term "user site testing" to 'encompass beta test, site validation, user acceptance test, installation verification, installation testing and any other testing that takes place outside of the developer's controlled environment' [FDA 2002] Moreover, the FDA guidance states that this kind of testing should [FDA] take place at 'a user's site with the actual hardware and software that will be part of the installed system configuration' [FDA 2002] Indeed, this kind of testing is accomplished 'through either actual or simulated use of the software being tested within the context in which it is intended to function' [FDA 2002] Furthermore, test planners should [FDA] check with the FDA officers to ensure whether or not there are any additional regulations that have to be met before performing testing at the user site.

For the testing activity, user site testing should [FDA] follow a 'pre-defined written plan with a formal summary of testing and a record of formal acceptance. Moreover, documented evidence of all testing procedures, test input data, and test results should be retained' [FDA 2002] Furthermore, the FDA guidance requires the submission of pieces of evidence to prove that versions of hardware and software have been installed and configured. The guidance points out that testing process should continue for the period specified in the plan. This period represents the 'sufficient time to allow the system to encounter a wide spectrum of conditions and events in an effort to detect any latent faults that are not apparent during more normal activities' [FDA 2002]

Attributes

As given in the following figure, the acceptance test execution process has the "User Acceptance Testing Necessities Document" as its input product. Such a product references both the test plan and the configuration/installation documentation. The test plan contains the the required values for acceptance, which appear as ranges of accepted values for test cases. The configuration/installation documentation describes the hardware or software which has to be configured or installed in the user's environment before testing starts.

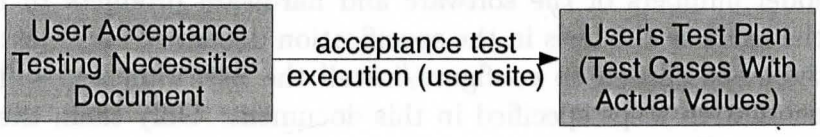


Figure 4.35. Acceptance test execution (user site)

Since the test plan has been already modelled (specified), the configuration/installation documentation is our concern. Such a document describes the configuration or installation steps which have to be undertaken to prepare the user site environment. The configuration/installation documentation is characterized by the following attributes:

- **Hardware description**; describes the set of all hardware products to be configured in the production environment. The description of each hardware product is a sub-product in itself. It can be characterized by the following attributes
  - *Hardware name*; describes, in a nominal scale, the name of the hardware to be configured.
  - *Hardware model no.*; describes, in a nominal scale, the model number of the hardware which is to be configured in the production environment as per the agreed-upon specification.
  - *Hardware specs*; describes the set of all specification items, each described as tuples of attribute name and value, as given by the hardware producer.
  - *Configuration steps*; describes the set of steps, each described as pieces of text, to be carried in sequence to ensure proper configuration of the hardware at the user's site environment.
- **Software description**; describes the set of all software to be installed in the production environment. Likewise hardware, the description of each software product is a sub-product in itself. It can be characterized by the following attributes:
  - *Software name*; describes, in a nominal scale, the name of the software to be installed.
  - *Software version*; describes, in a nominal scale, the version of the software which is to be installed in the production environment.
  - *Software features*; describes the set of features that this particular software provides. The list of all software features are given in the specification documents by its producers. Thus, the specification has to be referenced herein.
  - *Installation steps*; describes the set of steps, each described as pieces of text, to be undertaken to ensure proper installation of the software at the user's site environment.

Acceptance test execution involves two major validation steps. Firstly, ensuring that the versions and model numbers of the software and hardware products to be installed and configured are the same as the ones in the specification documents and test plan. Secondly, ensuring that the steps carried to configure/install the hardware and software follow the configuration/installation steps specified in this document. Only then, the test plan, with all testing results specified, sufficiently serves the FDA purpose in describing all the testing activities to confirm that the software is validated.

Relationships

All relationships among attributes of the test plan are applicable for this section. These attributes were thoroughly detailed in the quality planning section. Moreover, the "testing environment–hardware/software description consistency" is a new interrelationship between

the "testing environment" as an attribute of the test plan and "hardwarde/software description" as an attribute of the configuration/installation documentation. In other words, the value of testing environment attribute of the test plan must meet the values of hardware/software description in the configuration/installation documentation.

4.6.2 Test Results Evaluation

Test results evaluation is a typical task (subprocess) of the user site testing activity (process) Test results evaluation is concerned with comparing the results of test cases which were undertaken on the developer's environment during acceptance test execution, with results of test cases undertaken at the user's site environment, also during acceptance test execution.

Definitions

The FDA guidance document [FDA 2002] requires that some of the evaluations took place at the developer's control environment should [FDA] be repeated, but at the user's site. In this context, the guidance proposes that the developer may provide the user with some set of test data. Furthermore, the FDA guidance recommends that there should [FDA] be an evaluation to 'the ability of the users of the system to understand and correctly interface with it' [FDA 2002]

Attributes

The following figure describes "test results evaluation (user site)" process. Such a process has the "Developer-User Test Plan" as its input product. This product references both test plans resulted from acceptance test execution at developer's side and the corresponding ones at the user's site.

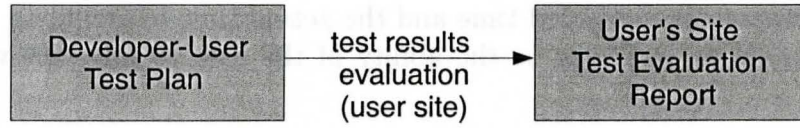


Figure 4.36. Test results evaluation (user site)

The results of test cases, as performed at both developer's and user's environment are specified in the test plans. Thus, having the developer-user test plan referencing these plans facilitates the comparison between the results of test cases at the developer's side and the ones at the user's site. Any test case which results in an unaccepted variation between the developer's test and the user's test has to be documented in the "User's Site Test Evaluation

Report". In other words, this report contains the test cases which failed to have same results (values) at both developer's and user's environments.

For the evaluation of users ability to understand and interface with the software, a user performance review template has to prepared to document the user performance while using certain functionalities. In this sense, the template can be specified (modelled) in terms of the following attributes:

- **User name**; describes the name of the user performing this demo.
- **Experience level**; describes, in an ordinal scale, the level of experience the user has. Values for this attribute are: novice, intermediate, expert.
- **Functionality to be carried**; describes, in a nominal scale, a particular functionality which the software provides and the user wants to try.
- **Functionality complexity**; describes, in a ordinal scale, the level of complexity of the functionality to be tried by a user. Values for this attribute can be: easy, moderate and difficult.
- **Estimated time to complete**; describes, in an absolute (counting) scale, the anticipated time (to the closest time unit) for the user to accomplish the demo.
- **Actual time to complete**; describes, in an absolute (counting) scale, the actual time (to the closest time unit) which took the user to finish the job (if completed successfully).
- **Success/failure in completing the job**; describes, in a nominal scale, whether the user managed to finish the demo in the estimated time or not. Values for this attributes are restricted to: success or failure.
- **User comments, if any**; describes any comments, each described as pieces of test, provided by the user after undertaking such an experience.

The difference between the projected time and the actual time to complete the job (if completed successfully) is an indicator to the ability of the user to interface with the system properly.

Any failure to complete the functionality by a potential user should be questioned. Questioning every user failure to complete the functionality, especially if that failure was due to a shortcoming in the software interfaces should lead to developing a test case to handle this shortcoming. Such a test case must be documented, in addition to other failed test cases, in the user's site test evaluation report. These defects will be considered in the next task (error evaluation/resolution).

Relationships

The test plan is a key product in this process. Thus, all relationships among its attributes are applicable to this section. For the user performance review template, and in addition to the co-attribute and time-stamp relationships, the following relationship exists among attributes of this product:

- **Experience level–Complexity mapping**; a mapping exists between the values given to the experience level which is required to test a functionality, and the complexity of that chosen functionality. For instance, it is not acceptable for a novice user to try a difficult functionality.

Thus, these relationships represent the elements of the R_B set as per the definition S_B=(E_B,R_B).

4.6.3 Error Evaluation/Resolution

Error evaluation/resolution is another typical task (subprocess) of the testing activity (process) which is recommended to be undertaken at the user site testing. This task is similar to the one which is conducted by the developer. However, it takes place at the user's control environment. Therefore, it is briefly discussed in this section.

Definitions

In this context, the FDA guidance requires keeping records of both 'proper system performance and any system failures that are encountered' [FDA 2002]. Also for defects found during user site testing, the flow of any reported error (defect) should follow the problem reporting and resolution procedures documentation.

Attributes

The following figure describes the "error evaluation/resolution (user site)" process. The "User's Site Test Evaluation Report" is the input product of this process. Whereas this process has the "User's Site Tickets (Defects) Document" as its output product.

Similarly to the tickets (defects) document at the developer's side, the user's site tickets (defects) document is a printout of the defects which resulted from the user's site test evaluation report. These defects are entered to the problem (defect) tracking system. Thus, each entered ticket (bug) represents an instance of the ticket definition as per the problem reporting and resolution procedures documentation. Such a definition describes the defect (error) throughout its lifecycle. As detailed in the quality planning activity, the ticket definition is characterized by the following attributes: *ticket Id; urgency; ticket summary; ticket*

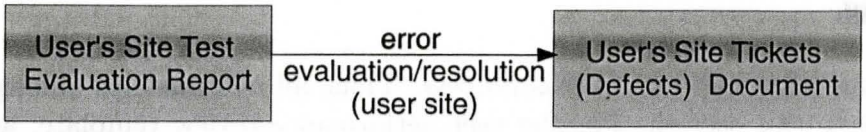


Figure 4.37. Error evaluation/resolution (user site)

(bug) status; reported by, assigned to, tested by, verified by; and the time at which the ticket status is changed. These attributes sufficiently characterize any ticket from the time of its issuance till it is resolved.

These attributes represent the elements of the set E_B as per the definition $S_B=(E_B,R_B)$. Furthermore, the final test report, characterized by these attributes, along with the test plan filled with actual testing results form an evidence that the software has been validated at the user's site.

Relationships

Herein also, and in addition to the co-attribute and time-stamp relationships, the relationships among attributes of ticket definition are: *ticket status sequence, verified–resolved by, reporting–resolving–verifying date, and duration of ticket closure–urgency.* These relationships have been thoroughly described in section 4.1.

4.6.4 Final Test Report

The preparation of a final test report is the last typical task (subprocess) of the testing activity (process) which is recommended to be undertaken in [FDA 2002] at the user's site. This task is also similar to the one which is conducted by the developer. However, it takes place at the user's control environment.

Definitions

No further definitions are given in [FDA 2002] with regard to this typical task.

Attributes

The following figure describes the "preparation of the final test report (user site)" process. Such a process has the "User's Site Tickets (Defects) Document", which was outputted from the error resolution process of the user site testing activity, as its input product. The user's

site tickets (defects) document contains the description of the life flow of defects discovered at the user's site. On the other hand, the "User's Site Final Test Report" represents the output product of this process.

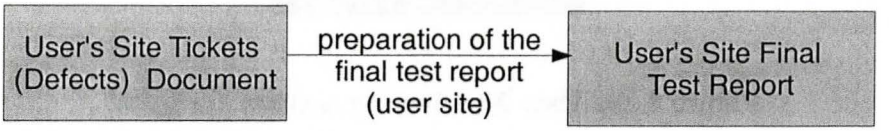


Figure 4.38. Preparation of the final test report (user site)

Herein also, the output product (user's site final test report) appears in the form of a document. Such a document specifies the conclusions of the testing committee about the final software. Thus, the document is modelled (specified) in terms of the following attributes: *date of issuance, decision makers (essentially representatives from the user's side, but may include others from the developer's side), conclusions, overall validation result, and recommendations, if any.* In this context, the final test report, characterized by these attributes, as well as the test plan (assigned with the values of actual results) represent two pieces of evidence to be submitted to the FDA for a testing process which took place at the user's site, where the actual hardware and software are used.

Relationships

The co-attribute and time-stamp relationships, detailed in chapter three, are the only two relationships over the attributes of this product.

4.6.5 User Site Testing, the big picture

The typical tasks (subprocesses) of the testing activity (process) undertaken at the user's site have been described. Each subprocess has been described in terms of input and output products. These products were modelled (specified) as black-box systems characterized by attributes whose measurement scale types are defined. The relationships among these attributes were also annotated. Abstractly, and as described in the following figure, the user site testing process has the "Requirements Of User Site Testing" as its input product. Whereas it has the "User Site Testing Outcome Report" as its output product.

The "Requirements Of User Site Testing" takes the form of a document. Such a document references the input products of the subprocesses of user site testing activity (process). In more detail, the "Requirements Of User Site Testing" references the "User Acceptance Testing Necessities Document", "Developer-User Test Plan", "User's Site Test Evaluation Report" and the "User's Site Tickets (Defects) Document". In the same manner, the "User

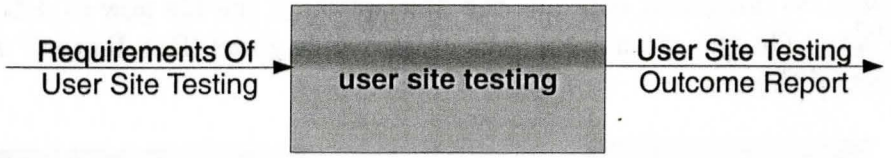


Figure 4.39. User Site Testing activity (process)

Site Testing Outcome Report" references the output products of the subprocesses of the user site testing. In more detail, the "User Site Testing Outcome Report" references the "User's Test Plan (Test Cases with Actual Values)", "User's Site Test Evaluation Report", "User's Site Tickets (Defects) Document" and the "User's Site Final Test Report". Such a description of this activity (process) goes with the P/p methodology requirements in the sense that user site testing process has only a single input product and a single output product.

From a marshalling perspective, the subprocesses of the user site testing have to undertaken in the sequence in which they were described. In other words, no parallel execution is possible between these processes. This is due to the fact that the output product of each process represents the input product of the subsequent one.

Similarly to any other activity (process) defined in [FDA 2002], the Product/process (P/p) approach represents any engineering problem in terms of processes and product. Thus, we used to model (specify) the user site testing activity (process). The FDA guidance describes this activity in terms of other typical tasks (subprocesses). For each typical task, we described it as given in the FDA guidance [FDA 2002] followed by its input and output products being modelled. The modelling mechanism aims to provide a simplified representation of the products in terms of attributes which characterize them, relationships among these attributes as well as measurement scales for these attributes. Such a specification helps software producers in measuring the attributes of these products. Furthermore, it forms a basis for the evaluation of these products afterwards.

4.7 Maintenance & Software Changes

Maintenance & software changes is the last activity which is recommended to be undertaken in [FDA 2002]. Therefore, it is described, similarly to other activities (processes), in terms of other typical tasks (subprocesses) that are concerned with the preparation of some products [FDA 2002]:

- Software Validation Plan Revision
- Problem Identification and Resolution Tracking

- Anomaly Evaluation
- Proposed Change Assessment
- Documentation Updating
- Task Iteration

The FDA guidance [FDA 2002] stresses on the difference between software maintenance and hardware maintenance in the sense that software maintenance includes corrective, perfective and adaptive maintenance but not preventive maintenance or component replacement as in hardware maintenance. By corrective maintenance the guidance means 'changes made to correct errors and faults in the software' [FDA 2002]. In the same manner, perfective maintenance is described as 'changes made to the software to improve the performance, maintainability, or other attributes of the software system' [FDA 2002]. Whereas adaptive maintenance is described as 'software changes to make the software system usable in a changed environment' [FDA 2002].

4.7.1 Software Validation Plan Revision

The process of revising the software validation plan is a typical task (subprocess) of the maintenance & software changes activity (process). This process is concerned with planning for validation of newly changes or any updates. In other words, the software validation plan is concerned with describing the validation effort of software changes.

Definitions

The guidance states that the 'validation effort necessary for each software change is determined by the type of change, the development products affected, and the impact of those products on the operation of the software' [FDA 2002]. Therefore, the validation plan is mainly concerned with demonstrating that any new change has been tested before it is plugged into the software. It is further concerned with demonstrating that all other components are not negatively impacted by such a change.

Attributes

The following figure describes the process of "revising the software validation plan". This process has the "Software Change Requirements" as its input product. This product takes the form of a document. It references both the "Design Communication Link Analysis Document" and the "Software Change Description Documentation". For its output product, this process has the "Revised Software Validation Plan".

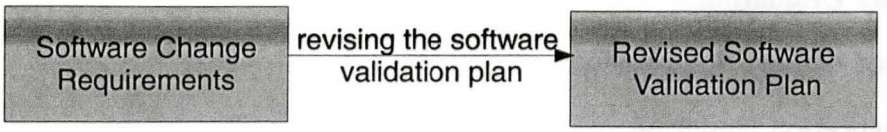


Figure 4.40. Revising the software validation plan

The software change description documentation describes the change to be undertaken. In more detail, the change is handled as a feature which has to be implemented. As described in the requirements development section, a feature is described as a set of requirements description. Each requirement description is characterized by the following attributes: *requirement Id, requirement type, requirement specification type and specified requirement.* On the other hand, the design communication link describes the components of the system. Since this description shows the dependencies between system components, it tells which components are affected by such a change.

A revised software validation plan is the output product of this process. It describes the set of validation tasks similarly to the verification & validation plan (detailed in the quality planning activity). Indeed, each validation task description is a sub-product of the validation plan. As previously described, a validation task description is characterized by the following attributes: *task name, personnel description, time, task duration, location, task complexity, description of the acceptance criteria, equipment description and effort.* These attributes suffice the description of the validation task, the key sub-product of the software validation plan. Therefore, no new attributes are added herein. However, new validation tasks for the newly added/modified features will be added to the set of all validation tasks.

Relationships

Likewise the relationships among attributes of the V&V plan, and in addition to the co-attribute and time-stamp relationships, the following two relationships exist among attributes of the revised software validation plan: *resources-duration-effort conformance and task difficulty-level of expertise.* These relationships are the elements of the R_B set.

4.7.2 Problem Identification and Resolution Tracking

Problem identification and resolution tracking is a typical task (subprocess) of the maintenance & software changes activity (process). This task is concerned with identifying the flaws that are discovered during the validation of the newly implemented software changes. Furthermore, it is concerned with entering these flaws as defects in the problem (bug) tracking system.

Definitions

The FDA guidance states that 'all problems discovered during maintenance of the software should be documented' [FDA 2002]. In a sense, the flow of any problem (defect) found during maintenance should follow the defect lifecycle defined in the problem reporting and resolution procedures documentation.

Attributes

The following figure describes the "problem identification and resolution tracking" process. The "Revised Software Validation Plan" is the input product to this process. Whereas this process has the "Software Changes Tickets (Defects) Document" as its output product.

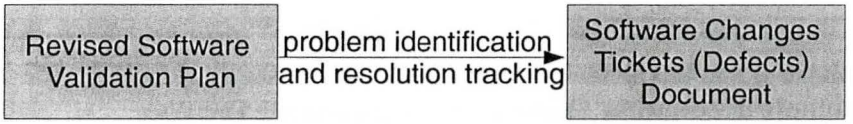


Figure 4.41. Problem identification and resolution tracking

The input product to this process was modelled (specified) in the previous process. Therefore, it is not reconsidered in this section. The acceptance criteria of each validation task is described in the input product. Thus, those tasks which do not meet their acceptance criteria are identified as defects. These defects will be entered to the problem tracking system. In this context, the output product is a printout of those defects which resulted from the change. Each defect (ticket), as in the problem reporting and resolution procedures, is characterized by the following attributes: *ticket Id; urgency; ticket (bug) status; ticket summary; reported by, assigned to, tested by, verified by; and the time at which the ticket status is changed.* These attributes are the sufficient descriptions of any defect, detected as a result of software change, from the time of its issuance till it is resolved.

Relationships

The relationships among attributes of the ticket definition were thoroughly specified. Also, the ones among attributes of the software validation plan were previously detailed.

4.7.3 Anomaly Evaluation

The evaluation of discovered anomalies is another typical task (subprocess) of the maintenance & software changes activity. An anomaly evaluation document results from an analysis

to the discovered flaws (defects) and it contains the recommendations to be considered in order to avoid the recurrence of these anomalies in the future.

Definitions

The FDA guidance states that software anomalies should [FDA] be 'treated as symptoms of process deficiencies in the quality system' [FDA 2002]. Thus, anomaly evaluation (analysis) has to be conducted. The motivation of such analysis is that software defects are usually repeated and recurred.

Attributes

The following figure describes the "anomaly evaluation" process. Such a process has the "Software Changes Tickets (Defects) Document" as its input product. On the other hand, it has the "Anomaly Evaluation Document" as its output product.

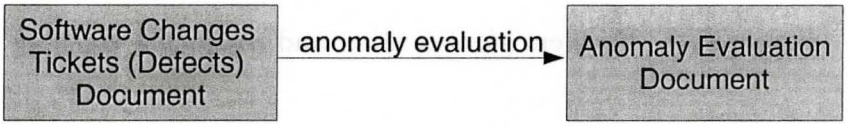


Figure 4.42. Anomaly evaluation

The software changes tickets (defects) document is the input product of this process. Since it has been already modelled, it is not reconsidered in this section. Whereas, the anomaly evaluation document is the output product of this process. It is mainly concerned with describing high level recommendations to be considered in order to avoid the recurrence and repetition of detected software defects. The tickets (bugs), maintained in the problem tracking system (bug tracking system) and appeared in the software changes defects (tickets) document, will be analysed to reach such recommendations. Thus, the anomaly evaluation document can be modelled in terms of the following attributes that describe (characterize) it:

- **Evaluation committee**; describes all members, identified by their names and job descriptions, who participate in the anomaly evaluation.
- **Strategies used**; describes the set of all strategies (mechanisms), each described as pieces of text, that are followed to reach the recommendations in this document. These strategies are well-known in the literature and thus they can just be referenced. An example of such a strategy as given in [FDA 2002] is "root cause analysis of anomalies".
- **Observations**; describes the set of all observations, each described as pieces of text, that are remarked by any of the evaluation committee members.

- **Conclusions**; describes the set of all conclusions, each described as pieces of text, that are reached by the evaluation committee members.
- **Recommendations**; describes the set of all recommendations, each described as pieces of text, which obtained the consensus of the evaluation committee members.
- **Components affected**; describes the set of all components, each nominally specified, to be reconsidered as per the recommendations made by the committee.

Specifying the attributes which characterize this product facilitates the development of this product. Furthermore, it enables evaluating it afterwards as the output product of the anomaly evaluation process.

Relationships

For the anomaly evaluation document, the co-attribute relationship and time-stamp relationship are the only two relationships in the R_B set.

4.7.4 Proposed Change Assessment

Proposed change assessment is another typical tasks (subprocess) of the maintenance & software changes activity (process). It is concerned with ensuring that all validation tasks for the new changes are assessed.

Definitions

In this context, the FDA guidance requires that 'all proposed modifications, enhancements, or additions should be assessed to determine the effect each change would have on the system' [FDA 2002]. Such an assessment is the determinant of the magnitude of effort needed to validate the new features and other components which may be affected by the change.

Attributes

The way in which the process of "revising the software validation plan" was specified implicitly included this process. In more detail, revising the software validation plan has the the software change requirements document as its input product. Such a document references both the design communication link analysis and software change description documentation. Software change description documentation which describes the change to be undertaken is also specified in terms of its attributes. The design communication link describes the

components of the system. Since this description shows the dependencies between system components, it tells which components will be affected by such a change. Thus, the validation effort is estimated and specified in the software validation plan per each validation task.

As a reminder, the revised software validation plan describes the set of validation tasks. Each validation task description is a sub-product of the validation plan. It is characterized by the following attributes: *task name, personnel description, time, task duration, location, task complexity, description of the acceptance criteria, equipment description and* ***effort.***

Relationships

As previously described, the relationships among attributes of the revised software validation plan are the same ones that exist among attributes of the software validation plan.

4.7.5 Documentation Updating

Documentation updating is a typical task of the maintenance & software changes activity which is recommended to be undertaken in [FDA 2002]. This task is concerned with ensuring that all documents affected by a software change are identified in order to be updated, before the change itself is implemented.

Definitions

The FDA guidance addresses documents that may be impacted by any software change. The guidance requires these documents to be maintained up-to-date. In fact, the FDA guidance states that all specification documents should [FDA] be 'updated before any maintenance and software changes are made' [FDA 2002].

Attributes

The following figure describes the "documentation updating" process. Such a process has the "Software Change Requirements" as its input product. On the other hand, the process has the "List Of Affected Documentation" as its output product. As previously described, the software change requirements document references both the software change description documentation and the design communication link analysis. The software change description documentation describes the change to be undertaken. Whereas, the design communication link analysis shows the main software components and the dependencies among them.

Given the software change as described in the software change description documentation, and the design communication link analysis which describes the components that will be

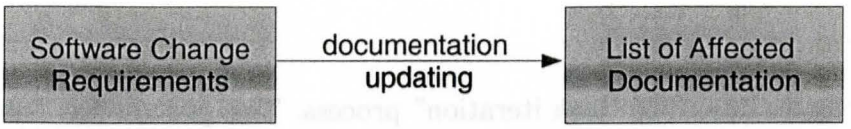


Figure 4.43. Documentation updating

affected by such a change, the list of affected documentation contains all documents which will also be affected by undertaking this change. The List of affected documentation has the document description as its key sub-product. This product is characterized by the following attributes:

- *Document name*; describes , in a nominal scale, the document which has to be updated as a result of the proposed change.
- *Document section*; describes , in a nominal scale, the particular section of the document which is affected by the proposed change and thus it has to be updated.
- *Change summary*; describes, as pieces of text, the modification steps which have to be undertaken in order to make sure that the document is maintained up-to-date.

Similarly to any other product, these attributes represent the elements of the E_B set as per the formal definition of a black-box system (S_B=(E_B,R_B)).

Relationships

For the relationships among the attributes of this product, the co-attribute relationship and time-stamp relationship are the only two relationships in the R_B set.

4.7.6 Task Iteration

Task iteration is a typical task (subprocess) of the maintenance & software changes activity (process). It is concerned with ensuring that changes are implemented correctly, have no negative effect on the original components and all documents affected by these changes are also updated.

Definitions

The FDA guidance requires that validation tasks should [FDA] be performed to 'ensure that planned changes are implemented correctly, all documentation is complete and up to date, and no unacceptable changes have occurred in software performance' [FDA 2002].

Attributes

The following figure describes "task iteration" process. This process has "Software Changes Documentation" as its input product. Whereas the process has the "Software Changes Validation Confirmation" as its output product.

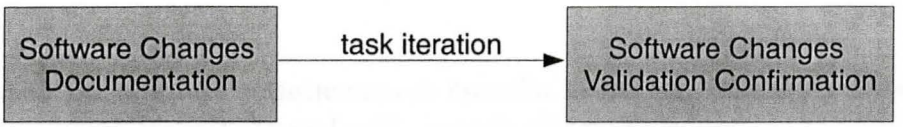


Figure 4.44. Task iteration

In this context, "Software Changes Documentation" references the "Revised Software Validation Plan", "Software Changes Tickets (Defects) Document" and the "List of Affected Documentation". Therefore, and given these pieces of evidence, this process is concerned with ensuring that all new changes have been validated, defects resulted from these changes have been resolved and documents specified in the list of affected documentation have been updated. Only then, this process produces the "Software Changes Validation Confirmation". The procedures (steps) followed by medical software producers to confirm the correct implementation are left out to software producers. However, and as specified in the modelling structure, some comments and suggestions are given. In this sense, regression testing should be carried over samples of the original test cases to make sure that no negative effect has been introduced. Furthermore, an auditing process should be undertaken over document modifications in order to make sure that the updates were successfully done. Likewise, an auditing process should be carried to ensure that reported anomalies were successfully fixed, tested and verified. On the other hand, the software changes validation confirmation takes the form of a document which describes the committee members who participated in the final verdict that the software changes were successfully validated, in addition to their validation strategy as well as any comments.

Relationships

For relationships over attributes of the input product, the relationships among attributes of the ticket definition have been already specified. Also, the ones among attributes of the software validation plan were detailed. For the relationships among attributes of the list of affected documentation, only the co-attribute and the time-stamp relationships exist among these attributes.

On the other hand, for relationships over attributes of the output product (software changes validation confirmation), the co-attribute and time-stamp relationships are the only ones.

4.7.7 Maintenance & Software Changes, the big picture

The typical tasks (subprocesses) of maintenance & software changes activity (process) have been described. Each subprocess has been specified in terms of input and output products. These products were modelled (specified) as black-box systems characterized by attributes whose measures are defined. The relationships among these attributes were also pointed out. The following figure gives an overall description to the "maintenance & software changes" process. Such a process has the "Software Changes Listing" as its input product. Whereas, it has the "Software Changes Validation Report" as its output product.

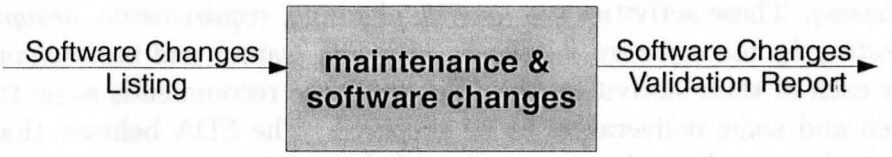


Figure 4.45. Maintenance & Software Changes activity (process)

In this sense, The "Software Changes Listing" takes the form of a document. Such a document references the input products of the subprocesses of the maintenance & software changes process. In other words, the "Software Changes Listing" references the "Software Change Requirements", "Revised Software Validation Plan", "Software Changes Tickets (Defects) Document" and "Software Changes Documentation". On the other hand, the "Software Changes Validation Report" references the output products of the subprocesses of the maintenance & software changes process. In other words, the "Software Changes Validation Report" references the "Revised Software Validation Plan", "Software Changes Tickets (Defects) Document", "Anomaly Evaluation Document", "List of Affected Documentation" and "Software Changes Validation Confirmation". Similarly to other activities, the description of this activity (process) goes with the P/p methodology requirements in the sense that this process has only a single input product and a single output product.

From a marshalling perspective, both "revising the software validation plan" and "documentation updating" can work in parallel. However, the "problem identification and resolution tracking" has to wait for the "revising the software validation plan" process to produce the "Revised Software Validation Plan" in order to start. Likewise the "anomaly evaluation" has to wait for the "problem identification and resolution tracking". Lastly, the "task iteration" process has to wait for "revising the software validation plan", "problem identification and resolution tracking" and "documentation updating" processes to finish in order to start.

In conclusion, the Product/process (P/p) method is used to model maintenance & software changes activity of the FDA validation approach. The (P/p) modelling approach takes a generic systems approach in simplifying any engineering problem to be represented as a set of products and processes. Thus, the products of these typical tasks (subprocesses) are modelled (specified) in terms of attributes which characterize each product. Measures were

given to these products. Relationships over these attributes were also described. Such a specification facilitates the evaluation of these products.

4.8 Summary

The FDA has published guidance documents concerning medical software validation. In more detail, the FDA guidance [FDA 2002] describes the software lifecycle in terms of some activities (phases). These activities are: *quality planning, requirements, design, construction or coding, testing by the software developer, user site testing and maintenance & software changes.* For each of these activities, the FDA guidance recommends some typical tasks to be undertaken and some deliverables to be prepared. The FDA believes that undertaking these typical tasks as well as the preparation of these deliverables supports the conclusion that the software is validated.

The guidance documents published by the FDA do not explicitly state what are the entities to be measured (processes or products of these processes), what attributes they should have, which characterize the products/processes, and even what kind of scale is needed to measure each attribute. Certainly, this leaves much room for subjective judgment and misunderstanding.

To overcome this problem, we used the Product/process (P/p) method to model the activities as defined by the FDA validation approach in [FDA 2002]. The (P/p) modelling approach, introduced by Myers and Kaposi in [Myers and Kaposi 2004], tends to be systematic and generic. It models any engineering problem as a set of products and processes. Thus, we described each validation-supportive activity in the FDA approach, as per the P/p methodology, as a process with input and output products. After that, we modelled each product in terms of attributes that characterized it. Furthermore, we defined the measurement scale types of each attribute as well as the relationships among attributes. In essence, the modelling activity does not aim to impose a definitive proposal on the FDA. Instead, it aims to illustrate the existing problems and the way in which these matters could be handled.

In brief, we illustrated the need for a precise agreement between software producers and certification authorities. Such an agreement should be explicit in order to be able to impose a contract between the two parties. By contract we mean a clear and explicit specification which describes the objects (products) that are subject to assessment, the measurable attributes that characterize them, the measurement scales types of these attributes as well as the interrelationships among them.

Chapter 5

Conclusion and Future work

This chapter describes the conclusions reached at the end of this thesis. It further describes the future work that may result from this thesis.

5.1 Conclusion

The FDA is a public agency that is concerned with the certification of medical device software. The FDA has published several guidance documents to express its current thinking about medical software validation. However, these guidance documents are not explicit enough to impose a contract between software producers and certification authorities. In this context, a contract means a clear and explicit specification which describes the objects that are subject to assessment, the measurable attributes that characterize them and the assessment procedures that are sufficient to tell whether the product is at a satisfactory quality or not.

In this thesis, we questioned the clarity and feasibility of the FDA approach for evaluating and certifying medical software. We evaluated the FDA approach by examining the guidance documents published by the FDA. In this sense, we started this thesis by showing the drawbacks of the current FDA validation approach as they appear throughout the FDA guidance documents. These drawbacks manifest in the inconsistency and the vagueness of the FDA validation function. For evidence of inconsistency, we showed the overlapping of the FDA approach between process and product evaluation. Furthermore, we showed how the FDA guidance document [FDA 2002] had no consistent approach to tackle the validation of deliverables (products). For evidence of vagueness, we described how the FDA guidance documents neither defined the attributes of the evidence to be measured, nor did they define the acceptable values for these attributes. We made these deficiencies obvious when the FDA validation approach was compared with another evaluation approach, the

Common Criteria (CC) for Information Technology Security Evaluation. Despite its focus on evaluating software security capabilities, the systematic methodology, which the CC follows, clearly defines the evaluation activities (the aspects which the FDA approach lacks).

From the FDA perspective, the FDA guidance document [FDA 2002] considers validation as an activity which should be planned at the very beginning of software development. Therefore, the guidance recommends some activities to be undertaken and some deliverables to be prepared during software development. However, and even for the deliverables of the FDA activities, the guidance does not explicitly describe the characteristics of these deliverables despite their crucial participation in the decision whether the software is at a satisfactory quality and it should be certified or not. Certainly and as described in chapter two, this leaves much room for subjective judgment and misunderstanding.

After highlighting the drawbacks of the FDA validation approach as they appear in the second chapter of this thesis, we believe that the FDA validation approach leads to confusion in the minds of both software producers and evaluators and it may well lead to lower quality software than desired being certified. To overcome this problem, the FDA validation approach has to be simplified and expressed in a more explicit manner. Thus, we considered modelling in the third chapter. P/p modelling helps in providing a simplified representation for any engineering situation. P/p modelling methodology represents any engineering problem as a system with processes and products. Therefore, we described each activity of the FDA validation approach as a process which composes other subprocesses that in turn have input and output products. Moreover, we modelled (specified) these products in terms of attributes that characterize them, along with the measurement scales of these attributes. Furthermore, we explained the relationships among attributes of these products. This specification forms the basis for measuring these attributes and, accordingly, validating these products by software producers. It also forms the basis for certifying the products by the FDA staff. In more detail, we used the P/p modelling methodology to specify the FDA validation approach in a more simplified way. This was accomplished by describing the FDA processes with their input and output products. We modelled (specified) these products in terms of attributes which characterize them. We further accompanied each attribute with a measurement scale type which describes the type of the value that will be given to this attribute. Moreover, we explained the interrelationships, relationships among attributes of the same product as well as relationships among attributes of different products. The description of these relationships aimed to ensure that some essential and required properties are maintained.

In response to preserving the flexibility characteristic of the FDA approach, we did not specify the measurement procedures that will be followed to measure these attributes and validate their measured values by software producers. Nevertheless, we suggested some procedures to be used. This enables software producers to choose the measurement methodology, which they believe to be most feasible, from the software engineering literature. However, the software producer has to document the measurement methodology which was used, or even reference it.

In conclusion, we have the activities of the FDA validation approach modelled in terms of input and output products. The products are modelled in terms of attributes that charac-

terize them and relationships among these attributes. The attributes are defined as measures in the sense that measurement scale types are given to these attributes. The measurement procedures that are followed by software producers in measuring these attributes should be documented by software producers. On the other hand, the FDA has to define the acceptance/rejection criteria which would be approved and followed in evaluating these products. The acceptance criteria are dependent on the variances discovered, during an audit, between actual values and accepted values of products' attributes. This evaluation (certification) mechanism is outside the scope of this thesis. However, we will touch on it in the following section.

5.2 Future work

As future work motivated by this thesis, objective certification criteria can be defined hand in hand with the FDA. Such certification criteria will describe the accepted level of tolerance for each attribute and, consequently, for each product. In a more detailed flow, the software producer has to submit evidences to the FDA staff which prove that the products' attributes have been measured and validated in accord with specific measurement procedures and validation methodology. After that, the FDA will hold an auditing activity to make the decision whether these products and, accordingly, the medical software have to be certified or not. The decision should be made based on a defined criterion which shows the actual faults detected per a product attribute versus the acceptable faults per a particular attribute of the product. This approach supports the fact that some attributes may have more significance and, thus, more weight than others in approving the product.

Bibliography

[FDA 1993] *Guidance on General Principles of Process Validation*, May 1987, Reprinted 1993, http://www.fda.gov/CDER/GUIDANCE/pv.htm

[FDA 1999] *Guidance for Industry, FDA Reviewers and Compliance on Off-The-Shelf Software Use in Medical Devices*, September 1999, http://www.fda.gov/cdrh/ode/guidance/585.pdf

[FDA 2002] *General Principles of Software Validation; Final Guidance for Industry and FDA staff*, January 2002, http://www.fda.gov/cdrh/comp/guidance/938.pdf

[FDA 2005] *Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices; Guidance for Industry and FDA staff*, May 2005, http://www.fda.gov/cdrh/ode/guidance/337.pdf

[CC 1999] *Common Criteria for Information Technology Security Evaluation, User Guide*, October 1999, http://www.commoncriteriaportal.org/public/files/ccusersguide.pdf

[CC 2006a] *Common Criteria for Information Technology Security Evaluation, part 1: Introduction and general model*, version 3.1, revision 1, September 2006, http://www.commoncriteriaportal.org/public/files/CCPART1V3.1R1.pdf

[CC 2006b] *Common Criteria for Information Technology Security Evaluation, part 2: Security Functional Requirements*, version 3.1, revision 1, September 2006, http://www.commoncriteriaportal.org/public/files/CCPART2V3.1R1.pdf

[CC 2006c] *Common Criteria for Information Technology Security Evaluation, part 3: Security Assurance Requirements*, version 3.1, revision 1, September 2006, http://www.commoncriteriaportal.org/public/files/CCPART3V3.1R1.pdf

[CC 2006d] *Common Methodology for Information Technology Security Evaluation, version 3.1*, revision 1, September 2006, http://www.commoncriteriaportal.org/public/files/CEMV3.1R1.pdf

[Abdeen, Kahl, Maibaum 2007] Marwan M. Abdeen, Wolfram Kahl, and Tom Maibaum, *FDA: Between Process & Product Evaluation*, in Proc. of HCMDSS/MD PnP 2007, to be published by the IEEE Computer Society Press.

[Basili, Caldiera, Rombach 1994] Victor R. Basili, Gianluigi Caldiera, and H. Dieter Rombach, *The Goal Question Metric Approach*, Encyclopedia of Software Engineering, Wiley&Sons Inc., 1994.

[Dobson 2003] Michael S. Dobson, *Project Management: How to Manage People, Processes, and Time to Achieve the Results You Need*, Adams Media Corporation, 2003.

[Fenton and Pfleeger 1997] Norman E. Fenton and Shari Lawrence Pfleeger, *Software Metrics (2nd Ed.): A Rigorous and Practical Approach.* PWS Publishing Co., 1997.

[Myers and Kaposi 2004] Margaret Myers and Agnes Kaposi, *A First Systems Book*, Imperial College Press, 2004.

[Perry 2000] William E. Perry, *Effective Methods for Software Testing*, Second Edition, John Wiley & Sons, 2000.

[Wiegers 2003] Karl E. Wiegers, *Software Requirements*, Second Edition, Microsoft Press, 2003.