MeRC

McMaster eBusiness Research Centre

A Proposed Intelligent Policy-Based Interface for Mobile eHealth Environments

By

Amir Tavasoli and Norm Archer

archer@mcmaster.ca

McMaster eBusiness Research Centre (MeRC)

WORKING PAPER No. 28
January 2009

McMaster University

A Proposed Intelligent Policy-Based Interface for Mobile eHealth Environments

Amir Tavasoli and Norm Archer
McMaster University

ABSTRACT

Users of mobile devices in eHealth systems are often novices, and the learning process for them may be very time consuming. In order for systems to be attractive to potential adopters, it is important that the user interface should be very convenient and easy to learn. However, the community of potential users of a mobile eHealth system may be quite varied in their requirements, so the system must be able to adapt easily to suit user preferences. One way to accomplish this is to have the user interface driven by intelligent policies. These policies can be refined gradually, using inputs from potential users, through agents supported by artificial intelligence approaches such as neural networks. These policies may then be used to support adaptation to interfaces that are suitable for users who may have different requirements. This paper develops a framework for policy refinement for eHealth mobile interfaces, based on dynamic learning from user interactions.

Keywords: Mobile eHealth, adaptive interfaces, artificial intelligence, policy-based networking, intelligent agents.

INTRODUCTION

Mobile wireless systems continue to increase in popularity in many disciplines and applications, for both leisure and work. These range from wireless online games, tourist information, and traffic news to teleconferencing and mobile commerce applications such as ticketing and e-mail (Buellingen and Woerter 2004). But there has also been a great deal of interest in mobile wireless applications in supporting healthcare (Orwat, Graefe et al. 2008). This discipline is known as mobile eHealth, with the more technical aspects sometimes described as healthcare compunetics (Bos, Laxminarayan et al. 2004). As such applications become more sophisticated, there is an inevitable trend towards the direct support of patients, particularly those with chronic diseases, so they can manage their own healthcare. To accomplish this individual support for the range of users who are either healthcare practitioners (physicians and nurses) or patients managing their own health, an intelligent environment would be extremely useful to help the system adapt to users with a wide range of needs, using mobile handheld devices such as personal digital assistants, smart phones, or cell phones. For instance patients suffering from diabetes may need to monitor their status regularly, with the system tracking and warning them about out-of-range indications, reminding them to take medications, or assisting in communicating with practitioners for emergency or regularly scheduled office visits. Moreover there are intelligent devices that patients can wear that monitor patient health indicators (Baker, Armijo et al. 2007). Ultimately, handhelds may help to diagnose disease status or help the patient in emergency situations. It is also expected that mobile eHealth applications will maintain or improve patient quality of life while at the same time reducing system costs, paper forms, delays and errors (Archer 2007).

In recent years much work has been done in order to solve technical issues around mobile wireless networks. Wireless network applications should be usable, adaptable, interoperable, change resistant, and secure (Kunz and Gaddah 2005; Archer 2007). To tackle these issues at the network level, where hundreds of devices may be in use, proper computer techniques need to be considered. One of the techniques that has been applied at the network level to address device and network management issues is policy-based networking (Kosiur 2001). Since most people that work with these systems are not computer experts, the system should be simple and convenient enough for anyone to use - i.e. the system should be usable and adaptable (Archer 2007). The user interface clearly plays a key role in these features, and this is especially important in mobile systems where the device screen is limited in size (Tarasewich 2003; Zhu, Nah et al. 2003). For example, the placement of a link on the device screen can affect usability and adaptability. What information is easily accessible and how it is displayed on the screen can be absolutely critical to its relevance and usefulness.

Advances in policy-based networking, coupled with artificial intelligence, have provided tools that will not only manage applications effectively, but will support adaptability to different users who may display a wide range on needs. This is particularly true with mobile eHealth applications, where some users such as nurses and physicians may be very familiar with healthcare requirements, but novices at using mobile devices. On the other hand, others such as patients with chronic diseases may be novices at managing their own healthcare and more or less familiar with the use of mobile devices. To cater to this broad range of end users requires a system interface that is not only highly usable but adaptable to a wide range of users.

Intelligent User Interfaces (IUI) (also known as adaptive user interfaces), which apply artificial intelligence concepts to user interfaces, are not new. IUIs have been used to handle standard application requirements such as information overflow or to help end-users to manage complex systems (Höök 2000; O'Grady and O'Hare 2008). IUIs are complex, and researchers have studied how to implement them efficiently for some time. For example, Langley (Langley 1997) suggested using machine learning for implementing IUIs. Various frameworks have also been proposed for managing IUI for mobile systems (Mitrovic, Royo et al. 2005; O'Grady and O'Hare 2008).

In this paper, we describe the general applicability of policy-based networks to the support of wireless applications for eHealth. Then we develop a framework for a policy-based interface that can adapt itself to user needs with the help of artificial intelligence, using a policy refinement method that refines the policy gradually, according to both user choices and learning from user actions. This implements the IUI concept in mobile eHealth applications, through a novel combination of artificial intelligence, agents, and policy-based networking.

POLICY-BASED NETWORKING

As computer networks continue to grow in size and complexity, their management is becoming more challenging. At the same time, these networks are becoming critical parts of business, so methods are needed to manage and control these sophisticated networks both efficiently and effectively using business parameters. Classical manual methods are too complex and slow for the large numbers of devices and users linked to such networks, and it is almost impossible for these methods to consider all real business concerns in a timely manner. Policy-based networking (PBN) (Kosiur 2001) is a way of managing network resources using pre-defined business policies. Instead of focusing on devices and interfaces, PBN focuses on users and applications. It is actually an efficient automated extension of manual classical management methods, allowing administrators to effectively manage resources according to their business needs. It also ensures the desired level of quality of service and security of the network using a centralized approach. This allows network managers to be more productive because they do not need to be concerned about network management details and can focus on generating real business value. Policy Management for Autonomic Computing (PMAC) is a good example of a platform that can be used for PBN (Agrawal, Lee et al. 2005). PMAC is basically a generic policy middleware platform that can support the management of large scale distributed system networks. It also provides software components that can be embedded in software applications to take input from policy-based management systems.

With PBN, each policy is a set of rules and instructions that dictates how the network is to use its resources. A policy is the management point of view of how the network should deal with its users, in a variety of situations ranging from IP assignment to security enforcement (Kosiur 2001). For example, one of a set of policies that could be used to enforce firewall security might be:

```
If userIPAdress is 192.168.1.2 then
      BlockUser;
```

In PBN, policies are mechanisms to make network management more efficient, such as setting application priorities. Due to its hierarchical data structure, ease of representation, and its broad support, XML is normally used to represent policies. For example, XML was used to implement the Core Information Model (CIM) stand-alone policies (CIM 2003).

Policy-based networking mostly deals with issues around network management. However, in this work we focus on the use of policies that can be used to build and manage user interfaces, a key business aspect of the eHealth environment because of the wide range of user capabilities and requirements. This becomes an even more critical issue because of limitations due to the small screen size of mobile devices (Tarasewich 2003). This extends the PBN concept to handle the user interface according to real user needs, requiring special types of policies that differ from those used in other PBN systems reported to date.

AGENTS IN POLICY-BASED ENVIRONMENTS

Agents are an innovative concept that can help to automate network management in policy-based mobile environments (Ganna and Horlait 2005). Moreover agents have proven to be useful in large-scale distributed networks (Bieszczad, Pagurek et al. 1998). In this paper we describe a special kind of policy which is quite different from traditional policies used in PBN. Traditional policies are built from rules (Kosiur 2001), which are conditional if-then-else statements. Implementing these rules in specific situations requires matching situation parameters with policy conditions. If there is a successful match, the condition body is applied; otherwise, nothing is done. We will refer to these policies as 'solid policies'.

This paper implements a special kind of policy which is actually a learner agent. An agent is anything that receives data from its environment and acts on it (Russell and Norvig 2003). In our system, the agent receives user inputs and its resulting actions are used to build or modify an intelligent interface. The agent's goal is to derive an output which will lead to the best interface that it can provide, when it receives user input. Because the agent needs to receive data from users and decide on the best output automatically, it is considered an intelligent agent (Wickramasinghe, Amarasiri et al. 2004). To implement an intelligent agent obviously involves more than if-then-else rules. What we really need is an intelligent policy. We call these policies 'dynamic policies'. To implement intelligent agents in these environments there are various artificial intelligence techniques available, including artificial neural networks, fuzzy logic, etc. For the purpose of building and modifying dynamic policies we have chosen artificial neural networks (ANNs) due to their powerful generalization, speed, flexibility and adaptability (Bailey and Thompson 1990).

Artificial Neural Networks

In ANNs (Negnevitsky 2005), the artificial neurons are connected to each other with weighted links, which magnify or reduce an incoming signal to a neuron. 'Learning' is the process of adjusting these weights in order to make the network give the best results, based on historical situations and experiences, from a database that has accumulated these experiences.

The use of neural networks for implementing dynamic agents can be justified by considering the properties of problems where ANN applications have been used successfully (Bailey and Thompson 1990). These properties include:
- They are data-exhaustive and depend on various interacting factors
- There is a great deal of historical data or examples
- A deterministic function to find a solution does not exist or is hard to discover

A survey of successful neural-network application developers (Bailey and Thompson 1990) also showed that ANN support of heuristics was successful in applications with:
- Insufficient conventional computer technology
- Requirements for intricate or qualitative quantitative reasoning

Dynamic policies have almost all of the requirements mentioned. There will be many users who will generate a great deal of data as they interact and adjust the system interface to suit their needs, so the system is data exhaustive. There are an enormous number of possible interface solutions and, because each user has individual interface needs, it will not be possible to find good solutions to these needs deterministically. Each user choice of a specific interface then becomes an example and thus can be stored as historical data for future solutions.

Some common properties of ANNs are adaptability, generalization, speed and flexibility (Bailey and Thompson 1990). These properties will, first, allow the interface agent to adapt to user needs automatically and dynamically. Second, the agent will be able to generalize a user's screen with some properties and then adjust to the best screen for users with somewhat different needs. Third, the agent must be fast enough to generate an interface with little noticeable delay for the users. Finally, the agent should be flexible enough to deal with large numbers of users. Neural networks can also tolerate irrelevant data (McCollum 1998), which will ensure that the interface reacts gradually to sudden requirement changes. ANNs have been used in some cases to develop conversational interfaces for mobile users (Toney, Feinberg et al. 2004; O'Grady and O'Hare 2008), reflecting some results upon which our research can build in this environment.

To apply a neural network to the proposed agent a vector of data that represents choices of a particular user will be input to the trained network. The network will generate a number vector as its output. Thus, user preferences need to be translated into the input vector and the output vector will be applied to build the interface. The next section discusses the translation of user choices into input vectors, how to interpret the output to build the required interface, and how to use the neural network to support policies.

POLICY-BASED INTERFACES

A policy-based interface framework includes both profiles and policies. Each user device or machine will have a user profile and an interface profile. There will be a Local Interface Agent at each user machine. At the server side we will have a Central Interface Agent. There are three ways to handle agents in a mobile environment (O'Grady and O'Hare 2008): 1) the network-based approach, where all of the agents reside on the server; 2) the distributed approach, where agents reside on both the mobile devices and the servers, and; 3) the embedded approach, where all agents are stored on the user devices. Although it is preferable to maintain agents locally (on the

devices) because of speed and network bandwidth savings, handheld devices may not be powerful enough to support Local Interface Agents. In our proposal, the Local Interface Policy does not require a great deal of local computation due to its simple design. Nevertheless, it would still be possible if desired to maintain all of the agents on the server. Since network and device bandwidth may not be high enough to handle this situation, the choice of where to maintain the agents requires performance tests to determine the optimal configuration.

For the purpose of this paper, we will assume that the handhelds and the network have the bandwidth and computing capacity to run the agents locally. The physical configuration of the proposed system is demonstrated in Figure 1.

User Profile

For the purpose of this proposal, we will assume that the users are patients being treated for chronic diseases such as diabetes, heart disease, asthma, etc. The system could also be used for healthcare practitioners such as physicians and nurses, but obviously their profiles would be much different. In this environment each user will have a profile which contains user information such as disease or diseases that afflict the user, drugs being taken, time schedules, treatments needed, recommended exercises, medical appointment schedule, health status (blood pressure, heart rate, blood sugar level, etc.) For applying these profiles to the Local Interface Agent, the attributes will need to be transformed into a number vector. Data (Weiss, Indurkhya et al. 2004) in these profiles are categorical data, so a codebook would be constructed from a set of numerical codes assigned to each item, and stored in the codebook. As an example, a disease codebook might contain entries such as those shown in Table 1.

Table 1. Example Disease Codebook

Disease	Code
Asthma	1
Diabetes	2
Nephrosis	3
…	…

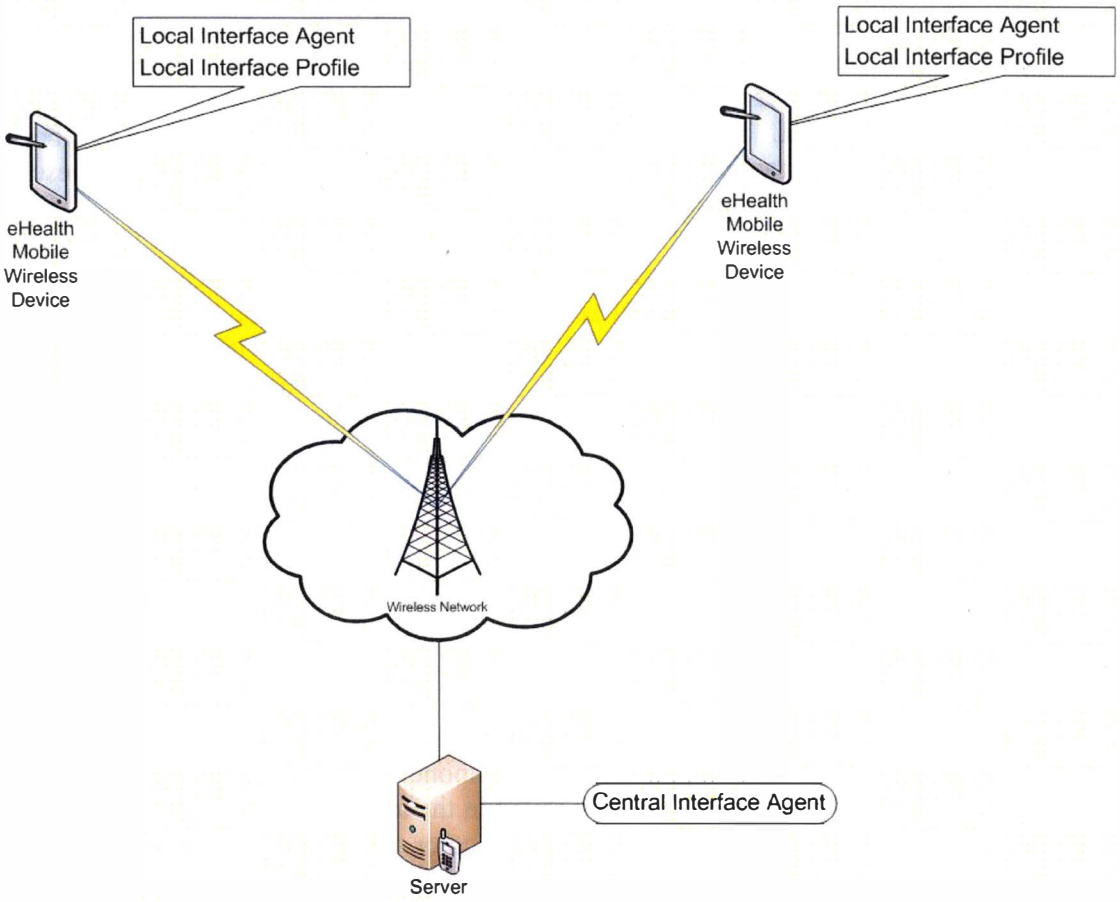

Figure 1. Physical Distribution of Agents and Profiles

Based on such a codebook, a profile for a patient who has diabetes and takes insulin injection medications could be represented as a numerical vector: <2, 1…>

Obviously more complex conversion methods could be used if required, based on existing standards for medical treatment regimens.

Local Interface Policy

There are two types of agents in the framework. One, Local Interface Agents, control the Local Interface Policy at each device, and the single Central Interface Agent controls the central interface policy, which we will describe shortly. The Local Interface Agents have dynamic policies that build and maintain the local interfaces. To build the interface, user profile information will be sent to the Local Interface Policy agent as agent input, and the agent output are used to set up links and other relevant information such as page configurations in the local device. Each time the local system boots up, the interface will be updated according to the policy that relates to that system. The general logical relationship of the network's interface profiles, policies, agents, and interfaces is demonstrated in Figure 2.

Note that what is meant by an interface component is anything appearing on the screen, ranging from hyperlinks to icons. The technique that will be used for building the initial interface is straightforward. First, the user profile is converted to a number vector, which becomes the input to the Local Interface Agent. The output of the Local Interface Agent will be a number[1] vector that sets priorities for the arrangements of the interface components, for example. These literally represent the priority levels of the components[2].

As an example, there are two ways to arrange the components on a screen. One uses scrolling and the other uses menus that require several screen pages (Wu, Cao et al. 2008). For a scrolling interface the components would be entered individually, starting with the top priority, and working down from there, with higher priority components clustered so they are more easily reached without excessive scrolling. When a multiple page interface is used, each page can be assigned a set of say eight components, and the components assigned to other pages by working down in priority from there. Then the pages can be prioritized according to the component numbers in them. Other properties such as size and colour can also be taken into account. It is expected that issues such as component placement, size, and colour would be significant considerations in designing and maintaining optimal interfaces for individual users.

Interface Profile

Each user will have an interface profile, which records user changes to interface components. For example when a user moves the drug-scheduling component to the top of the screen this revised information about component position will be saved in the user profile. The profile can obviously be used to save other component properties like size, color, icon used, placement in the hierarchy tree menu, etc. User changes will dynamically override initial fixed choices that are the starting position for the user profile, and they will be saved for future sessions with the device since they represent user choices and preferences. This ensures that user choices will be preserved for future sessions and will not be revised by the dynamic policy each time a new session is initiated.

Central Interface Policy

The Central Interface Policy is a server-based dynamic policy that keeps track of the interface choices of all the users dynamically. This will help in designing a suitable interface for all new users. From the agent perspective, the inputs to this agent are profiles of all of the current users of the system and their preferences in building their interfaces. Each time (Kosiur 2001) an interface is created for a new user, the initial Local Interface Policy will be inherited from the Central Interface Policy, which the user can then adapt to suit that particular user's needs. Therefore, each new Local Interface Agent that is built from the Central Interface Agent is an object that has inherited the characteristics of the Central Interface Agent. This object thus includes all of the current knowledge about interfaces in the network. As more users enter the

---

[1] *Input numbers are integer numbers and outputs can be decimal numbers. Since outputs are used for prioritizing, output numbers only need to be comparable.*

[2] *Each component in the interface is related to an item in the user profile. For example for disease we have a component to show diseases. Because of the nature of feed-forward back-propagation neural networks the number of inputs and outputs can differ, and any relation needed can be assumed. The neural network will choose the preference pattern best suited to this particular user.*

system, the Central Interface Agent will continue to learn more about the range of inputs needed, and what they can lead to in terms of outputs. For example, when a new user tries to change an icon in a menu, not only will the Local Interface Agent learn from this, but the Central Interface Agent will also learn something new. Should more users change this icon, eventually the central interface will make this change for other new users of the system. A further change that we might consider in our architecture is allowing the Central Interface Agent to update unchanged components for all similar users, so all users can benefit from the system's experience with all the interfaces.

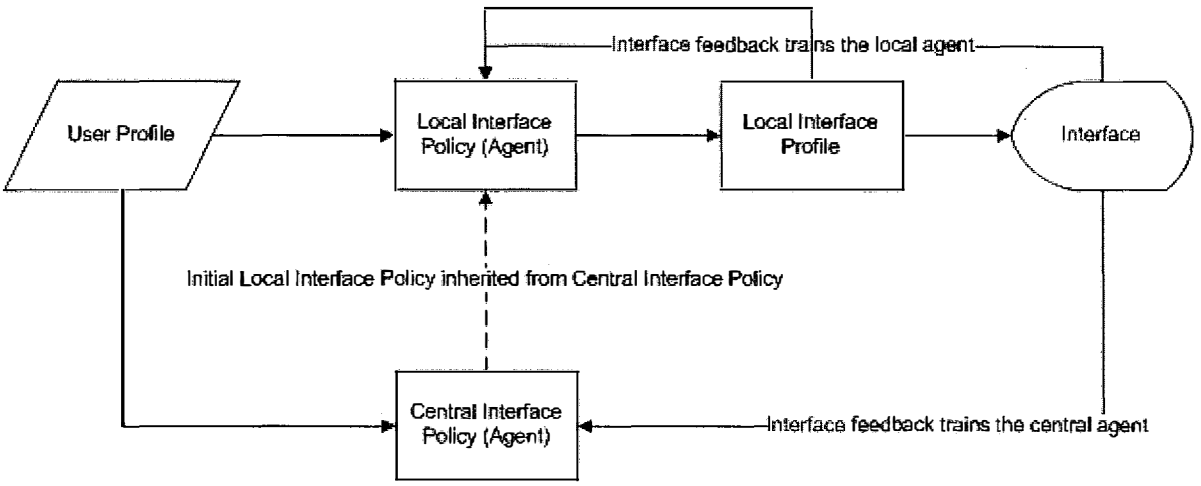

Figure 2. Logical Relationships of Network Profiles, Policies, and Agents

The property of pattern matching in neural networks, which can find similar interfaces for users with the same profiles, can be built with only a few weighting matrices, making object inheritance very convenient for the system.

ANNs can be built with only a few weighting matrices (Negnevitsky 2005), making implementation and object inheritance very convenient for the system. However, we need to consider the fact that just copying the weights to Local Interface Agents most probably will not result in the agent inheriting the properties desired for that specific user interface, since the Central Interface Agent has learned its properties from a large number of users. It might therefore take too long for the Local Interface Agent to adapt to that particular user's choices. Other methods for transferring learned properties that can be considered include training the Local Interface Agent with the inputs and outputs of the Central Interface Agent a controlled number of times. Clearly, this will involve testing in a controlled environment to find the most appropriate method.

Learning

As already mentioned, in order to implant the learning process in the system we will use Local Interface Agents, Local Interface Profiles, and a Central Interface Agent. When a new user is added to the system the user's interface will inherit its initial Local Interface Policy from the

Central Interface Policy. The interface this generates will be a representation of all the previous system interfaces for current users. The new user will modify the interface to suit his or her tastes, by dragging components to different locations, changing a link hierarchy in a menu tree, changing colours or fonts of a component, or changing a component in order to view certain information on a page according to needs. The user may otherwise begin working with the page as is. In any case, user hits on specific components on each page will be tracked, to enable training of the Local and Central Interface Agents with the observed activity (that is, the interface feedback to Local and Central Interface Policies will train and adapt them to user choices). If the user changes the place of a specific component or changes the information that it displays, the priority of that component may change. If this continues regularly, then the priority of that component will increase relative to the components that are being used less. In this way, the interface will gradually move closer to user needs and the agents will learn more about user preferences. Local Interface Agents will only be trained for their specific users, but the Central Interface Agent will receive Interface feedback from all the users, based on the structure that we have just described.

It may be that the feedback from each interface to the Central Interface Agent in a worst case scenario could result in a significant amount of data that will overwhelm the server (O'Grady and O'Hare 2008). Although this can only be determined through environmental analysis and testing, one possible solution is for Local Interface Agents and the Central Interface Agent to communicate using a standard language like KQML (Mayfield, Labrou et al. 1994). Allowing agents to communicate with each other readily will train and adapt the Central Interface Agent with less data required from devices, saving on bandwidth requirements. Again, determining whether this is an optimal solution will require testing in a real environment to determine if the additional complexity gives the desired payback.

The Local Interface Profile will save user choices about such decisions as the placement of components on the screen, and the results of the Local Interface Agent will be ignored when it comes to this profile. This is because, when the user changes component priorities, it takes time for the agent to learn, but the user expects to see an immediate change. The Interface Profile will save user choices about properties of components such as place and size, and this profile will be taken into account each time the Local Interface Agent is trained. This places an emphasis on the concrete choices of users. These concrete choices need to be emphasized locally, but emphasizing them globally through the Central Interface Agent may overwhelm the Central Interface Agent's learning process. Therefore the User Profile will only be transmitted to the Central Agent each time it is changed through Interface feedback. The learning process is demonstrated in the logical framework shown in Figure 2.

Designing the Artificial Neural Network

There are several types of ANNs and there are several types of learning algorithms for each of them. We have chosen a supervised learning back propagation approach (Russell and Norvig 2003). There are two factors to be considered. First, the ANN must be designed carefully so it converges to a good solution in a reasonable time. With, for example, a multilayer feed-forward back-propagation neural network, which seems best for this situation due to ease of implementation and training, it is important to consider the number of layers and number of

perceptrons in each layer, using a complete testing regime to establish that convergence will occur in a reasonable time. Second, before starting the project, the initial tests will make use of randomly generated data, with the participation of several human participants to make sure that the Central Interface Agent has learned enough and has enough capability to go into production. The training process for neural networks tends to be complex, so it is essential to design good evaluations and tests that will exercise the system properly before implementing the framework.

SUMMARY

In a mobile eHealth environment the visual display and other characteristics of the mobile devices that are used are relatively small and many of the users are novices. This makes an easily adaptable interface a critical issue. In this environment, for example, placement of a link on the screen is coupled with the usability and adaptability of the system. It is therefore important to place a great deal of emphasis on interface adaptability, quality, and reliability. Our proposal uses agents and policies to build an intelligent interface that can adapt itself gradually to user needs. As each user works with the system, the system will learn from user preferences and finally adapt itself to user tastes. A Central Interface Agent will be used to gather all of the knowledge generated as new users choose and adapt their interfaces, allowing the system to develop good choices for possible interfaces. Neural networks will be used as agents due to their generalization, speed, flexibility, and adaptability which support pattern recognition and generalization (Bailey and Thompson 1990).

There are two important considerations before implementing such a system. First, it will be necessary to determine whether the network and device bandwidth are capable of supporting Local Interface Agents on the handheld devices. If not, they will have to be run on the central server. Second, the Central Interface Agent must be designed initially so it presents good choices to the first users of the system. In real situations, this would require a significant amount of experimentation and data collection from users before the system is released for production.

FUTURE WORK

The initial priority for this project will be to design agents that will provide acceptable performance in the context of eHealth mobile environments, using published narratives about developing high quality neural networks (Bailey and Thompson 1990; Tsoukalas and Uhrig 1997). The second major project phase will be to design and implement a test interface, using simulations and human participants to establish the effectiveness of the system before it is actually considered for production applications.

Additionally this work could be extended in a whole policy-based network framework. Neural networks have been used for building policies in other fields (Lin and Ouyang 1997). It is quite possible to extend these dynamic policies in order to manage all of the resources in the network and not just the interface. The design of a framework to accommodate this would need to be considerably different from the one proposed in this paper. Some work has already been done, using agents to automate network management (Ganna and Horlait 2005). But these approaches have not included policy refinement. Using neural networks is a new way of building policies

that are actually intelligent and provide considerable flexibility in network management approaches.

REFERENCES

Agrawal, D., K.-W. Lee, et al. (2005). Policy-based management of networked computing systems. IEEE Communications Magazine. 43 (10): 69-75.

Archer, N. (2007). Mobile eHealth: Making the case. Mobile Government: An Emerging Direction in e-Government. I. Kushchu. Hershey, PA, Idea Group: 155-170.

Bailey, D. and S. Thompson (1990). "How to develop neural networks." AI Expert 5(6): 38-47.

Baker, C. R., K. Armijo, et al. (2007). Wireless sensor networks in home health care. 21st International Conference on Advanced Information Networking and Applications.

Bieszczad, A., B. Pagurek, et al. (1998). "Mobile agents for network management." IEEE Communications Surveys 1(1).

Bos, L., S. Laxminarayan, et al. (2004). Health care compunetics. Medical and Care Compunetics. London, UK, ISO Press.

Buellingen, F. and M. Woerter (2004). "Development perspectives, firm strategies, and applications in mobile commerce." Journal of Business Research 57(12): 1402-1408.

CIM (2003). Policy model white paper http://www.dmtf.org/standards/documents/CIM/DSP0108.pdf (Jan 9 2009).

Ganna, M. and E. Horlait (2005). On using policies for managing service provisioning in agent-based heterogeneous environments for mobile users. Sixth IEEE International Workshop on Policies for Distributed Systems and Networks, Stockholm, Sweden, IEEE.

Höök, K. (2000). "Steps to take before intelligent user interfaces become real." Interacting With Computers 12(4): 409-426.

Kosiur, D. (2001). Understanding Policy-Based Networking. New York, Wiley.

Kunz, T. and A. Gaddah (2005). Adaptive mobile applications. Encyclopedia of Information Science and Technology. M. Khosrow-Pour. Hershey, PA, IGI Global: 47-52.

Langley, P. (1997). Machine learning for adaptive user interfaces. Lecture Notes in Computer Science Vol. 1303. London, UK, Springer-Verlag: 53-62.

Lin, H. P. and Y. C. Ouyang (1997). Neural networks based traffic prediction for cell discarding policy. International Conference on Neural Networks.

Mayfield, J., Y. Labrou, et al. (1994). KQML as an agent communication language http://www.cs.umbc.edu/~finin/papers/kqml-eval.ps (Jan 19 2009).

McCollum, P. (1998). An introduction to back-propagation neural networks http://www.seattlerobotics.org/encoder/nov98/neural.html (Jan 9 2009). Seattle, WA, Seattle Robotics Organization.

Mitrovic, N., J. A. Royo, et al. (2005). Adaptive user interfaces based on mobile agents: Monitoring the behavior of users in a wireless environment. Symposium on Ubiquitous Computation and Ambient Intelligence, Madrid, Spain, Thomson - Paraninfo.

Negnevitsky, M. (2005). Artificial Intelligence: A Guide to Intelligent Systems. New York, NY, Addison Wesley.

O'Grady, M. J. and G. M. P. O'Hare (2008). Intelligent user interfaces for mobile computing. Handbook of Research on User Interface Design and Evaluation for Mobile Technology. J. Lumsden. Hershey, PA, IGI Global: Chapter 20.

Orwat, C., A. Graefe, et al. (2008). "Towards pervasive computing in health care: A literature review." BMC Medical Informatics and Decision Making 8(26).

Russell, S. J. and P. Norvig (2003). Artificial Intelligence: A Modern Approach. Upper Saddle River, NJ, Prentice Hall/Pearson Education.

Tarasewich, P. (2003). Wireless devices for mobile commerce: User interface design and usability. Mobile Commerce: Technology, Theory, and Applications. B. Mennecke and T. Strader. Hershey, PA, Idea Group Publishing: 26-50.

Toney, D., D. Feinberg, et al. (2004). Acoustic features for profiling mobile users of conversational interfaces. Lecture Notes in Computer Science Volume 3160. Berlin, Springer-Verlag: 394-398.

Tsoukalas, L. H. and R. E. Uhrig (1997). Fuzzy and Neural Approaches in Engineering. New York, Wiley.

Weiss, S., N. Indurkhya, et al. (2004). Text Mining: Predictive Methods for Analyzing Unstructured Information. New York, Springer.

Wickramasinghe, L. K., R. Amarasiri, et al. (2004). "A hybrid intelligent multiagent system for e-business." Computational Intelligence 20(4): 603-623.

Wu, B., F. Cao, et al. (2008). Scrolling versus menus on small mobile device screens: A mobile portal usability study. McMaster eBusiness Research Centre Working Paper #24. Hamilton, Ontario, DeGroote School of Business, McMaster University: 24.

Zhu, W., F. F.-H. Nah, et al. (2003). Factors influencing user adoption of mobile computing. Managing E-Commerce and Mobile Computing Technologies. J. Mariga. Hershey, PA, IRM Press: 260-271.

McMaster University
1280 Main St. W. DSB A201
Hamilton, ON
L8S 4M4

Tel: 905-525-9140 ext. 23956
Fax: 905-528-0556
Email: ebusiness@mcmaster.ca
Web: http://merc.mcmaster.ca