Task Migration in a Pervasive Multimodal Multimedia Computing System for Visually-Impaired Users*

Ali Awde[1], Manolo Dulva Hina[1,2], Yacine Bellik[3], Amar Ramdane-Cherif[2], and Chakib Tadj[1]

[1] LATIS Laboratory, Université du Québec, École de technologie supérieure
1100, rue Notre-Dame Ouest, Montréal, Québec H3C 1K3 Canada
{ali.awde.1@ens,manolo-dulva.hina.1@ens,ctadj@ele}.etsmtl.ca
[2] PRISM Laboratory CRNS, Université de Versailles-Saint-Quentin-en-Yvelines
45, avenue des Etats-Unis, 78035 Versailles Cedex, France
rca@prism.uvsq.fr
[3] LIMSI-CRNS, Université de Paris-Sud
B.P. 133, 91043 Orsay, France
yacine.bellik@limsi.fr

Abstract. In a pervasive multimodal multimedia computing system, the user can continue working on a computing task anytime and anywhere using forms of modality that suit his context. Similarly, the media supporting the chosen modality are selected based on their availability and user's context. In this paper, we present the infrastructure supporting the migration of a visually-impaired user's task in a pervasive multimodal multimedia computing environment. Using user's preferences which quantify user's satisfaction, we derive the user's task feasible configuration. The heart of this work is the machine learning-derived training to acquire knowledge leading to configuration optimization. Data validation is presented through scenario simulations and design specification. This work is our continuing contribution to advance research on making informatics more accessible to handicapped users.

1 Introduction

As the consequence of computing being present in many facets of our lives, a computing system needs to evolve to become adaptive to the environment and user's needs. For a pervasive computing [1], its infrastructure must allow users to continue working on their task when and where they wish to. This requirement should serve all types of users, including those with disability, such as the visually-impaired ones.

As the user moves from one *multimodal multimedia* (MM) system to another, computing resources and user context change. In our work, *media* refers to a set of physical interaction devices (and software supporting physical devices) while *modality* refers to the logical interaction structure. For a visually-impaired user to continue working on a task, the system takes account of user's profile, data and current

* This work has been made possible the funding awarded by the Natural Sciences and Engineering Research Council (NSERC) of Canada, and the scholarship grants from *Décanat à la formation* of École de technologie supérieure and of the National Bank of Canada.

C. Cérin and K.-C. Li (Eds.): GPC 2007, LNCS 4459, pp. 459 – 471, 2007.

environment conditions. This means determining the form of modality the user could work on a computing task. Available resources and their constraints determine the media supporting the chosen modality to use. A basic requirement of a MM infrastructure is that it must have sufficient media devices that support various forms of modality.

This paper is a continuation of our previous work [2] which presented the architectural model of a pervasive MM computing system for users with visual disability. In that work, we defined the relationships among data format, environment conditions, user's preferences and modalities and media selections. In this paper, we visualize a mobile visually-impaired user and the migration of his data and task as he moves across computing environments. We set the foundation for feasible configuration to realize the user's task. Our objective is to realize a self-adaptive system taking into account the user's needs and the changes in his environment. This work is our contribution to improving visually-impaired users' access to information and computing.

The rest of this paper is structured as follows. Related work is presented in Section 2. The building of a machine learning knowledge for feasible configuration is discussed in Section 3. The system's specification and scenario simulations are presented in Section 4. The paper is concluded in Section 5.

2 Related Work

There are various tools for people with visual disabilities to access electronic information, such as screen reader, transcription data for Braille, access to mathematics [3, 4] and speech synthesis. For instance, for screen data access, JAWS [5] identifies and interprets what is displayed on the screen. It is then presented to blind users as speech (through text-to-speech software) or as translated data meant for Braille terminal. This is integrated into our work as one data conversion tool. HOMERE [6] allows blind users to use haptic/touch and audio modalities to explore virtual environments. Although functional, the system's effectiveness is limited as the modalities for user interaction are already pre-defined. In contrast, a computing system becomes more flexible if no pre-defined input-output modalities are set. In fact, the output presentation of information should be based on the user's application and interaction context (user, system, and environment) which could possibly be in constant evolution. The framework for intelligent multimodal presentation of information [7] is an example. The system's user interface also should be adaptive to these context variations while preserving its usability. Demeure's work [8] exhibits plasticity in context adaptation. Indeed, the forms of modality should be chosen only based on their merits to a user's interaction context. This is the approach adopted in our work.

Our focus has always been pervasive multimodality for the blind. This work was initially inspired by [9]. As our work evolves, however, the knowledge representation that we have derived becomes different as we affirm that our optimization model is best reflective for our intended user. The methodology is different as this paper uses *machine learning* (ML) to acquire knowledge. Such knowledge is stored onto the knowledge database (KD) accessible from a member of server group so that it can be made omnipresent, accessible anytime and anywhere via wired or wireless networks.

A major challenge in designing systems for the blind is how to deliver autonomy onto the user. To this end, several tools and gadgets have emerged in recent years, among them are the GPS (global positioning system), walking stick that detects user context [10] and the talking Braille [11]. Our work aims the same goal. Our system is adaptive to user's condition and environment. Through pervasive computing networks, the ML knowledge, and user's profile and task all become omnipresent; our system's user task configuration is generated without any human intervention.

3 Building a ML Knowledge for Configuration Optimization

3.1 Machine Learning Training to Build User Preferences

A *task* is a computing work the user needs to do. To accomplish the task, the user runs one or more computing applications. For example, a user wishing to shop for a second-hand car may access a web browser, a text editor and a video player.

Given a filename (*filename*.extension), the first function to be learned, $\mathbf{f_1}$, is a mapping of a data type to an application (**$\mathbf{f_1}$: data format → Application**). A diagram showing the learning process is shown in Fig. 1. Each mapping is given a score of *H* (high), *L* (low) or *I* (inappropriate). For example, the mapping (.doc, Text Editor) gets H, (.doc, Web Browser) has an L, and (.doc, Video Player) gets an I. The knowledge obtained from this mapping contains a set of data format and application mappings whose scores are H. The following is a sample set of mappings of $\mathbf{f_1}$:

f_1 = {(.txt, Text Editor), (.doc, Text Editor), (.rtf, Text Editor), (.html, Web Browser), (.xml, Web Browser), (.wav, Audio/Video Player), (.mp3, Audio/Video Player), (.mpg, Audio/Video Player), etc.}

An application may have several suppliers. Another function to be learned, $\mathbf{f_2}$, maps an application to the user's preferred supplier (**$\mathbf{f_2}$: Application → Preferred supplier, Priority**). For simplicity purposes, we assume that the user chooses his 3 preferred suppliers and ranks them by priority. The learned function is saved onto KD and is called user supplier preference. The following is a sample content of $\mathbf{f_2}$:

f_2 = {(Text Editor, (MSWord, 1)), (Text Editor, (WordPad, 2)), (Text Editor, (NotePad, 3)), (Web Browser, (Internet Explorer, 1)), (Web Browser, (Netscape, 2)), (Web Browser, (BrailleSurf, 3)), (Audio/Video Player, (Windows Media Player, 1)), (Audio/Video Player, (Real One Player, 2)), etc.}

An application has its *quality of service* (QoS) dimensions that consumes computing resources. Here, the only important QoS dimensions are those that are valuable to blind users. A function $\mathbf{f_3}$ maps an application and its QoS dimensions that the user prefers (**$\mathbf{f_3}$: Application *i* → QoS dimension *j*, Priority**) where $1 \leq i \leq$ app_max (max. no. of applications) and Application $i \in$ user task. Also, $1 \leq j \leq$ qos_max (max. no. of QoS dimensions) and QoS dimension $j \in$ Application *i*. Priority is of type $\mathbb{N}_1$. Since there are many possible values for each QoS dimension, the user arranges these values by their priority ranking. A sample $\mathbf{f_3}$ is given below:

f_3 = {(Text Editor, (40 characters per line, 1)), (Text Editor, (60 characters per line, 2)), (Text Editor, (80 characters per line, 3)), (Web Browser, (medium page loading, 1)), (Web Browser, (high page loading, 2)), (Web Browser, (low page loading, 3)), (Audio/Video Player, (medium volume, 1)), etc.}

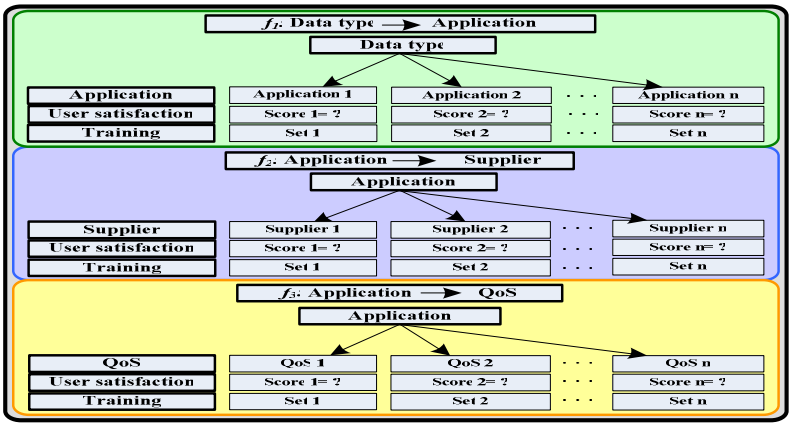

Fig. 1. The training process: (Top) the mapping of data type to an application, (Middle) building a user's preferred supplier list, and (Bottom) building a user's preferred QoS dimensions

3.2 Alternative Configuration Spaces

Given a user task, one or more applications are instantiated. For an application, however, there are some suppliers and QoS dimensions selections that can be invoked. Respecting the user's preferences is the way to instantiate an application, but if it is not possible, the dynamic reconfiguration mechanism must look upon the various configuration spaces and determine the one that is feasible to the user's needs. Fig. 2 shows typical invoked applications for a computing task of a blind user. Note the data type, the suppliers and the QoS dimensions that are mapped with an application. Also shown are the modalities and media that are invoked. The modalities abbreviations are as follows: SP_{in}=Speech input, SP_{out}=Speech output, T_{in}= Tactile input and T_{out} = Tactile output. For media devices, MIC=microphone, KB=keyboard, OKB= overlay keyboard, SPK=speaker, HST=headset, BRT = Braille terminal, TPR = tactile printer.

Application	*Data Type*	*Supplier*	*Selected QoS Dimensions*	*Modality*	*Media*
Text Editor	<filename> txt <filename> doc <filename> rtf		40 Characters per line 60 Characters per line 80 Characters per line	SP_{In}, T_{In} SP_{Out}, T_{Out}	MIC KB OKB SPK HST BRT TPR
Web Browser	<filename> htm <filename> html <filename> xml		Medium page loading High page loading Low page loading	SP_{In}, T_{In} SP_{Out}, T_{Out}	MIC KB OKB SPK HST BRT TPR
Audio/V Player	<filename> wav <filename> mp3 <filename> wmv		Medium volume High volume Low volume	SP_{In}, T_{In} SP_{Out}	MIC KB OKB SPK HST BRT TPR

Fig. 2. The user task as a collection of applications; instantiation of application is based on supplier and QoS dimension preferences. Needed modalities and media are also shown.

In the next sections, the following logic symbols appear: $\otimes$ = Cartesian product yielding a set composed of tuples, the basic logical connectives $\wedge$ (AND) and $\vee$ (OR), and $(a, b]$ denotes that a valid data is higher than a and up to a maximum of b.

A *QoS dimension* is an application's parameter that consumes *computing resources* (battery, CPU, memory, bandwidth). As an application's QoS dimension improves, then the application's quality of presentation (e.g. sound, crispiness of images, etc.) also improves but at the expense of larger resources' consumption. Given a task that is implemented by various applications, the task's QoS dimension space is given by:

$$\text{QoS Dimension space} = \otimes_{i=1}^{qos\_max} D_i \tag{1}$$

In this work, the QoS dimensions that matter are those that are valuable to the blind, namely: *the character per line* (for Text editor and Web browser), the *volume* (for Video and Audio player), and *page loading latency* (for Web browser). Given two applications *s* and *t*, their dimension space is $D_i(s) \otimes D_i(t)$.

The supplier's space, given below, denotes all possible applications' suppliers combinations for user's task, given that every application *i* has its own set of suppliers.

$$\text{Supplier space} = \otimes_{i=1}^{app\_max} Supp_i \tag{2}$$

3.3 Optimizing Configuration of User's Applications

A *feasible configuration* is a set-up that tries to satisfy the user's preferences given the user's context, and the resources' constraints. When the configuration is feasible, it is said that the user's satisfaction is achieved. Let the *user's satisfaction* to an outcome be within the *Satisfaction space*. It is in the interval of [0, 1] in which 0 means the outcome is totally unacceptable while a 1 corresponds to a user's satisfaction. Whenever possible, the system strives to achieve an outcome that is closer to 1.

Given an application, the user's satisfaction is enhanced if his preferences are enforced. The supplier preferences in instantiating an application are given by:

$$\text{Supplier preferences} = h_s^{x_s} \bullet f_s\, c_s \tag{3}$$

where $s \in$ Supplier space is an application supplier and the term $c_s \in [0, 1]$ reflects how the user cares about supplier *s*. Given an application, if it has *n* suppliers which are arranged in order of user's preference, then $c_{supplier1} = 1$, $c_{supplier2} = 1 - 1/n$, $c_{supplier3} = 1 - 1/n - 1/n$, and so on. The last supplier therefore has c_s close to zero which means that the user cares not to have it if given a choice. In general, in each application, the c_s assigned to supplier i, $1 \le i \le n$, is given by:

$$c_{supplier\,i} = 1 - \sum_{1}^{i-1}(1/n) \tag{4}$$

The term f_s: dom(s)→[0,1] denotes the expected features present in supplier *s*. The supplier *features* are those that are important to the user, other than the QoS dimensions. For example, in a text editor application, the user might prefer a supplier that provides *spelling and grammar checking*, or *equation editor* or feature to *build a table*,

etc. For example, if the user listed $n = 3$ preferred features for an application, and the selected supplier supports them, then $f_s = 1$. If, however, one of these features is missing (either because the feature is not installed or the supplier does not have such feature), then the number of *missing* feature $m = 1$ and $f_s = 1 - m/(n + 1) = 1 - ¼ = 0.75$. In general, the user satisfaction with respect to application features is given by:

$$f_{supplier} = 1 - \frac{m}{n+1} \tag{5}$$

The term $h_s^{x_s}$ expresses the user's satisfaction with respect to the change of the supplier, and is specified as follows: $h_s \in (0, 1]$ is the user's tolerance for a change in the supplier. If this value is close to 1, then the user is fine with the change while a value close to 0 means the user is not happy with the change. The optimized value of h_s is:

$$h_s = \arg\max (c_s + c_{rep})/2 * c_s \tag{6}$$

where c_{rep} is a value obtained from equation (4) for replacement supplier. x_s indicates if change penalty must be considered. $x_s = 1$ if the supplier exchange is due to the dynamic change of environment, while $x_s = 0$ if the exchange is instigated by the user.

Similarly, a user's preferences for QoS dimensions of his applications as given by:

$$\text{QoS preferences} = h_q^{x_q} \bullet c_q \tag{7}$$

where and q ∈ QoS dimension space is a QoS dimension of an application. Note that equations (3) and (7) are almost identical except for the differences in the subscripts and the absence of feature in QoS dimensions. The algorithms for finding the optimized QoS and supplier configuration of any application are given in Fig. 3. In each algorithm, the default configuration is compared with other possible configurations until the one yielding the maximum value of user's satisfaction is found and is returned as result of each algorithm. A feasible configuration is achieved if the user's

```
Optimize_QoS
Input
        1  Application a  current QoS c_curren
        2  All QoS q 1 ≤ i ≤ n for an application a
Outpu  Best_QoS for application a
Procedure
1       get c_curren of c_curren
2       Preference = c_curren
3       Max = Preference
4       Best_QoS = q_curren
5       for each QoS q dc
6         if c_curren is replaced with q by dynamic configuration
7           then X=1
8         else X=0
9         endif
10      get c of QoS replacement q
1       h =(c_curren + c)/(2*c_curren)
12      User_Satisfaction = Preference * h ' X
13      if User_Satisfactior > Max then
14        Max= User_Statisfaction
15        Best_QoS = q
16      endif
17 endfor
18 return Best_QoS
Endprocedure
```

```
Optimize_Supplier
Input
        1  Application a  current Supplier S_curren
        2  All suppliers S 1 ≤ i ≤ n for an application a
Output  Best_Supplier for application a
Procedure
1       calculate l_curren and c_curren of supplier S_curren
2       Preference = l_curren * c_curren
3       Max = Preference
4       Best_Supplier = S_curren
5       for each supplier S dc
6         if S_curren is replaced with S by dynamic configuration
7           then X=1
8         else X=0
9         endif
10      get c of replacement supplier S
1       h =(c_curren + c)/(2*c_curren)
12      User_Satisfaction = Preference * h ' X
13      if User_Satisfactior > Max then
14        Max= User_Statisfaction
15        Best_Supplier= S
16      endif
17 endfor
18 return Best_Supplier
Endprocedure
```

Fig. 3. Algorithms for optimized QoS and supplier configuration of an application

task can be realized by appropriate applications that are instantiated using the user's preferred suppliers and QoS dimensions. The feasible configuration is given by:

$$\arg\max_{\substack{app(a)\in task \\ s\in supplier(app(a)) \\ q\in QoS\,dim\,(s)}} = \prod_{a=1}^{app\_max} \text{Supplier preferences (a)} \bullet \text{QoS preferences (a)} \tag{8}$$

The algorithm for finding the feasible configuration of applications within the user's task is shown in Fig. 4. It finds the feasible configuration in every application.

As earlier said, [9] has positively influenced our work. Equations (1), (2), and (8) were taken from such work. Although previously defined in the same reference, Equations (3) and (7) have since evolved that their final forms in this paper have become ours. The rest of the other equations are all ours.

```
Optimize_task
Input
        1  Application a  1 ≤ i ≤ max_app
        2  Current Supplier for each application a
        3  Current QoS for each application a
Output  Appropriate suppliers and QoS for the task
Procedure
for i = 1 to max_app
 optimize_supplier()
 optimize_QoS()
 configuration_a  = Best_QoS * Best_Supplier
endfor
Endprocedure
```

Fig. 4. The algorithm for optimizing user's task configuration

3.4 Realizing User Task Through Appropriate Modalities and Media

Having known the user's task and context, then a feasible modality needs to be found for the computing to proceed. The modalities available to the user are *speech input* (SP_{in}), *speech output* (SP_{out}), *tactile input* (T_{in}) and *tactile output* (T_{out}). At any time, each of these modalities is either active or inactive (on or off). The truth table for all possible combinations of various modalities for the blind is shown in Fig. 5 (Left). A value of *T* (true) means a modality is possible. Hence, successful modality is given by:

$$\text{Modality} = (SP_{in} \vee T_{in}) \wedge (SP_{out} \vee T_{out}) \tag{9}$$

Therefore, a successful modality can be implemented if there is at least one input modality and at least one output modality.

Given the user's task and the applications to realize it, we then determine when a modality is possible or not which, for a blind user, is a function of the user's computing device and the noise level in his workplace. Given that:

Application = {Web Browser, Text Editor, Audio/Video Player}, Modality = {SP_{in}, T_{in}, SP_{out}, T_{out}}
Computing Device = {PC, MAC, Laptop, PDA, Cell phone}, Noise Level = {Quiet/Acceptable, Noisy}

then modality is possible under various parameters' combinations. There exists, however, a system and environment condition where modality is not possible as given by:

$$\text{Modality Failure} = (\text{Computing Device} = (\text{PDA} \vee \text{Cellphone})) \wedge (\text{Noise Level} = \text{Noisy}) \tag{10}$$

In this condition, tactile input and output, and speech input are not possible leaving only speech output as the remaining possible modality. As stated in (9), a modality requires at least one mode for data input and at least one mode for data output. Such restriction is violated in the preceding condition which renders multimodality to fail.

Let there be a function $\mathbf{f_4}$ that maps a specific modality to its appropriate media device(s) as given by $\mathbf{f_4}$**: Modality → Media**. See Fig. 5 (Right). The presence of the necessary media is important if a modality is to be implemented. The function $\mathbf{f_4}$ for visually-impaired users would be similar to the one given below:

$$f_4 = \{(SP_{in}, \text{Microphone}), (SP_{in}, \text{Speech Recognition}), (SP_{out}, \text{Speaker}), (SP_{out}, \text{Headset}), (SP_{out}, \text{Text-to-Speech}), (T_{in}, \text{Keyboard}), (T_{in}, \text{Overlay Keyboard}), (T_{out}, \text{Braille Terminal}), (T_{out}, \text{Tactile Printer})\}$$

Note that although media technically refers to hardware components, a few software elements, however, are included in the list as speech input modality would not be possible without a speech recognition software and the speech output modality cannot be realized without the presence of a text-to-speech translation software. From $\mathbf{f_4}$, we can obtain the relationship in implementing multimodality:

$$f_4(SP_{in}) = \text{Microphone} \wedge \text{Speech Recognition}, \quad f_4(SP_{out}) = (\text{Speaker} \vee \text{Headset}) \wedge \text{Text-to-Speech}$$
$$f_4(T_{in}) = \text{Keyboard} \vee \text{Overlay Keyboard}, \quad f_4(T_{out}) = \text{Braille Terminal} \vee \text{Tactile Printer}$$

Modality	SP_{in} SP_{out}			
T_{in} T_{out}	00	01	11	10
00	F	F	T	F
01	F	F	T	T
11	T	T	T	T
10	F	T	T	F
$Modality = (SP_{in} \vee T_{in}) \wedge (SP_{out} \vee T_{out})$				

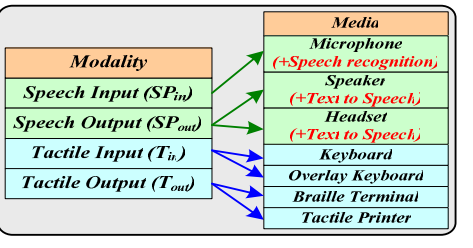

Fig. 5. (Left): The truth table to realize an effective implementation of modality, (Right): Media selections to realize a modality operation

Therefore, the assertion of modality, as expressed in equation (9), with respect to the presence of media devices becomes:

$$\begin{aligned} \text{Modality} = &((\text{Microphone} \wedge \text{Speech Recognition}) \vee (\text{Keyboard} \vee \text{Overlay Keyboard})) \\ &\wedge (((\text{Speaker} \vee \text{Headset}) \wedge \text{Text - to - Speech}) \vee (\text{Braille Terminal} \vee \text{Tactile Printer})) \end{aligned} \tag{11}$$

Therefore, in order to realize a pervasive multimodal multimedia computing, given the constraints that we have considered, it is imperative that equation (10) *should not exist* and equation (11) *should be satisfied*.

The presence of needed media devices does not automatically mean the success of a modality. Why? First, an available media device may not be working at all. Some methods for detecting device failure is available in [12]. Second, it is possible that even if a media device is present and functional, it still cannot be used due to the restriction imposed on the environment (e.g. in a library where “silence is required”, a

functional microphone serves no use at all). Hence, a failure in modality as a function of the media devices failure and environment restriction is given by:

$$\begin{aligned}\text{Modality Failure} = {} & (T_{in} = \text{Failed}) \wedge [(SP_{in} = \text{Failed}) \vee (\text{Environment Restriction} = \text{SilenceRequired})] \vee \\ & \{(T_{out} = \text{Failed}) \wedge [(SP_{out} = \text{Failed}) \vee (\text{HST} = \text{Failed}) \wedge (\text{Environment Restriction} = \text{SilenceRequired})]\}\end{aligned} \quad (12)$$

4 Design Specification and Scenario Simulations

Having formulated various ML knowledge to optimize the configuration setting of user's task, this knowledge is then put to test via sample scenarios. The design specification comes along as these scenarios are further explained.

4.1 Specification for User's Task

Consider a student user who wishes to do his homework which compares the works of two great composers, *Beethoven* and *Mozart*. To do so, our user needs access to a web browser, a text editor and a video player. Our user would work on his homework at home using his personal computer. The following day, he may continue working on his task in the school's library. In this case,

f_1 = {(assignment1.doc, Text Editor), (www.classicalmusic.com/Beethoven.html, Web Browser), (www.classicalmusic.com/Mozart.html, Web Browser), (beethoven1.wav, Audio/Video Player), (beethoven2.wav, Audio/Video Player), (mozart1.wav, Audio/Video Player), etc.}.

Formally, $\forall \times$: data format, $\exists$ y: Application | x$\rightarrow$y $\in$ $\mathbf{f_1}$. Consider our user being in the school library working on his task using a laptop. After logging in, our system determines the applications that are suitable to the data format of the latest files in his task folder. This is done with reference to function $\mathbf{f_1}$. Using $\mathbf{f_2}$, the system determines the supplier for each application. Since a supplier priority is involved in $\mathbf{f_2}$ then the most-preferred supplier is sought. Fig. 6 shows a sample tabulation of user's preferences. Using equation (4), the following are the numerical values for user preferences: (i) if Priority= 1 (High), then User Satisfaction= 1,(ii) if Priority= 2 (Medium), then User Satisfaction= 2/3, and (iii) if Priority= 3 (Low), then User Satisfaction= 1/3.

Consider a case wherein the user's preferred audio/video player supplier – the Windows Media Player – is absent as it is not available in the user's laptop. The method by which the system finds the feasible supplier configuration is shown below:

Case 1: (MSWord, Internet Explorer, Windows Media Player) → not possible,
Case 2: (MSWord, Internet Explorer, Real One Player) → alternative 1
Case 3: (MSWord, Internet Explorer, JetAudio) → alternative 2

then the feasible selection is based on user satisfaction score:

User Satisfaction: Case 2 = (1 + 1 + 2/3)/3 = 8/9 = 0.89, and Case 3 = (1 + 1 + 1/3)/3 = 7/9 = 0.78

Hence, Case 2 is the preferred alternative. Formally, if **$\mathbf{f_2}$: Application → (Supplier, Priority)** where Priority: $\mathbb{N}_1$, then the chosen supplier is given by: $\exists$ x: Application, $\forall$ y: Supplier, $\exists$ p1: Priority, $\forall$ p2: Priority | y ● x$\rightarrow$(y, p1) $\in$ $\mathbf{f_2} \wedge$ (p1 < p2).

Application	Supplier & Priority		QoS & Priority	
Text Editor	MSWord	1	40 Characters per line	1
	Wordpad	2	60 Characters per line	2
	Notepad	3	80 Characters per line	3
Web Browser	Internet Explorer	1	Medium page loading	1
	Netscape	2	High page loading	2
	BrailleSurf	3	Low page loading	3
Audio Player	Windows Media Player	1	Medium volume	1
	Real One Player	2	High volume	2
	JetAudio	3	Low volume	3

Fig. 6. Tabulation of user's preferences (Supplier and QoS) and their priority rankings

4.2 Optimizing User's Task Configuration

Consider a scenario where all suppliers for an application are available. For example, for a Text Editor application, the amount of user satisfaction with different suppliers would be like:

$$c_{MSWord} = 1.0,\ c_{Wordpad} = 2/3\ ,\ c_{Notepad} = 1/3$$

This indicates that the user is most happy with the top-ranked supplier (MSWord) and least happy with the bottom-ranked supplier (Notepad). In an MSWord set-up, if an equation editor, for example, is not installed, the user's satisfaction decreases, as given by the relationship $c_{MSWord} * f_{MSWord} = (1.0)(0.75) = 0.75$.

Now, consider a case of a dynamic reconfiguration wherein the default supplier is to be replaced by another. Not taking f_s into account yet (assumption: $f_s = 1$), if the current supplier is WordPad, then the user's satisfaction is $c_{Wordpad} = 2/3 = 0.67$. What would happen if it will be replaced by another text editing supplier through dynamic reconfiguration ($x_{supplier} = 1.0$)? Using the relationship $h_{supplier} = (c_{supplier} + c_{replacement}) / 2 * c_{supplier}$ then the results of possible alternative configurations are as follows:

Replacing WordPad (supplier 2):
Case 1: Replacement by MSWord (supplier 1): $(0.67)(1) * [(0.67 + 1)/2*(0.67)]^1 = 0.835$
Case 2: Replacement by itself (supplier 2): $(0.67)(1) * [(0.67 + 0.67)/2*(0.67)]^1 = 0.67$
Case 3: Replacement by Notepad (supplier 3): $(0.67)(1) * [(0.67 + 0.33)/2*(0.67)]^1 = 0.50$

Hence, if the reconfiguration aims at satisfying the user, then the second-ranked supplier should be replaced by the top-ranked supplier.

In a similar fashion, the QoS dimensions are given the same scores for their priority ranking. With characters per line as QoS parameter in a text editor application, then

$$c_{40\ characters\ per\ line} = 1.0,\ c_{60\ characters\ per\ line} = 0.67,\ c_{80\ characters\ per\ line} = 0.33$$

Indeed, the feasible configuration for a text editor application is given by:

$$\arg\max_{Text\ Editor} = (f_{MSWord}{}^{c_{MSWord}})(f_{40\ characters\ per\ line}) = 1.0$$

4.3 Specification for Detecting Suitability of Modality

Petri Net [13] is a formal, graphical, executable technique for the specification and analysis of a concurrent, discrete-event dynamic system. Petri nets are used in deterministic and in probabilistic variants; they are a good means to model concurrent or

collaborating systems. They also allow for different qualitative or quantitative analysis that can be useful in safety validation. In the specifications in this paper, only a snapshot of one of the many outcomes is presented due to space constraints. We use HPSim [14] in simulating Petri Net.

In Fig. 7, a Petri Net specification is shown with user's task, modalities, computing device, and the workplace's noise level as inputs to user's condition. As shown, many of these combinations render the modality possible. There is, however, a condition that makes modality and therefore computing for the blind user fail, and that is when the user's computing device is either a PDA or a cellular phone and his workplace is noisy. This is given in equation (10) and is traceable in the Petri Net diagram.

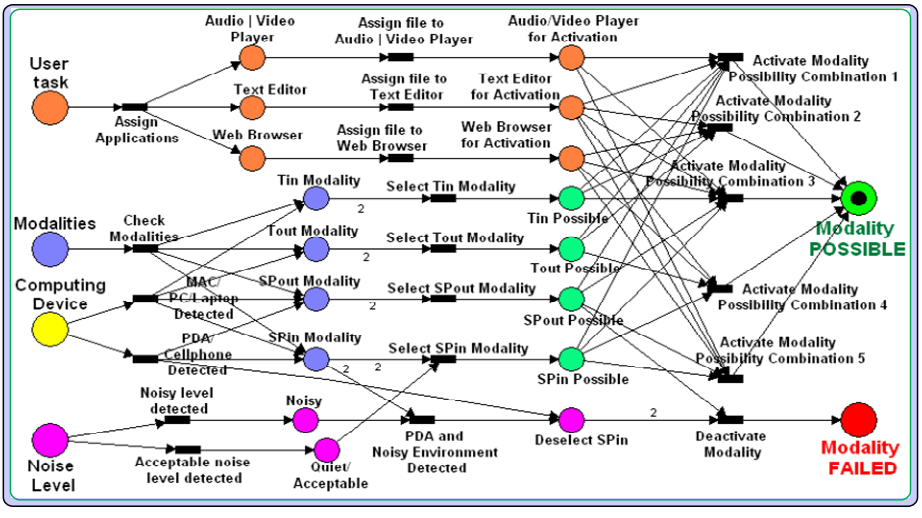

Fig. 7. Petri Net diagram showing the possibility or failure of modality based on the environment's noise level, the computing device and user's task

4.4 Experimental Results

Using user's preferences, we have simulated the variations in user's satisfaction as these preferences are modified through dynamic configuration. The results are presented through various graphs in Fig. 8. The first two graphs deal with application supplier, and the variation of user's satisfaction as additional parameters are taken into account, supplier features and alternative replacements being the parameters. The last one deals with QoS dimensions and their variations.

Graph (a) shows that user's satisfaction does not only rely on a supplier's ranking but also on its features. For example, supplier 2 of no missing feature satisfies the user better than supplier 1 that has 2 missing features. In addition, the result in graph (b) illustrates such satisfaction as a function of current supplier and its features and its alternative supplier replacement. Similarly, the QoS dimension is not only a function of priority but also of its alternative replacements, as shown in graph (c). In general,

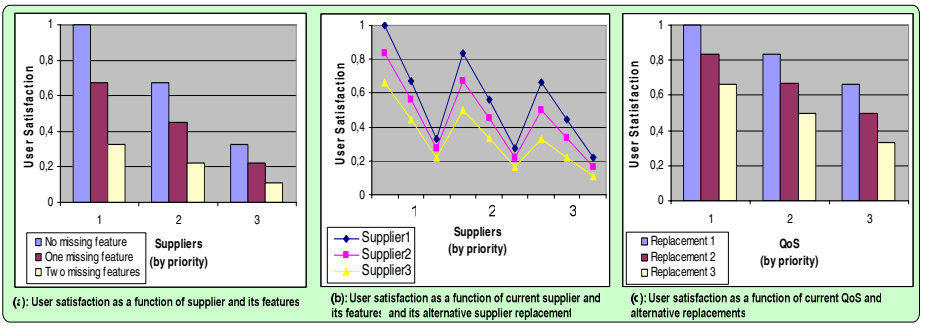

Fig. 8. Various graphs showing variations of user's satisfaction with respect to its preferred supplier and QoS dimension and their replacements

user is satisfied if the supplier and its desired features and QoS dimensions are provided. Whenever possible, in a dynamic configuration, the preferred setting result is one in which the set-up parameters are those among the user's top preferences.

5 Conclusion

This paper presented the methodology for a successful migration and execution of user's task in a pervasive MM computing system. Through ML training, we illustrated the acquisition of positive examples to form user's preferred suppliers and QoS dimensions for selected applications. In a rich computing environment, alternative configuration spaces are possible which give the user some choices for configuring the set-ups of some of his applications. We have illustrated that configuration could be dynamic or user-invoked, and the consequences, with respect to user's satisfaction, of these possible configurations. Optimization is achieved if the system is able to configure set-up based on user's preferences.

In this work, we have listed various modalities that are available to a blind user. A modality is possible if there is at least one mode for data input and also at least one mode for data output. Given sets of applications, modalities, computing devices, and environment's noise level, we formulated the conditions where modality would succeed and fail. Similarly, we illustrated the scenario wherein even if a specific modality is already deemed possible, still it is conceivable that modality would fail if there are not sufficient media devices that would support it or the environment restriction imposes the non-use of the needed media devices. We validated all these affirmations through various scenario simulations and formal specifications.

Future works include performance details of user task configurations as simulated on various types of processors and software platforms. This would include evaluation of dynamic configuration performance as the system searches alternative application supplier and QoS dimensions.

References

[1] M. Satyanarayanan, "*Pervasive Computing: Vision and Challenges*", *IEEE Personal Communications*, vol. 8, pp. 10-17, 2001.

[2] A. Awdé, et al, "A Paradigm of a Pervasive Multimodal Multimedia Computing System for the Visually-Impaired Users", GPC 2006, 1st International Conference on Grid and Pervasive Computing, Tunghai University, Taichung, Taiwan, 2006.

[3] H. Ferreira, D. Freitas, "Enhancing the Accessibility of Mathematics for Blind People: The AudioMath Project", ICCHP, 2004, pp. 678-685.

[4] V. Moco, D. Archambault, "Automatic Conversions of Mathematical Braille: A Survey of Main Difficulties in Different Languages", ICCHP *Conference*. Paris, France, 2004.

[5] A. Solon, et al, "Mobile multimodal presentation", New York, NY, USA, 2004.

[6] A. Lécuyer, et al, "HOMERE: a Multimodal System for Visually Impaired People to Explore Virtual Environments", Proc. IEEE Virtual Reality, Washington, USA, 2003.

[7] C. Rousseau, et al, "A Framework for the Intelligent Multimodal Presentation of Information", *Signal Processing Journal*, vol. 86, pp. 3696 - 3713, 2006.

[8] A. Demeure, et al, "Le Modèle d'Evolution en Plasticité des Interfaces: Apport des Graphes Conceptuels", 15ème Conf. francophone sur l'Interaction Homme-Machine (IHM 2003), Caen, France, 2006.

[9] V. Poladian, et al. "Task-based Adaptation for Ubiquitous Computing", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 36, pp. 328 - 340, 2006.

[10] C. Jacquet, et al, "A Context-Aware Locomotion Assistance Device for the Blind", ICCHP 2004, 9th Intl. Conf. on Comp. Helping People with Special Needs, Paris, France.

[11] D. A. Ross, et al, "Talking Braille: A Wireless Ubiquitous Computing Network for Orientation and Wayfinding", 7th Intl. ACM SIGACCESS Conf. on Comp., MD, USA 2005.

[12] M. D. Hina, et al, "Design of an Incremental Learning Component of a Ubiquitous Multimodal Multimedia Computing System", WiMob 2006, 2nd IEEE Intl. Conf. on Wireless and Mobile Computing, Networking and Communications, Montreal, Canada, 2006.

[13] "Petri Net", http://www.winpesim.de/petrinet/

[14] "HPSim", http://www.winpesim.de/