2

# A Multi-Agent based Multimodal System Adaptive to the User's Interaction Context

Manolo Dulva Hina[1,2], Chakib Tadj[1],
Amar Ramdane-Cherif[2] and Nicole Levy[2]
[1]*Université du Québec, École de technologie supérieure,*
[2]*Université de Versailles-Saint-Quentin-en-Yvelines,*
[1]*Canada*
[2]*France*

## 1. Introduction

*Communication* is an important aspect of human life; it is with communication that helps human beings connect with each other as individuals and as independent groups. In informatics, the very purpose of the existence of computer is *information dissemination* – to be able to send and receive information. Humans are quite successful in conveying ideas with one another and reacting appropriately because we share the richness of our language, have a common understanding of how things work and have an implicit understanding of everyday situations. When human communicate with human, they comprehend the information that is apparent to the current situation, or *context*, hence increasing the conversational bandwidth. This ability to convey ideas, however, does not transfer when human interacts with computer. On its own, computers do not understand our language, do not understand how the world works and cannot sense information about the current situation. In a typical impoverished computing set-up where providing computer with information is through the use of mouse, keyboard and screen, the result is we explicitly provide information to computers, producing an effect that is contrary to the promise of *transparency* and *calm technology* in Marc Weiser's vision of *ubiquitous computing* (Weiser 1991; Weiser 1993; Weiser and Brown 1996). To reverse this, it is imperative that methodologies are developed that will enable computers to have access to context. It is through *context-awareness* that we can increase the richness of communication in human-computer interaction, through which we can reap the most likely benefit of more useful computational services.

*Context* (Dey and Abowd 1999; Gwizdka 2000; Dey 2001; Coutaz, Crowley et al. 2005) is a subjective idea and its interpretation is personal. Context evolves and the acquisition of contextual information is essential. However, we believe that the one with the final word on whether the envisioned context is correctly captured/acquired or not is the *end user*. Current research works indicate that some contextual information are already predefined by their systems from the very beginning – this is correct if the application domain is fixed but is incorrect if we infer that a typical user does different computing tasks in different occasions. With the aim of coming up with more conclusive and inclusive design, we conjure that the contextual information that is important to the user should be left to the judgment of the end

user. This leads us to the *incremental* acquisition of context where context parameters are added, modified or deleted one context parameter at a time.

In conjunction with the idea of inclusive context, we enlarge the notion of context that it has become interaction context. *Interaction context* refers to the collective context of the user (i.e. user context), of his working environment (i.e. environmental context) and of his computing system (i.e. system context). Each of these interaction context elements – *user context*, *environmental context* and *system context* – is composed of various parameters that describe the state of the user, of his workplace and his computing resources as he undertakes an activity in accomplishing his computing task, and each of these parameters may evolve over time. For example, user location is a user context parameter and its value will evolve as the user moves from one place to another. The same can be said about noise level as an environment context parameter; its value evolves over time. This also applies to the available bandwidth, which continuously evolves, which we consider as a system context parameter.

The evolution of the interaction context, from the time the user starts working on his computing task up to its completion, informs us that the contextual information changes instantaneously, and as such the computing system needs to adapt appropriately. Too often, a regular computing system remains static – it does nothing – even in the eventuality that a certain interaction context parameter changes that the user has no option but to intervene. For instance, when the bandwidth becomes too limited, downloading data becomes too slow that the user needs to intervene to stop the software application. This is the challenge of our time – **how can we design a computing system that adapts appropriately to the constantly evolving interaction context?** Our review of the state-of-the-art indicates that in the existing context-sensitive applications, very large efforts were expended by researchers in defining how to capture context and then disseminate it to the system. And yet, precise answer is still missing as to how the application itself will adapt to the given context. It is in this last direction that this chapter work registers.

The remaining contents of this chapter are as follows. Section 2 focuses on the review of the state-of-the-art; it tells us what has been done by other researchers in this domain and what is missing or lacking in the current endeavors. Section 3 introduces us to agents and the multi-agent system that will attempt to provide solutions to the cited problem. Section 4 is concentrated on modalities and the multimodal computing system. The multi-agent system's adaptation to interaction context is the main focus of Chapter 5. This work is concluded in Chapter 6.

## 2. Review of the state-of-the-art

The term "*context*" comes in many flavours, depending on which researcher is talking. In Shilit's early research, (Schilit and Theimer 1994), context means the answers to the questions "*Where are you?*", "*With whom are you?*", and "*Which resources are in proximity with you?*" He defined context as the changes in the physical, user and computational environments. This idea is taken later by Pascoe (Pascoe 1998) and Dey (Dey, Salber et al. 1999). Brown considered context as "*the user's location, the identity of the people surrounding the user, as well as the time, the season, the temperature, etc.*" (Brown, Bovey et al. 1997). Ryan defined context as the environment, the identity and location of the user as well as the time involved (Ryan, Pascoe et al. 1997). Ward viewed context as the possible environment states of an application (Ward, Jones et al. 1997). In Pascoe's definition, he added the pertinence of

the notion of state: "*Context is a subset of physical and conceptual states having an interest to a particular entity*". Dey specified the notion of an entity: "*Context is any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and application themselves*" (Dey 2001). This definition became the basis for Rey and Coutaz to coin the term interaction context: "*Interaction context is a combination of situations. Given a user U engaged in an activity A, then the interaction context at time t is the composition of situations between time* $t_0$ *and t in the conduct of A by U*" (Rey and Coutaz 2004).

The efforts made in defining context within the domain of context awareness were in fact attempts in formalism, as in the case of definition proposed in (Abowd and Mynatt 2000). Other researchers, not satisfied with general definitions, attempted to define context formally (Chen and Kotz 2000; Dey 2001; Prekop and Burnett 2003). Pascoe (Pascoe 1998) and Dey (Dey and Abowd 1999) brought more precision in context definition by specifying that context is a set of information that describes an entity.

In other works related to sensitivity to context, various researchers started resolving the issue concerning the user's mobility. Then, research deepens within their emphasis on the whereabouts of the user. For example, *Teleporting* (Bennett, Richardson et al. 1994) and *Active Map* (Schilit and Theimer 1994) are few works on applications that are sensitive to the geographic location of a user. Dey (Dey 2001) and Chen and Kotz (Chen and Kotz 2000) made constraints on context research by putting emphasis on applications, the contextual information that is being used and their use. Gwizdka (Gwizdka 2000) identified two categories of context: internal and external. The categorization, however, was done with respect to the user's status. Dey and Abowd (Dey and Abowd 1999) and even Schilit (Schilit, Adams et al. 1994) categorize contextual information by levels. In the case of Dey's work, the primary level contains information that is related to the user's location, activity and time whereas with Schilit, the primary level refers to the user's environment, the physical environment and the computing environment. One more time, the contextual information considered in these categorizations did not sufficiently take environment context in depth. To respond to the problems raised in the previous categorizations, Razzaque (Razzaque, Dobson et al. 2005) proposed a finer categorization of contextual information. Dey's Context Toolkit (Dey, Salber et al. 2001) is one of the first architectures which considered three (3) important steps in works on context sensitivity (that is, the capture, representation and exploitation of context). In this architecture, the modeling of context uses an approach called sets of pairs of (entity, attribute).

Other approaches in context representation used *RDF* (Resource Description Framework) which is an extension of *W3C CC/PP* (World Wide Web Consortium Composite Capabilities/Preferences Profile) as in the work proposed by (Held 2002) and (Indulska, Robinson et al. 2003). *Ontology* was also used in context modeling in which approach context is considered as a set of entities having aspects describing its characteristics (Strang and Linnhoff-Popien 2003).

After modeling and storage, context needs to be disseminated to the application. Here, we draw our attention to the conceptual platforms of the architectural aspects of systems that are sensitive to context (Dey, Salber et al. 2001; Kindberg and Barton 2001). The works of (Indulska, Loke et al. 2001) and (Efstratiou, Cheverst et al. 2001) present service platforms related to providing necessary services to the user based on a given context. The interoperability environments dealing with the resolution of problem related to heterogeneity and mobility of a user are presented in (DeVaul and Pentland 2000) and

(Eustice, Lehman et al. 1999). Other works were oriented towards the development of distributed applications which deals with the conception of physical and logical infrastructure in developing distributed systems as in the case of works presented in (Banavar, Beck et al. 2000) and (Esler, Hightower et al. 1999). After the publication of the work of Weiser on distributed information systems (Weiser 1993), various works on context sensitivity in this genre of application has allowed the development of *ParcTab* (Schilit, Adams et al. 1993; Want, Schilit et al. 1995), *Mpad* (Kantarjiev, Demers et al. 1993), *LiveBoard* (Elrod, Bruce et al. 1992) and other interesting works (Dey 2001; Kephart and Chess 2001). The *Active Badge* project (Want, Hopper et al. 1992) of Olivetti Research and the *InfoPad* project (Truman, Pering et al. 1998) of Berkeley also embraced this axis of research on distributed computing, as in the case of other various centers of excellence, such as the *Carnegie Mellon University* (CMU_CS 2010), *IBM* (Horn 2001; Kephart and Chess 2001) and *Rutgers* (CS_Rutgers 2010), just to cite a few. We also note the works of (Kantarjiev, Demers et al. 1993), (Want, Schilit et al. 1995) and (Garlan, Siewiorek et al. 2002) which are some of the contributions in the research on adaptations of distributed applications on based on the given context. Also, an important work on the taxonomies of input devices include that of (Buxton 1983).

In perspective, in existing context-sensitive applications, very large efforts were expended by researchers in defining how to capture context and then disseminate it to the system. And yet, precise answer is still missing as to how the application itself will adapt to the given context. It is in this last direction that this work registers. Towards this end, we will present our proposed agents-based computing system that adapts accordingly based on the evolution of the interaction context.

Also, to obtain the full benefit of the richness of interaction context with regards to communication in human-machine interaction, the modality of interaction should not be limited to the traditional use of mouse-keyboard-screen alone. *Multimodality* (Dong, Xiao et al. 2000; Oviatt 2002; Ringland and Scahill 2003) allows for a much wider range of modes and forms of communication, selected and adapted to suit the given user's interaction context, by which the end user can transmit data with computer and computer responding or yielding results to the user's queries. In multimodal communication, the weaknesses of one mode of interaction, with regards to its suitability to a given situation, is compensated by replacing it with another mode of communication that is more suitable to the situation. For example, when the environment becomes disturbingly noisy, using voice may not be the ideal mode to input data; instead, the user may opt for transmitting text or visual information. Multimodality also promotes inclusive informatics as those with permanent or temporary disability are given the opportunity to use and benefit from information technology advancement. With mobile computing within our midst coupled with wireless communication that allows access to information and services, pervasive and adaptive multimodality is more than ever apt to enrich communication in human-computer interaction and in providing the most suitable modes for data input and output in relation to the evolving interaction context. A look back at recent research works inform us that much research efforts on ubiquitous computing were devoted on various application domains (e.g. identifying the user whereabouts, identifying services and tools, etc.) but there is barely, if ever, an effort made to make multimodality pervasive and accessible to various user situations. In this regard, this work fills the gap.

The solution that will be presented in this chapter is different from the rest of previous works in the sense that while others capture, disseminate and consume context to suit its

preferred domain of application, this new multi-agent system captures interaction context and reconfigure its system architecture dynamically with the aim of providing the user with an infrastructure that will enable the user continue working on his task anytime, anywhere. The multi-agent system that will be presented, along with all of its mechanisms being generic in design, can be adapted/integrated into various computing systems in different domains of applications with ease or little amount of modification.

## 3. Agents, multi-agent system and interaction context

Our goal is to design a paradigm of a multimodal multimedia computing system capable of undergoing dynamic reconfiguration based on the given user's interaction context. To this end, we propose automation solution which will reinforce the system's adaptability to the user's situation as well as to support system decision in general and multimodal multimedia in particular. The proposed solution is an automated mechanism for the selection of modalities and supporting media devices that suit the given interaction context. This pertains to finding optimal configuration and quantifying it. The diagram demonstrating these proposed solutions is shown below (see Fig. 1).

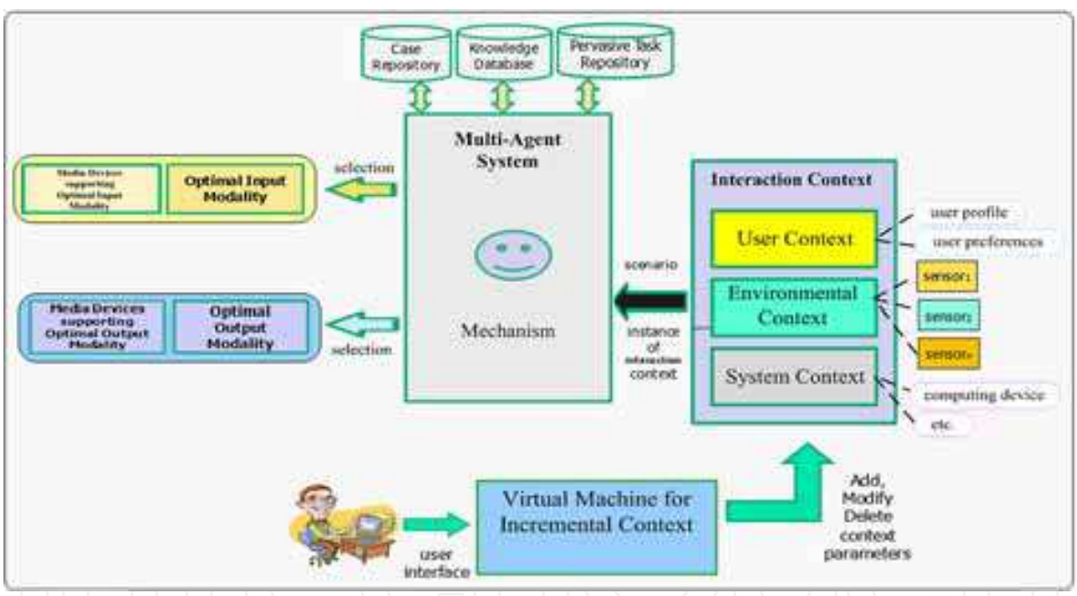



Fig. 1. The overall structure of our proposed multimodal multimedia computing system

In the proposed solution, the selection of optimal configuration is based on a compromise in which we take into account the constraints related to the user, his material environment, software and other factors. This contextual information represents the interaction context of the user. This context is the combination of situations that exist while the user undertakes an activity. These situations are real-time, existing from the time the user starts working on a task up to the time of its completion. During the execution of this activity, some situations remain stable while others change or evolve. Briefly, the interaction context is made up of the context of the user, of his environment and of his computing system. A change in the interaction context may result in the modification of appropriate modalities (and therefore of the media devices that support the modalities). We examined how an ever changing

interaction context affects the stability of the multimodal multimedia computing system so as it will continue providing services to the user.
The discussion that follows discusses the architectural framework and the design of the mechanism of the system's adaptation to the given interaction context.

### 3.1 The multi-agent system's architectural framework

The structure presented above (Fig. 1) is to be redrawn to make it a multi-agent system. The rationale behind this is due to the fact that given that the proposed mechanism is complex, it is preferable that the overall task is broken down into smaller pieces of works, and each work is to be delegated to an *agent*. **Why adapt agency?** The various tasks listed below are complicated and implementing them by just using objects or functions will not suffice. On the other hand, *agents* are *persistent*, *autonomous*, *social* and *reactive* entities, capable of learning, that they suit as **solutions** to implement our proposed solution.

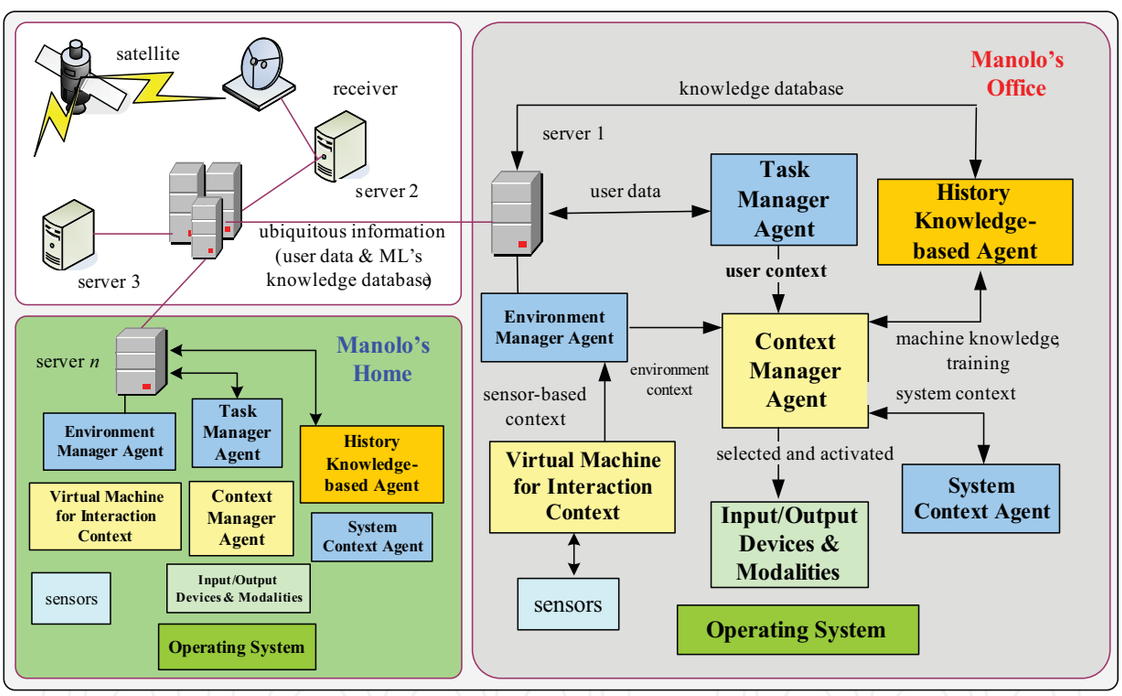



Fig. 2. Architecture of interaction context-sensitive pervasive multimodal multimedia computing system

The resulting multi-agent based multimodal system adaptive to the user's interaction context is shown in Fig. 2. The overall task is now distributed as smaller tasks to different agents. The components of this multi-agent system and their tasks are as follows:

- **The Task Manager Agent** (TMA) – manages user's profile, task and related data and their deployment from a server to the user's computing device, and vice versa.
- **The Context Manager Agent** (CMA) – detects interaction context taken from sensors and user profile, environment and computing system and select the modality and its supporting media devices that are most suitable to the given context.
- **The History and Knowledge-based Agent** (HKA) – responsible for machine learning training and knowledge acquisition.

- **The Virtual Machine for Interaction Context** (VMIC) – detects sensor-based context and allows the incremental definition of context by considering one context parameter at a time.
- **The Environmental Manager Agent** (EMA) – detects available and functional media devices in the user's environment.
- **The System Context Agent** (SCA) – detects the status of available computing devices and computing resources (e.g. bandwidth, CPU, memory and battery).

As shown in the diagram, a user (i.e. Manolo) may work at home, logs off and later on reconnects to a computing device in order to continue working on an interrupted task whenever and wherever he wishes. Due to user's mobility, there are variations in the user's interaction context as well as on available resources; these variations are compensated by corresponding variations in the selection of modalities and activation of supporting media devices.

As shown in Fig. 3, different parameters make up the interaction context. The User Context Agent detects the user's context; the Environment Context Agent in coordination with VMIC agent detects the context of the user's environment and the System Context Agent detects the available computing resources. All these parameters are consolidated and form the *overall interaction context* at that particular instance.

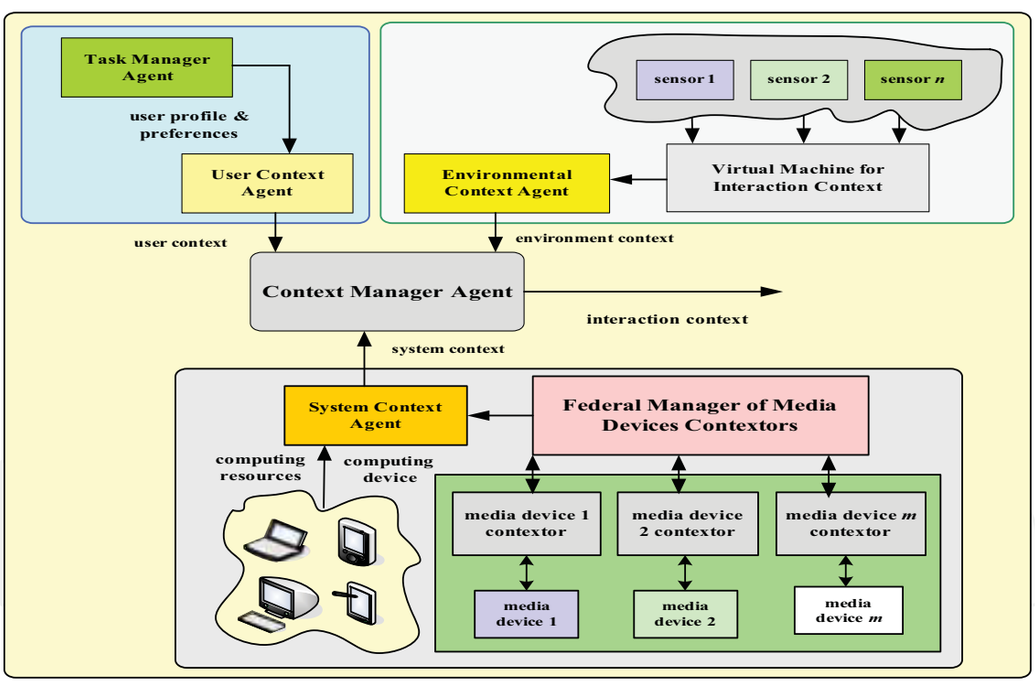



Fig. 3. The parameters that are used to determine interaction context

### 3.2 The virtual machine for incremental interaction context

To realize *incremental context* that is sensor-based, meaning that certain context parameters are interpreted based on the values obtained from certain sensors, we developed the VMIC using layered architectural approach (see Fig. 4). These architectural layers interact with one another; specifically the adjacent layers do interact with one another directly. *Layering* is a technique that is used to prevent possible cascading of errors or ripple effect whenever one

wishes to debug or modify an element of a particular layer. Whenever possible, layering is chosen as a design consideration due to this benefit. Generally, in this structure, the *top* layer is associated with the interface interacting directly with an end user while the *bottom* layer is usually associated with gadgets or hardware elements.

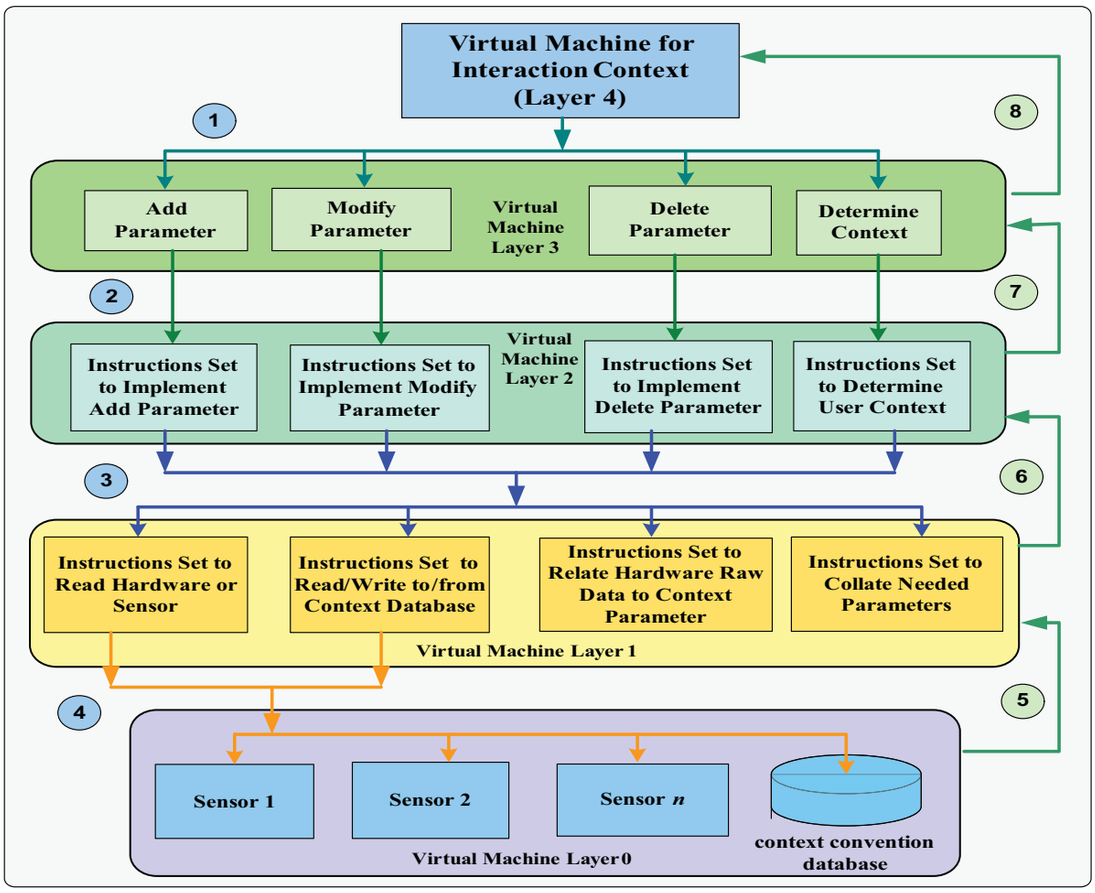



Fig. 4. The design of a virtual machine for incremental interaction context

As shown in the diagram, the **VM Layer 4** acts as the human-machine interface; its "*instruction set*" are the four functions found in **Layer 3** – the "*add parameter*", "*modify parameter*", and "*delete parameter*" are basic commands that manipulate the sensor-based context parameters while "*determine context*" yields the sensor-based context based on the values of currently-defined parameters. **VM Layer 2** is a "*library of functions*" that collectively supports Layer 3 instructions while **Layer 1** is another "*library of functions*" that acts as a link between Layer 2 and Layer 0. **Layer 0** is assigned to a collection of sensors (or machines or gadgets) that generate some raw data representing the value of a certain context parameter. Each lower layer supports the upper layer by providing the results to the functions demanded by the latter. This interdependence continues top down up to the very last layer. Consequently, the transfer of resulting data is propagated bottom up (meaning from layers 0 to 4). Layers 4, 3 and 2 are robust: the functions in these layers are independent of the context parameters, and therefore could be used by any system that deals with sensor-based context. If a new parameter needs to be added, then a minor

modification may be needed in the functions in Layer 1 and the probe, one that will supply raw data for a certain parameter, may be need to be installed in layer 0. For example, the interactions among the layers to add a new context parameter (i.e. Noise Level) are shown in Fig. 5, the deletion of a context parameter in Fig. 6 and the detection of the sensor-based context in Fig. 7. Further details on how to add, modify and delete a context parameter as well as the detection of the current sensor-based context are provided in our work in (Hina, Tadj et al. 2009). These tools are designed in generic fashion such that they can be used and integrated into any kind of system, independent of the system's application.

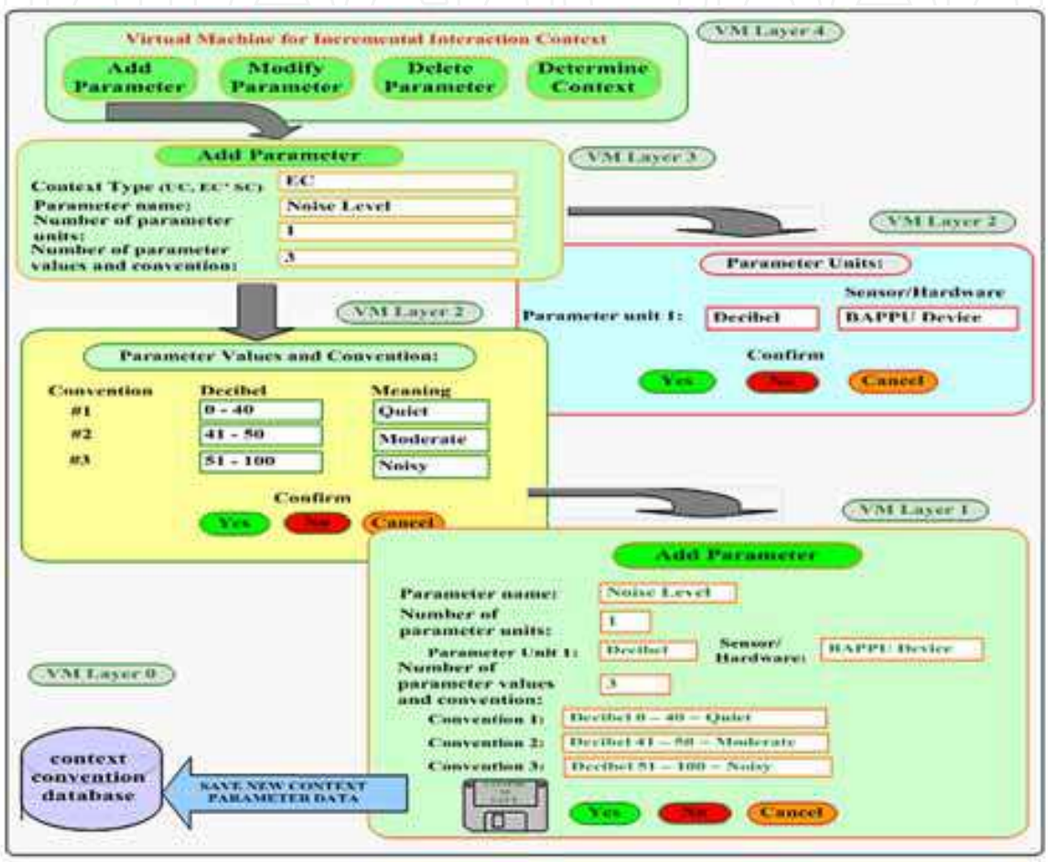



Fig. 5. The interactions among layers to add new context parameter: "*Noise Level*"

*Why a virtual machine*? The rationale for designing a virtual machine is always to come up with an efficient, isolated duplicate of a real machine. The *real machine* is always complicated, difficult to understand, and its behaviour is usually controlled by its designer. The aim therefore of *virtual machine* is to provide regular users ways of controlling and using the real machine without the necessity of having to know the intricacies of the actual machine. The end users, therefore, control the actual machine, asks it to do something for him/her using very simple instructions.

## 4. Modality and multimodal computing system

*Multimodal interaction* provides the user with multiple modes of interfacing with a computing system. Multimodal user interfaces are a research area in *human-computer*

*interaction* (HCI). In the domain of multimodal interfaces, two groups have emerged – the multimodal input and the multimodal input and output.

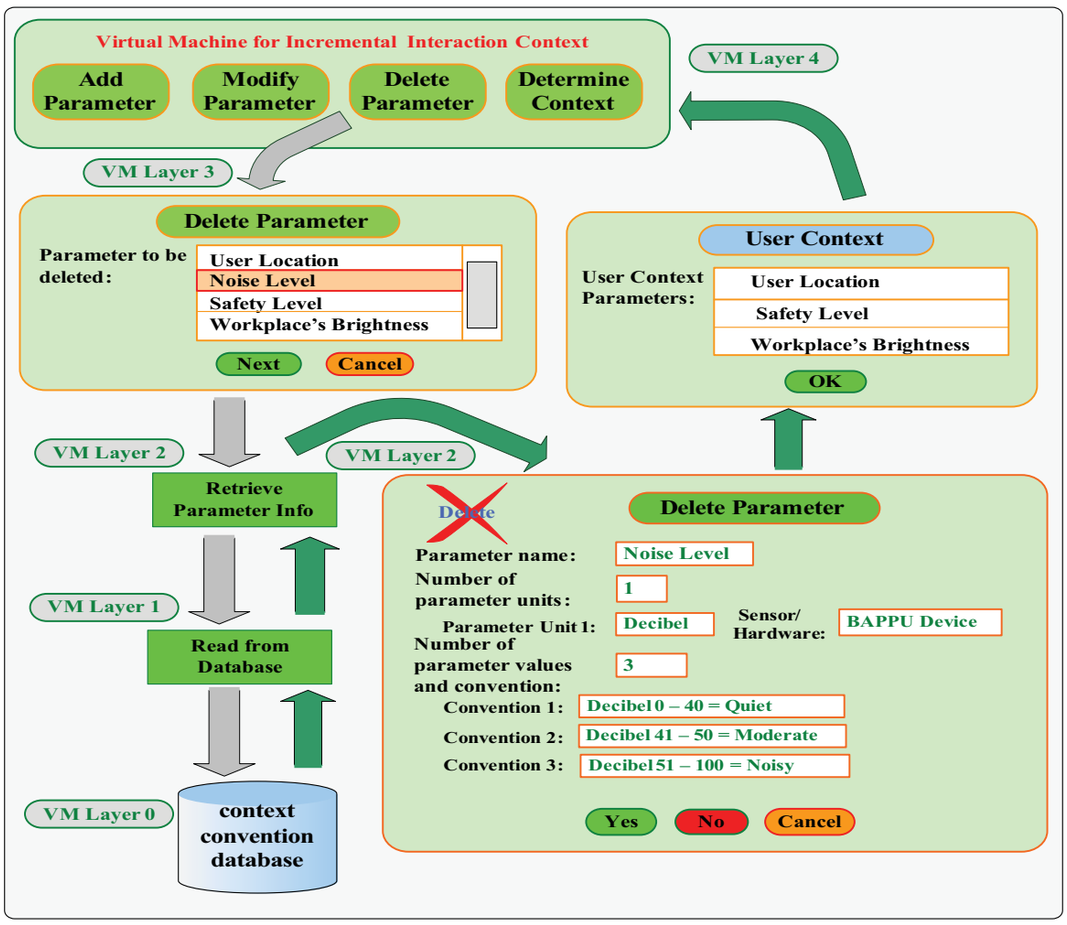



Fig. 6. The VM layers interaction to realize "*deleting a user context parameter*"

### 4.1 Multimodal input and output

The first group of multimodal interfaces combine various user input modes, beyond the usual keyboard and mouse input/output, such as speech, pen, touch, manual gestures, gaze and head and body movements. The most common of such interface combines a visual modality (e.g. a display, keyboard, and mouse) with a voice modality (speech recognition for input, speech synthesis and recorded audio for output). However other modalities, such as pen-based input or haptic input/output may be used. A sample detailed work in which mouse and speech were combined to form a multimodal fusion of input data is that of (Djenidi, Ramdane-Cherif et al. 2002; Djenidi, Ramdane-Cherif et al. 2003; Djenidi, Lévy et al. 2004).

The second group of multimodal systems presents users with multimedia displays and multimodal output, primarily in the form of visual and auditory cues. Other researchers also started to make use of other modalities, such as touch and olfaction. Proposed benefits of multimodal output system include synergy and redundancy. The information that is presented via several modalities is merged and refers to various aspects of the same process.

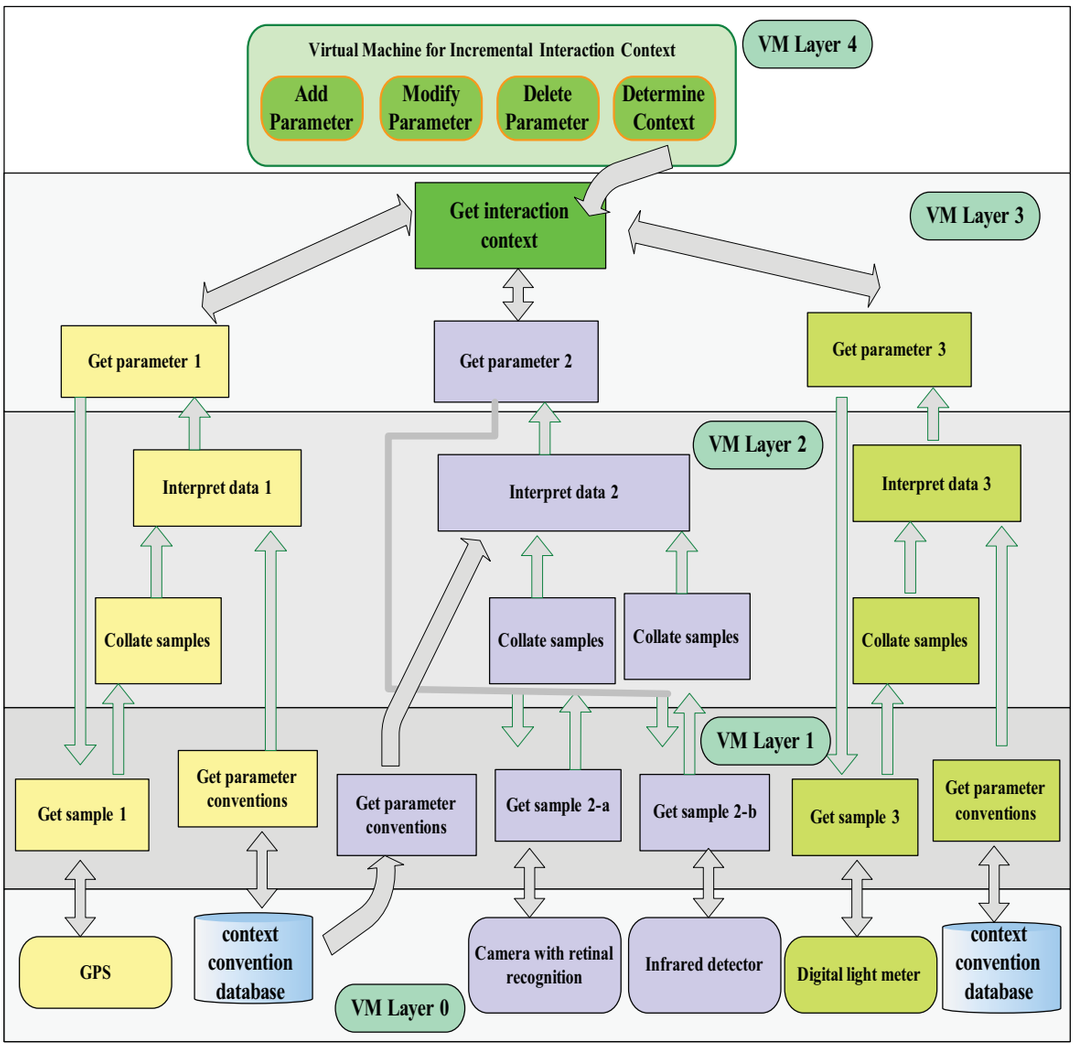



Fig. 7. VM layers interaction in detecting the current interaction context using specimen sensors

### 4.2 Modalities and media devices and their relationships

In this work, *modality* refers to the *logical* structure of man-machine interaction, specifically the *mode* for *data input* and *output* between a user and computer. Using natural language processing as basis, we classify modalities into 6 different groups:

1. **Visual Input** ($VI_{in}$) – the user's eyes are used as mechanism for data entry.
2. **Vocal Input** ($VO_{in}$) – voice or sound is captured and becomes the source of data input.
3. **Manual Input** ($M_{in}$) – data entry is done using hand manipulation or human touch.
4. **Visual Output** ($VI_{out}$) – data output is presented in the form as to be read by the user.
5. **Vocal Output** ($VO_{out}$) – sound is produced as data output; the user obtains the output by listening to it.
6. **Manual Output** ($M_{out}$) – the data output is presented in such a way that the user would use his hands to grasp the meaning of the presented output. This modality is commonly used in interaction with visually-impaired users.

To realize multimodality, there should be at *least one* modality for data input and at *least one* modality for data output that can be implemented.

There are *two* different meanings of *multimedia*. The *first* definition is that multimedia is media and content that uses a combination of different content forms. The term is used to describe a medium having multiple content forms. The term is used in contrast to media which only use traditional forms of printed or hand-produced material. Multimedia includes a combination of text, audio, still images, animation, video, and interactivity content forms. The *second* definition is that of multimedia describing electronic media devices used to store and experience multimedia content.

In this work, we take the *second* definition of multimedia and refer to the individual media as physical device that is used to implement a modality. Regardless of size, shape, colour and other attributes, all media devices – past, present or future – can be classified based on our part that uses the device to generate data input and the body part that uses the device to consume the output data. Hence, our classification of media devices is as follows:

1. **Visual Input Media** (VIM) – these devices obtain user input from human sight,
2. **Visual Output Media** (VOM) – these devices generate output that is meant to be read,
3. **Audio Input Media** (AIM) – devices that use user's voice to generate input data,
4. **Audio Output Media** (AOM) – devices whose output are meant to be heard,
5. **Touch Input Media** (TIM) – these devices generate input via human touch,
6. **Manual Input Media** (MIM) – these devices generate input using hand strokes, and
7. **Touch Output Media** (TIM) – the user touches these devices to obtain data output

It is necessary that we build a relationship between modalities and media devices for if we find a specific modality to be suitable to the given interaction context, it follows that the media devices supporting the chosen modality would be automatically selected and activated on the condition that they are available and functional. We will use formal specification in building this relationship. Let there be a function $g_1$ that maps a modality to a media group, given by **$g_1$: Modality → Media Group**. This relationship is shown in Fig. 8.

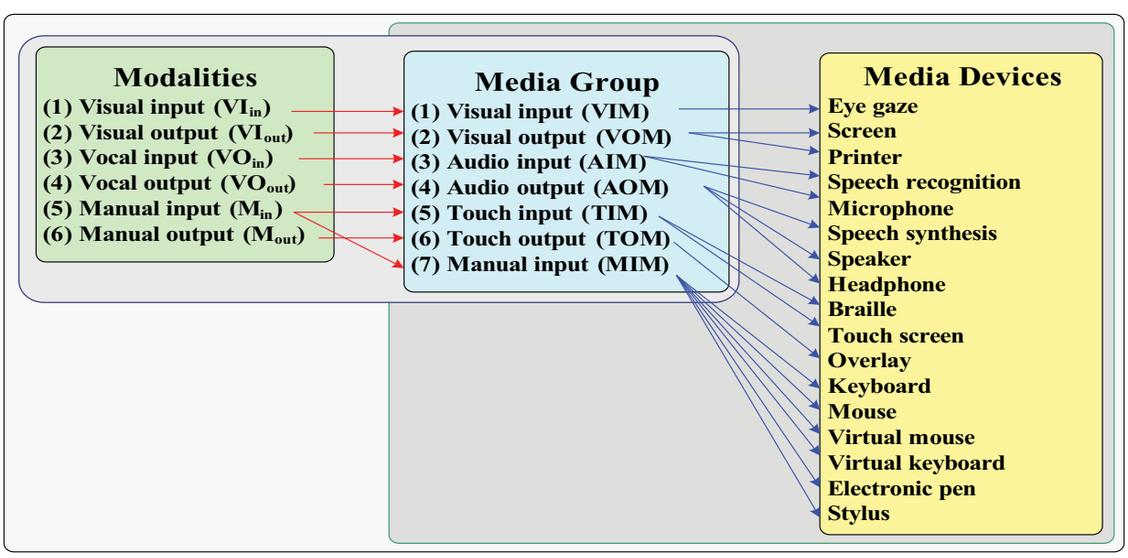



Fig. 8. The relationship between modalities and media, and media group and media devices

Often, there are many available devices that belong to the same media group. If such is the case then instead of activating them all, devices activation is determined through their

priority rankings. To support this scheme, let there be a function $g_2$ that maps a media group to a media device and its priority rank, and is denoted **$g_2$: Media Group → (Media Device, Priority)**. Hence sample elements of these functions are:

$g_1$ = {($VI_{in}$, VIM), ($VI_{out}$, VOM), ($VO_{in}$, AIM), ($VO_{out}$, AOM), ($M_{in}$, TIM), ($M_{in}$, MIM), ($M_{out}$, TOM)}

$g_2$= {(VIM, (eye gaze, 1)), (VOM, (screen, 1)), (VOM, (printer, 1)), (AIM, (speech recognition,1)), (AIM, (microphone, 1)), (AOM,(speech synthesis, 1)), (AOM, (speaker, 2)), (AOM, (headphone, 1)), etc.}.

It must be noted, however, that although media technically refers to a hardware element, we opted to include a few software elements without which $VO_{in}$ and $VO_{out}$ modalities could not possibly be implemented. These are the speech recognition and speech synthesis software.

### 4.3 Ranking media devices

The priority ranking of media devices is essential in determining which device would be activated, by default, when a certain modality is selected as apt for a given interaction context. Here, we outline the rules for prioritizing media devices:

1. The priority ranking of media devices shall be based on the relationship **$g_2$: Media Group → (Media Device, Priority)** and the elements of the function $g_2$.
2. When two or more media devices happen to belong to one media group, the priority of these devices would be based on these rules:

a. If their functionalities are identical (e.g. a mouse and a virtual mouse), activating both is incorrect because it is plain redundancy. Instead, one should be ranked higher in priority than the other. The most-commonly-used device gets the higher priority.
b. If their functionalities are complementary (e.g. a mouse and a keyboard), activating both is acceptable and their priority is identical.
c. In case that one device is more commonly used than the other (i.e. they do not always come in pair), then the more-commonly-used one gets the higher priority.
d. If both devices always come together as a pair, then both are ranked equal in priority.

| Media Group | Media Devices | | | | |
|---|---|---|---|---|---|
| | Priority = 1 | Priority = 2 | Priority = 3 | ... | Priority = *n* |
| Visual Input | Eye Gaze | | | | |
| Audio Input | Microphone, Speech Recognition | | | | |
| Touch Input | Touch Screen | Braille Terminal | | | |
| Manual Input | Mouse, Keyboard | Virtual Mouse, Virtual keyboard | Electronic Pen | Stylus | Braille |
| Visual Output | Screen | Printer | Electronic Projector | | |
| Audio Output | Speaker | Headphone, Speech Synthesis | | | |
| Touch Output | Braille | Overlay Keyboard | | | |

Table 1. A sample media devices priority table (MDPT)

In the early stage of setting up the pervasive multimodal multimedia computing system, it is essential that the end user provides this ranking. For example, in a quiet workplace, a speaker can be the top-ranked hearing output device. In a noisy environment, however, the headphone gets the top priority. An important component that implements this priority ranking is the *media devices priority table* (MDPT). See Table 1.

## 5. System adaptation to interaction context

In this section, we will present our proposed system's adaptation mechanism to the given instance of interaction context. To do this, we will derive the tools, relationships and mathematical formulas that the system has to learn and apply.

### 5.1 The History and Knowledge-based Agent (HKA)

As shown in Fig. 9, HKA is the agent tasked with the selection of modalities, of supporting media devices, and of the applications configurations based on the given instance of interaction context. Here, we discuss the concept behind HKA's knowledge acquisition. A *scenario* is the base of such knowledge.

Fig. 9 shows the generic diagram demonstrating how knowledge is acquired by HKA. The input to the *machine learning* (ML) component (responsible for analysis) is called the *pre-condition scenario* while the resulting output is called the *post-condition scenario*. The pre-condition scenarios as well as those of the post-condition scenarios are stored in a storage called *scenario repository* (SR). Whenever the ML component encounters a situation, it takes into account the parameters involved in the pre-condition scenario as well as consult its *initial knowledge* (also called *a priori knowledge*) or other previously-stored knowledge. If the pre-condition scenario is already found similar (or identical) to the one held in the scenario repository, the ML component simply takes in the corresponding post-condition scenario which is then taken as the necessary output that is bound for implementation.

In this work, we proposed the use of *case-based reasoning with supervised learning* (CBR) as a learning tool in order to automate the system's adaptation. If no knowledge is found (meaning, the scenario is new), the ML component performs calculations, determining the corresponding post-condition scenario which afterwards will be stored in the scenario repository. Over time, the ML component will accumulate enough knowledge that it will be able to "*learn*" most situations and that it will be able to react accordingly in the future.

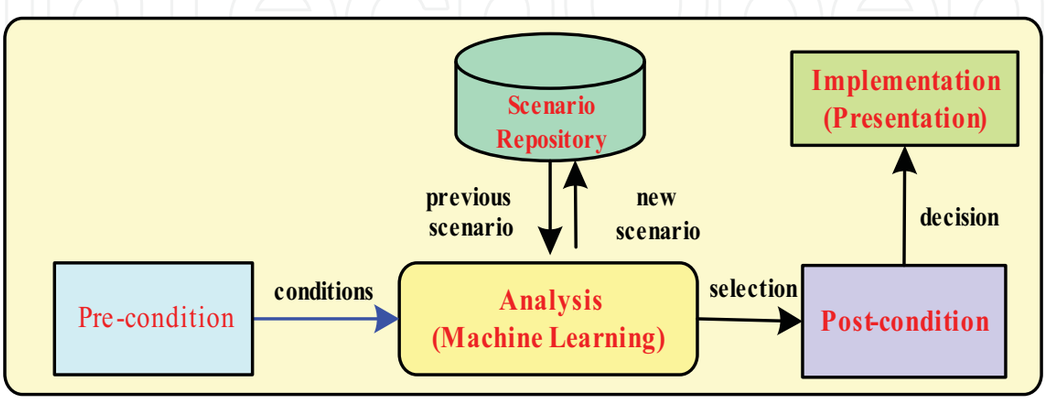



Fig. 9. Diagram showing knowledge acquisition within HKA

Using **CBR** makes it possible to find scenarios of related case. The technique, however, necessitates that a case must be identified. To use this approach, we need to model a *case* in such a way that we will end up finding a solution to the problem on hand. As stated by Kolodner (Kolodner 1993), a *case* is always composed of the same components, regardless of whatever application domain is in consideration. Its components are made up of a problem, a solution and an eventual evaluation:

**The problem** – this corresponds to the pre-condition scenario.

**The solution** – this corresponds to the resulting post-condition scenario

**The evaluation** – this would refer to the rate of relevance of the proposed solution.

The process of reuse consists of, for a new case, recovering a previously stored similar case, evaluate it and then store the new case onto the repository. This process is made up of the following steps:

**Problem representation** – For every scenario that is sent to HKA, we consult an identical case or cases that are most similar to the one in question. To do this, we need to formalize the problem part as if it is a new case in order to compare it against others that are already stored in the repository.

**Similarity calculation** – the case that is most pertinent is generally found through its similarity score with the new case. In order to do this, we come up with similarity calculation algorithm that helps in facilitating the search for similar cases.

**Search of pertinent result** – the search is based on the highest similarity score.

**Memorization** – the memorization is a choice that is usually left to the end user to decide since he is the most apt in deciding if the new case needs to be remembered or not. Even then, we proposed to the user to memorize his case.

Inspired by (Lajmi, Ghedira et al. 2007), we modify the similarity scoring scheme to reflect the needs of our system. Hence, given a *new case* (NC) and an individual case stored in the knowledge database, also called *memorized case* (MC), the similarity of the problem between the two cases (i.e. subscript indicates which case is considered) is equal to their *similarity* in their interaction context (**IC**) parameters. This relationship is given by:

$$Sim(IC_{NC}, IC_{MC}) = \frac{\sum_{i=1}^{NCC} Sim(IC_{i_{NC}}, IC_{MC})}{\max(IC_{NCC}, IC_{MCC})} \tag{1}$$

where $IC_{NC}$ and $IC_{MC}$ are the interaction context of the new case and the memorized case, respectively. NCC and MCC are the total number of community (i.e. total number of parameters) of the new case and the memorized case, respectively. Hence, $IC_{NCC}$ tells us the number of community (i.e. parameters) that makes up the interaction context of the new case while $IC_{MCC}$ denotes the number of community (i.e. parameters) that makes up the interaction context of the memorized case. The term $IC_{iNC}$ denotes the $i^{th}$ interaction context parameter of the new case where *i* is a variable that loops from 1 to NCC. The expression max($IC_{NCC}$, $IC_{MCC}$) takes whichever is greater between the number of parameters of NC and MC. $Sim(IC_{iNC}, IC_{MC}) = \max_{j=1...MCC} Sim(IC_{iNC}, IC_{jMC})$ where $IC_{jMC} \in$ ICMC and $Sim(IC_{iNC}, IC_{jMC}) \in [0, 1]$ is the similarity between parameter *i* of NC and parameter *j* of MC.

For the comparison of parameter *i* of NC against parameter *j* of MC, we need to compare how similar they are in terms of their names and values. Hence, the similarity between parameter *i* of NC and parameter *j* of MC is given by:

$$Sim(IC_{i_{NC}}, IC_{j_{MC}}) = Sim(IC_{i}Name_{NC}, IC_{j}Name_{MC}) * Sim(IC_{i}Value_{NC}, IC_{j}Value_{MC}) \quad (2)$$

The numerical value associated to the results of the comparisons of the names and values of parameters *i* of NC and *j* of MC is given below:

$$Sim(IC_{i}Name_{NC}, IC_{j}Name_{MC}) = \begin{cases} 0 \text{ if } IC_{i_{Name_{NC}}} \neq IC_{j_{Name_{MC}}} \\ 1 \text{ if } IC_{i_{Name_{NC}}} = IC_{j_{Name_{MC}}} \end{cases} \quad (3)$$

The relationship above indicates that when the parameter name of *i* of NC and parameter name of *j* of MC are different, the similarity score between the two is zero. That means, if we compare, for instance, the parameter name "*temperature*" against parameter name "*noise level*", the similarity between them is automatically zero.

If and when the two parameters have the same name, their values do matter. The relationship is given by:

$$Sim(IC_{i}Value_{NC}, IC_{j}Value_{MC}) = 1 - \left( \frac{\left| IC_{j_{Value_{MC}}} - IC_{i_{Value_{NC}}} \right|}{\left| IC_{j_{Value_{MC}}} - IC_{i_{Value_{NC}}} \right| + 1} \right) \quad (4)$$

Hence, if we compare the name of the parameter of NC against the name of the parameter of MC and both are found to be identical, say both parameters are named "*noise level*", then the similarity score for their names is 1 and we will then proceed to comparing their parameter values.

In this work, each context parameter has a name and a value, and such value (or range of values) is associated with a convention number. For example, for noise level (see Fig. 5), if the measured or sampled value is 0 to 40 dB, we say that noise level = "*quiet*" (this is considered convention 1), noise level is from 41 to 50 dB, the noise level = "*moderate*" (this is convention 2) and noise level is 51 dB or more, then noise level = "*noisy*" (this is convention 3). Indeed, for every given context parameter, it is associated with one or more conventions. In our work, as a rule of thumb, if the convention number is 1, the value of the context parameter is ideal to the user. As the convention number increases in numerical value, the context parameter is beginning to shift to the unwanted condition. For example, taking into account the conventions, if noise level = 1, then the working environment is quiet, whereas if the noise level = 3, the working environment is noisy. The convention number associated with a context parameter will be used in measuring the similarity score of two parameters being compared.

If the name of the parameter in NC and the name of the parameter of MC are the same, we then proceed to determining their similarity score with regards to their values. Consider the following cases:

**Case 1**: Parameter of NC: Noise level = 1. Parameter of MC: Noise level = 1. In this case, $Sim(Name_{NC}, Name_{MC}) = 1$ and $Sim(Value_{NC}, Value_{MC})$ will be computed as follows: 1 – (1-1)|/|1-1|+1) = 1 – (0/1) = 1 – 0 = 1. They are completely the same, both in names and values.

**Case 2**: Parameter of NC: Noise level = 1. Parameter of MC: Noise level = 2. Again, their names are the same, Sim($Name_{NC}$, $Name_{MC}$) = 1. Sim($Value_{NC}$, $Value_{MC}$) will be computed as follows: 1 – (|1-2|/|1-2|+1)| = 1 – (1/2) = 0.5. The value indicates that they are quite closed enough (i.e. 50% similar with each other).
**Case 3**: Parameter of NC: Noise level = 1. Parameter of MC: Noise level = 3. Again, their names are the same. Sim($Value_{NC}$, $Value_{MC}$) will be computed as follows: 1 – (|1-3|/|1-3|+1)| = 1 – (2/3) = 1/3. The value indicates that they are quite far from each other and is 33.3% similar with one another.
**Case 4**: Parameter of NC: Noise level = 2. Parameter of MC: Noise level = 3. Again, their names are the same. Sim($Value_{NC}$, $Value_{MC}$) will be computed as follows: 1 – (|2-3|/|2-3|+1)| = 1 – (1/2) = 1/2. The value indicates that they are quite closed to each other (50% similar).
In general, the similarity value of two parameters having identical names is 1/distance between them. It means, if they have the same value, their similarity score is 1; if they are 2 values apart, their similarity score is ½. If they are 3 values apart, their similarity score is 1/3, and 1/*n* if they are *n* values apart from each other.
Previously, we had specified that interaction context (**IC**) is actually the composition of all the elements of user context (**UC**), environment context (**EC**) and system context (**SC**). In this respect, we made a decision that the weight of **UC**, **EC** and **SC** in the composition of IC is identical (meaning, UC = EC = SC = 1/3 of IC), hence the similarity formula specified in Equation 1 becomes:

$$\begin{aligned} Sim(NC, MC) = {} & \tfrac{1}{3} Sim(UC_{NC}, UC_{MC}) + \\ & \tfrac{1}{3} Sim(EC_{NC}, EC_{MC}) + \tfrac{1}{3} Sim(SC_{NC}, SC_{MC}) \end{aligned} \tag{5}$$

The similarity between the UC of NC vs. the UC of MC is given by:

$$Sim(UC_{NC}, UC_{MC}) = \frac{\sum_{i=1}^{UC_{NCC}} Sim(UC_{i_{NC}}, UC_{MC})}{\max(UC_{NCC}, UC_{MCC})} \tag{6}$$

For the similarity measures of **EC** and **SC** of NC vs. MC, the same principle as Equation 3 must be applied, with the formula adjusted accordingly to denote **EC** and **SC**, respectively, yielding:

$$Sim(EC_{NC}, EC_{MC}) = \frac{\sum_{i=1}^{EC_{NCC}} Sim(EC_{i_{NCC}}, EC_{MCC})}{\max(EC_{NCC}, EC_{MCC})} \tag{7}$$

$$Sim(SC_{NC}, SC_{MC}) = \frac{\sum_{i=1}^{SC_{NCC}} Sim(SC_{i_{NC}}, SC_{MC})}{\max(SC_{NCC}, SC_{MCC})} \tag{8}$$

where $UC_{NCC}$ denotes the total number of community (parameters) in the **UC** of NC. The same principle applies to $EC_{NCC}$ (total number of **EC** parameters in NC) and $SC_{NCC}$ (total number of **SC** parameters in NC). This also applies to $UC_{MCC}$, $EC_{MCC}$ and $SC_{MCC}$. Note that

the number of community (number of parameters) of a new case is equal to the sum of all of its community (i.e. parameters) in **UC**, **EC** and **SC**. That is, $NCC = UC_{NCC} + EC_{NCC} + SC_{NCC}$. Similarly, $MCC = UC_{MCC} + EC_{MCC} + SC_{MCC}$.

As shown in the above relationship, we are in the assumption that the weights of **UC**, **EC** and **SC** are equal (each is 33.3%) but this figure can be easily adjusted by the expert (i.e. end user) to suit his needs.

For simplicity of discussion, we revert back our discussion of **IC** (rather than the individual components **UC**, **EC** and **SC**) to simplify the issue of finding scenarios that are similar to the new case in consideration. The formula given in Equation 1 is used to compare its similarity against other scenarios that are within its "*neighborhood*". Those considered *neighbours* are actually *scenarios* that are very close to the case in question with respect the values of its interaction context parameters. Using these parameters, we can identify the index of the case in question within the database of scenarios. We call this index "*scenario number*" or *scenNum*. For example, if scenario *i* is composed of individual interaction context parameters $IC_1, IC_2, \ldots, IC_{max}$, that is $IC_i = (IC_1, IC_2, \ldots, IC_{max})$, the scenario index assigned to this specific case is given by:

$$ScenNum = IC_{\max} + \sum_{i=1}^{\max-1} ((IC_i - 1) \bullet \prod_{j=i+1}^{\max-1} card(IC_j)) \tag{9}$$

where $card(IC_j)$ is the cardinality of the total number of values for the convention of a specific interaction context parameter j.

In the discussion that will follow, we will elaborate on the learning and adaptation mechanisms that are used by the HKA agent.

### 5.2 Selection of modalities and supporting media devices suitable to the instance of interaction context

Let *interaction context*, **IC** = $\{IC_1, IC_2, \ldots, IC_{max}\}$, be the set of all parameters that manifest the user's interaction context. At any given time, a user has a specific interaction context *i* denoted as $IC_i$, $1 \le i \le max$, which is composed of variables that are present during the conduct of the user's activity. Each variable is a function of the application domain which, in this work, is multimodality. Formally, an **IC** is a tuple composed of a specific user context (**UC**), environmental context (**EC**) and system context (**SC**). An instance of **IC** is given as:

$$IC_i = UC_k \otimes EC_l \otimes SC_m \tag{10}$$

where $1 \le k \le \max_k$, $1 \le l \le \max_l$, and $1 \le m \le \max_m$, and $max_k$, $max_l$ and $max_m$ = maximum number of possible user contexts, environment contexts and system contexts, respectively. The Cartesian product (symbol: ⊗) denotes that **IC** yields a specific combination of **UC**, **EC** and **SC** at any given time.

The user context **UC** is composed of application domain-related parameters describing the state of the user during his activity. A specific user context *k* is given by:

$$UC_k = \bigotimes_{x=1}^{\max_k} IC_{kx} \tag{11}$$

where $IC_{kx}$ = parameter of $UC_k$, *k* = the number of **UC** parameters. Similarly, any environment context $EC_l$ and system context $SC_m$ are specified as follows:

$$EC_l = \bigotimes_{y=1}^{\max_l} IC_{ly} \tag{12}$$

$$SC_m = \bigotimes_{z=1}^{\max_m} SC_{mz} \tag{13}$$

The first knowledge that the ML component must learn is to relate the interaction context to appropriate modality. Let function $\mathbf{s_1}$**: IC** $\rightarrow$ **M** maps interaction context with appropriate modalities. This function takes an instance of **IC** = {$IC_1$, $IC_2$,…, $IC_{max}$} as pre-condition scenario input to HKA and as a result returns a set of optimal modalities $M_o$ = {$m_1$, $m_2$,…, $m_{max}$}as post-condition output.
Let **M** = {$VI_{in}$, $VO_{in}$, $M_{in}$, $VI_{out}$, $VO_{out}$, $M_{out}$} be the set of modalities. Modalities are possible when the following condition holds:

$$Modality\ Possible = (VI_{in} \vee VO_{in} \vee M_{in}) \wedge (VI_{out} \vee VO_{out} \vee M_{out}) \tag{14}$$

Consequently, modality fails under the following condition:

$$\begin{aligned} Modality\ Failure = &((VI_{in} = Failed) \wedge (VO_{in} = Failed) \wedge (M_{in} = Failed)) \vee \\ &((VI_{out} = Failed) \wedge (VO_{out} = Failed) \wedge (M_{out} = Failed)) \end{aligned} \tag{15}$$

wherein symbols $\wedge$ and $\vee$ represent logical AND and OR, respectively.
Let $M_j$ = element of the power set of **M**, that is, $M_j \in \mathbb{P}(\mathbf{M})$ where $1 \leq j \leq mod_{max}$ (maximum modality). Also, let $\hat{M}$ = the most suitable $M_j$ for a given interaction context $IC_i$. This set is given by the following relationship:

$$\hat{M} = \arg\max_j P(M_j \mid IC_i) \tag{16}$$

To simplify calculation, Bayes Theorem (Kallenberg 2002), given below, can be adopted, and $P(M_j/IC_i)$ becomes:

$$P(M_j \mid IC_i) = \frac{P(IC_i \mid M_j) \times P(M_j)}{P(IC_i)} \tag{17}$$

The implementation of *Bayes Theorem* leads to the *Naive Bayes algorithm* (Mitchell 1997). The Naive Bayes algorithm is a classification algorithm that assumes that the $IC_i$ attributes $IC_1$,…,$IC_{max}$ are all conditionally independent of one another given a post condition $M_j$. The representation of $P(IC_i \mid M_j)$ becomes:

$$\begin{aligned} P(IC_i \mid M_j) &= P(IC_1,...,IC_{max} \mid M_j) \\ &= P(IC_1 \mid M_j) \times ... \times P(IC_{max} \mid M_j) \\ &= \prod_{i=1}^{max} P(IC_i \mid M_j) \end{aligned} \tag{18}$$

Here, our goal is to train a classifier that, given a new $IC_i$ to classify, will provide the probability distribution on all possible values of **M** (i.e. $M_1$, $M_2$, …, $M_{max}$). Given that $IC_i$ = ($IC_1$, $IC_2$, ...,$IC_{max}$ ), then the equation above becomes:

$$P(M_j | IC_1 ... IC_{max}) = \frac{P(M_j)P(IC_1 ... IC_{max} | M_j)}{\sum_{k=1}^{m} P(M_k)P(IC_1 ... IC_{max} | M_k)} \tag{19}$$

The above equation can also be written as:

$$P(M_j | IC_1 ... IC_{max}) = \frac{P(M_j)\prod_{i=1}^{max} P(IC_i | M_j)}{\sum_{k=1}^{m} P(M_k)\prod_{i=1}^{max} P(IC_i | M_k)} \tag{20}$$

which is the fundamental equation for the *Naive Bayes classifier*. Given a new instance of interaction context $IC_i = (IC_1, \ldots, IC_{max})$, the equation shows how to calculate the probability that $M_j$ will take given the observed attribute values of $IC_i$ and given that the distributions $P(M_j)$ and $P(IC_i | M_j)$ are estimated values taken from training data (SR). If we are interested only in the most suitable value of $M_j$, then we have the *Naive Bayes classification rule*:

$$\hat{M} = \arg\max_j \left( \frac{P(M_j)\prod_{i=1}^{max} P(IC_i | M_j)}{\sum_{k=1}^{m} P(M_k)\prod_{i=1}^{n} P(IC_i | M_k)} \right) \tag{21}$$

Given that the denominator does not depend on parameter *j*, then the above equation becomes

$$\hat{M} = \arg\max_j \left( P(M_j)\prod_{i=1}^{max} P(IC_i | M_j) \right) \tag{22}$$

Given that $IC_i$ is composed of elements of **UC, EC** and **SC**, then $P(IC_i/M_j)$ can be written as:

$$P(IC_i | M_j) = P(UC_k | M_j) \times P(EC_l | M_j) \times P(SC_m | M_j) \tag{23}$$

Note that in $IC_i$, the parameters $IC_1, \ldots IC_{max}$ are mutually exclusive. Using Equation 11, Equation 12 and Equation 13, we replace each element of $IC_i$ with the corresponding value. For example, the relationship involving user context $UC_k$ given the modality $M_j$ is given by:

$$P(UC_k | M_j) = \prod_{i=1}^{uc_{\max}} P(UC_i | M_j) \tag{24}$$

In conclusion, the **optimal modality** for whatever instance of interaction context is given by:

$$\hat{M} = \arg\max_j \left( \left( \prod_{k=1}^{uc_{\max}} P(UC_k | M_j) \times \prod_{l=1}^{ec_{\max}} P(EC_l | M_j) \times \prod_{m=1}^{sc_{\max}} P(SC_m | M_j) \times P(M_j) \right) \right) \tag{25}$$

where P(Mj) = frequency count of $M_j$ in scenario repository (SR) ÷ cardinality of SR and $uc_{max}$, $ec_{max}$ and $sc_{max}$ are, respectively, the total number of **UC** parameters, **EC** parameters and **SC** parameters within the interaction context being considered.

To illustrate the usage of the derived equation above, let us consider, for example, an interaction context that is composed of the following parameters: **IC** = ($IC_1$, $IC_2$, $IC_3$, $IC_4$, $IC_5$, $IC_6$, $IC_7$) wherein

- $IC_1$ = {true | false} = if user is manually disabled,
- $IC_2$ = {true | false} = if user is mute,
- $IC_3$ = {true | false} = if user is deaf,
- $IC_4$ = {true | false} = if user is familiar with Braille,
- $IC_5$ = {quiet | noisy} = environment's noise level,
- $IC_6$ ={silence required | silence optional} = environment's noise level restriction, and
- $IC_7$ = {PC or Laptop or MAC | PDA | Cell phone} = user's computing device.

Let us assume further that the intended user is a **visually-impaired** one. In this case, $Modality\ Possible = (VO_{in} \vee M_{in}) \wedge (VO_{out} \vee M_{out})$ wherein there are only 4 suitable modalities, namely *vocal* input and output and *tactile* (or manual) input and output.

Given the above constraints, the set of possible modalities is given by **M** = {$M_1$, $M_2$, $M_3$, $M_4$, $M_5$, $M_6$, $M_7$, $M_8$, $M_9$} wherein $M_1$ = {$M_{in}$, $M_{out}$}; $M_2$ = {$M_{in}$, $VO_{out}$}; $M_3$ = {$VO_{in}$, $VO_{out}$}; $M_4$ = {$VO_{in}$, $M_{out}$}; $M_5$ = {$VO_{in}$, $M_{out}$, $VO_{out}$}; M6 = {$M_{in}$, $M_{out}$, $VO_{out}$}; $M_7$ = {$M_{in}$, $VO_{in}$, $VO_{out}$}; $M_8$ = {$M_{in}$, $VO_{in}$, $M_{out}$}; $M_9$ = {$M_{in}$, $VO_{in}$, $M_{out}$, $VO_{out}$}. In this example, let us assume the following interaction context: (a) **user context**: blind with no further handicaps, familiar with Braille; hence $IC_1$ = False, $IC_2$ = False, $IC_3$ = False, $IC_4$ = True; (b) **environmental context**: the user is in a classroom, then $IC_5$ = noisy, $IC_6$ = silence required (c) **system context**: the user works on a laptop; $IC_7$ = Laptop.

The system now finds the modality that suits the given interaction context. Let us assume that a certain multimodal computing system's SR contains recorded scenarios as shown in Fig. 10. The given figure is generated by using WEKA (*Waikato Environment for Knowledge Analysis*) (Witten and Frank 2005) which is a collection of machine learning algorithms for data mining tasks. It is used in testing a machine learning algorithm as it contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization.

As shown in the diagram, there are already 15 scenarios representing the system's acquired knowledge. The 16th scenario represents a new case. Using Equation 22, and with reference to the given interaction context and SR, the suitability score of Mj (where j = 1 to 9) can be calculated. Let us consider, for instance, the calculations involved with modality $\mathbf{M_1}$:

**Suitability_Score (M1)** = $P(IC_1 = False \mid M_1) * P(IC_2 = False \mid M_1) * \ldots * P(IC_7 = Laptop \mid M_1) * P(M_1) = 1 * 0.5 * 0 * \ldots * 2/15 = \mathbf{0}$

wherein $P(IC_1 = False \mid M_1) = 2/2$, $P(IC_2 = False \mid M_1) = ½$, $P(IC_3 = False \mid M_1) = 0/2$, $P(IC_4 = True \mid M_1) = 2/2$, $P(IC_5 = Noisy \mid M_1) = ½$, $P(IC_6 = silence\ required \mid M_1) = ½$, and $P(IC_7 = Laptop \mid M_1) = ½$. Also, $P(M_1) = 2/15$.

Similarly, we do calculate the suitability score of all other remaining modalities. Using the same procedure, the modality that yields the highest suitability score is $\mathbf{M_6}$:

Suitability_Score ($M_6$) = $P(IC_1 = False \mid M_6) \times P(IC_2 = False \mid M_6) \times \ldots \times P(IC_7 = Laptop \mid M_6) \times P(M_6) = 1 \times 2/3 \times 1 \times 1 \times 2/3 \times 1/3 \times 1/3 \times 3/15 = \mathbf{0.00976}$.

By applying the ML algorithm (see Fig. 11), $\mathbf{M_6}$ appears to respect the conditions on possibility of modality; hence, it is chosen as the *optimal modality* for the given **IC**. This new scenario will be added to SR as a newly-acquired knowledge (i.e. as scenario #16 in SR).

ARFF-Viewer - D:\JMM\SR.arff

File Edit View

SR.arff

Relation: Selection_of_Modality

| No. | A1 Nominal | A2 Nominal | A3 Nominal | A4 Nominal | A5 Nominal | A6 Nominal | A7 Nominal | class Nominal |
|---|---|---|---|---|---|---|---|---|
| 1 | False | False | False | False | Noisy | Optional | CellPhone | M2 |
| 2 | True | False | False | False | Quiet | Optional | Laptop | M3 |
| 3 | False | False | False | True | Noisy | Required | PC | M6 |
| 4 | False | False | True | True | Quiet | Required | PC | M1 |
| 5 | False | True | False | True | Quiet | Optional | Laptop | M6 |
| 6 | False | False | False | True | Quiet | Optional | PC | M9 |
| 7 | False | False | False | False | Quiet | Optional | PC | M7 |
| 8 | False | True | True | True | Noisy | Optional | Laptop | M1 |
| 9 | False | False | False | True | Quiet | Optional | CellPhone | M7 |
| 10 | False | False | True | True | Quiet | Optional | MAC | M8 |
| 11 | True | False | False | False | Quiet | Optional | PDA | M3 |
| 12 | False | False | False | True | Quiet | Optional | Laptop | M9 |
| 13 | False | False | False | True | Noisy | Optional | PC | M6 |
| 14 | False | False | True | True | Quiet | Optional | PC | M8 |
| 15 | False | True | False | False | Quiet | Optional | Laptop | M2 |
| 16 | False | False | False | True | Noisy | Required | Laptop | ??? |

Fig. 10. A sample snapshot of a scenario repository (SR)

Fig. 11 shows the algorithm in finding the optimal modalities given an instance of interaction context. The first algorithm calculates the suitability score of each element of the power set of **M**, $\mathbb{P}$(**M**) for the given context. The second algorithm finds the optimal modality.

**Suitability Score of Modality $M_j$**
Input:
1. Interaction Context $IC_i$.
2. Modality $M_j$
3. Scenario repository SR
Output: Suitability score of $M_j$
Procedure:
1. get $IC_i$, $UC_{max}$, $EC_{max}$ and $SC_{max}$
2. for each UC Parameter $UC_i$ do
3. $P(UC_k/M_j) = P(UC_k/M_j) * P(UC_i/M_j)$
4. endfor
5. for each EC Parameter $EC_i$ do
6. $P(EC_l/M_j) = P(EC_l/M_j) * P(EC_i/M_j)$
7. endfor
8. for each SC Parameter $SC_i$ do
9. $P(SC_m/M_j) = P(SC_m/M_j) * P(SC_i/M_j)$
10. endfor
11. $P(M_j) = \#(M_j)/card(SR)$
12. $Suitability\_Score\_M_j = P(UC_k/M_j)*P(EC_l/M_j)*P(SC_m/M_j)*P(M_j)$
13. Return $Suitability\_Score\_M_j$
Endprocedure

**Finding_Optimal_Modality**
Input:
1. Interaction Context $IC_i$.
2. Modality set M
3. Scenario Repository SR
Output: Optimal modality
Procedure:
1. get $IC_i$ and Modality set M
2. init = 1
3. Max = Suitability Score of modality $M_{init}$
4. Optimal_modality = $M_{init}$
5. for each element $M_j$ of power set of M
6. if ($VI_{in}$ OR $VO_{in}$ OR $M_{in}$) $\in M_j$ AND ($VI_{out}$ OR $VO_{out}$ OR $M_{out}$) $\in M_j$ then
7. if Suitability Score of $M_j$ > Max then
8. Max = Suitability Score of $M_j$
9. Optimal_modality = $M_j$
10. endif
11. endif
12. endfor
13. Return Optimal_modality
Endprocedure

Fig. 11. Algorithms: (Left) Given an interaction context $IC_i$, the algorithm calculates the suitability score of each modality $M_j$ belonging to the power set $\mathbb{P}$ (**M**), (Right) Algorithm for finding the optimal modality

Once the optimal modalities for data input and output to the instance of interaction context are chosen, we then proceed to determining whether there are sufficient media devices that will support the selected modalities. If such is the case, then the concerned media devices are activated, otherwise, the optimal modality that was chosen earlier needs to be updated. That is to say that the modality that cannot be supported by available media devices is taken out from the list. This results in the HKA's evaluation if Equation 11 still holds true. If the result is affirmative, then modality is possible, the replacement modality is calculated and the newly found optimal modality is implemented. Otherwise, there is a failure of modality.
Generally, there is more than one media device that supports a specific modality. Consequently, when too many devices are available to support the chosen modality, then only the top-ranked media device needs to be activated. Moreover, the ranking of media devices also serves purpose in finding replacement to a failed device. When a top-ranked device is malfunctioning or not available then the device that is next in priority ranking is activated. Given a specific media device $D_i$ ($1 \le i \le n$), where $i$ is priority index and $n$ is the number of media devices that can support the chosen modality $M_j$, then the probability that it is adopted to implement the chosen modality is given by:

$$P(D_i \mid M_j) = 1 - \sum_{1}^{i-1} (1/n) \tag{26}$$

To demonstrate the essence of the above equation, supposed that there are 3 media devices (n = 3) supporting modality $M_j$, denoted by $D_1$ (first priority), $D_2$ (second priority) and $D_3$ (third priority). The probability that $D_1$ is selected for implementation of modality $M_j$ is 1 – 0 = 1, that of $D_2$ is 1 – 1/3 = 2/3 and that of $D_3$ is 1 – (1/3 + 1/3) = 1 – 2/3 = 1/3. Note that in this work, the numerical subscripts of those devices with higher priority are numerically smaller than the numerical subscript of those with lower priority. Example is device $D_1$ (subscript is 1) has a higher priority than device $D_2$ (subscript is 2).
The *media devices priority table* (MDPT) is a tool that we used in implementing the media devices' priority ranking. In MDPT, devices are grouped based on their category and ranked by priority. The top-ranked media device is always chosen for activation. When the highly-ranked device is found defective, the MDPT helps in determining which one replaces the defective one.
When a new media device $d_{new}$ is added or introduced to the system for the first time, the device is associated to a modality and is given a priority ranking $r$ by the user. What happen to the rankings of other devices $d_i$, ($1 \le i \le n$, and n = number of media devices) which are in the same modality as $d_{new}$ in the MDPT? Two things may happen, depending on the user's selection. The *first possibility* is that after having the new device's priority **Priority($d_{new}$)** set to $r$ then the priority of the other device i, ($1 \le i \le n$) denoted **Priority($d_i$)**, remains the same. The *second possibility* is the priority rankings of all media devices ranked $r$ or lower are adjusted such that their new priority rankings are one lower than their previous rankings. Formally, in Z (Lightfoot 2001), this is specified as: $\forall \mathbf{i}, \exists \mathbf{r}: \mathbb{N}_1; \forall \mathbf{d_i}, \exists \mathbf{d_{new}}: \mathbf{Devices} \mid (\mathbf{Priority(d_{new}) = r} \wedge \mathbf{Priority(d_i) \ge r}) \Rightarrow \mathbf{Priority(d_i)' = Priority(d_i) + 1}$.
We also proposed mechanism for dynamic reconfiguration of the system to make it more fault tolerant, keeping the system persistent and able to resist breakdown in case certain media devices supporting chosen modalities are found to be missing or malfunctioning. Fig. 12 shows how a MDPT is assigned to a specific scenario. Fig. 13 demonstrates how the system finds a replacement to a missing or defective media device.

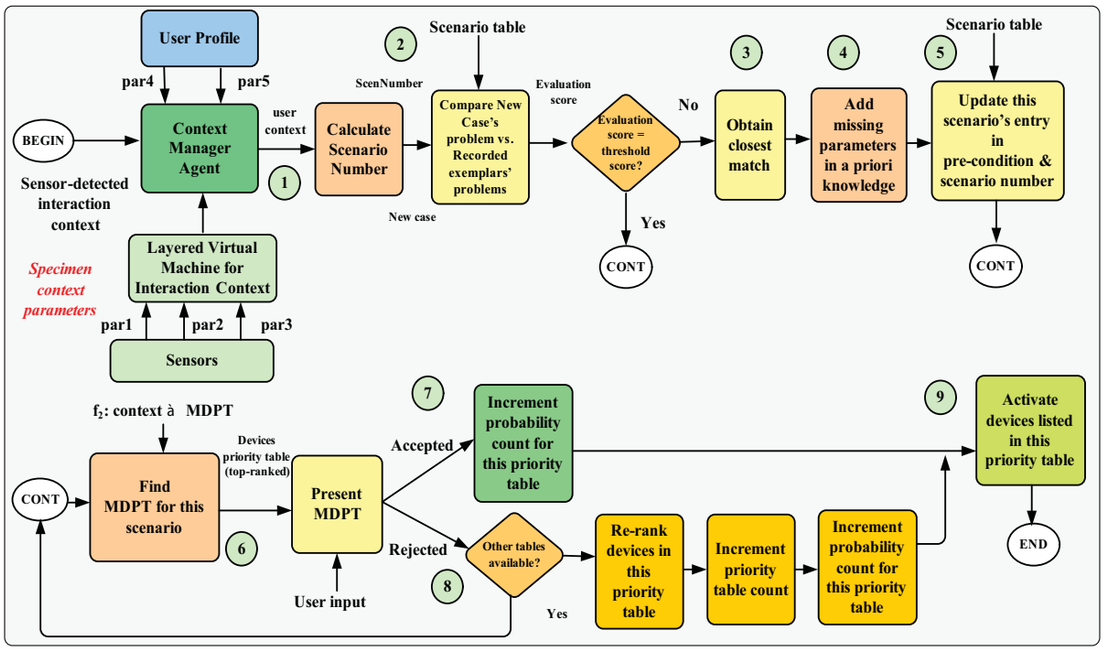



Fig. 12. The training for choosing appropriate MDPT for a specific context

## 6. Conclusion

This work – **A multi-agent based multimodal system adaptive to the user's interaction conte**xt – is an intelligent system design. It supports pervasive computing, multimedia and multimodal computing, dynamic configuration of software architecture and machine learning. It is a contribution in the domain of the advancement of pervasive multimodality.

Our work supports *transparency* and *calm technology*, realizing Marc Weiser's vision of *ubiquitous computing*. In our work, the computer, as it takes its dynamic configuration to adapt itself to the current instance of interaction context, becomes an information tool that does not demand the focus and attention of the user. The adaptation of the system to provide the end user with the necessary and suitable modality of human-machine communication, supported by its associated media devices yields a result in which the end user concentrates on his computing task and not bothers himself with the intricacies of context awareness and the system's adaptation to it. In effect, the realization of this concept renders the computer to become servant to the user needs rather than the user spending time and effort to serve the needs of the computer.

In connection with our idea that context should be inclusive – meaning that it will fit all definitions of context given by previous researchers – we broaden the notion of context to become interaction context. *Interaction context* refers to the collective context of the user (i.e. user context), of his working environment (i.e. environment context) and of his computing system (i.e. system context). Each of these interaction context elements – user context, environment context and system context – is composed of various parameters that describe the state of the user, of his workplace and his computing resources as he undertakes an activity in accomplishing his computing task, and each of these parameters may evolve over time. To realize the incremental definition of incremental context, we developed a tool called layered *virtual machine for incremental interaction context* which can be used to add, modify

and delete a context parameter on one hand and determine the sensor-based context (i.e. context that is based on parameters whose values are obtained from raw data supplied by sensors) on the other.

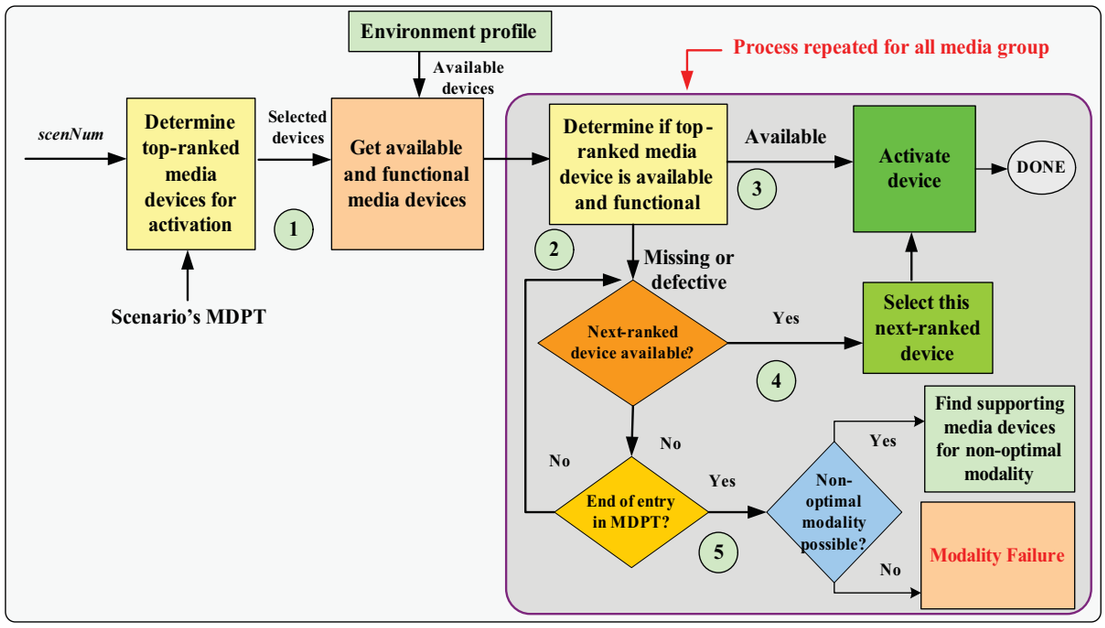



Fig. 13. The process of finding replacement to a failed or missing device

To benefit from the richness of interaction context with regards to communication in human-machine interaction, we invoke the adoption of multimodality which allows for a much wider range of modes and forms of communication, selected and adapted to suit the given user's interaction context. In multimodal communication, multimodality is beneficial to the end user because with multimodality, the weaknesses of one mode of interaction, with regards to its suitability to a given situation, is compensated by replacing it with another mode of communication that is more suitable to the situation. Multimodality also promotes inclusive informatics as those with permanent or temporary disability are given the opportunity to use and benefit from information technology advancement.

In our investigation of the current state of the art, we realize that a great deal of efforts were exerted in defining what context is all about, how to acquire it, how to disseminate it within the system and use it to suit the needs of a system in a specific domain of application (e.g. healthcare, education, etc.). Also, a great deal of efforts on ubiquitous computing were devoted on some application domains (e.g. identifying the user whereabouts, identifying services and tools, etc.) but there was no effort made with regards to making multimodality pervasive and accessible to various user situations. To this end, our work provides for the much needed solutions and answers. Our work is an architectural design that exhibits adaptability to a much larger context called interaction context. It is intelligent and pervasive, adaptive even when the user is stationary or mobile. It has mechanisms serving a specific purpose. That is, given an instance of interaction context, one which evolves over time, our system determines the optimal modalities that suit such interaction context. By optimal, we mean the selection is based on the trade-offs on appropriate multimodality after considering the given interaction context, available media devices that support the

modalities and user preferences. We designed a mechanism (i.e. a paradigm) that does this task and simulated its functionality with success. This mechanism employs machine learning and uses case-based reasoning with supervised learning. An input to this decision-making component is an instance of interaction context and its output is the most optimal modality and its associated media devices that are for activation. This mechanism is tasked to continuously monitor the user's interaction context and on behalf of the user continuously adapts accordingly. This adaptation is through dynamic reconfiguration of the pervasive multimodal system's architecture.

Our work differs from previous works in the domain in the sense that while others capture, disseminate and consume context to suit its preferred domain of application, our system captures the interaction context and reconfigures its architecture dynamically in generic fashion with the aim that the user may continue working on his task anytime, anywhere he wishes regardless of the application domain he wishes to undertake. In effect, the system that we come up with, being generic in design, can be integrated with ease or little amount of modification to various computing systems of various domains of applications. This is our main contribution.

## 7. References

Abowd, G. D. and Mynatt, E. D. (2000). Charting Past, Present and Future Research in Ubiquitous Computing. *ACM Transactions on Computer-Human Interaction* 7(1): pp. 29 - 58.

Banavar, G., Beck, J., et al. (2000). Challenges: An Application Model for Pervasive Computing. *6th Annual Intl. Conf. on Mobile Computing and Networking* (MobiCom 2000), Massachusetts, USA.

Bennett, F., Richardson, T., et al. (1994). Teleporting - Making Applications Mobile. *IEEE Workshop on Mobile Computing Systems and Applications*. Santa Cruz, California**:** pp. 82 - 84.

Buxton, W. (1983). Lexical and Pragmatic Considerations of Input Structures. Computer Graphics *Journal of the ACM* 17(1): pp. 31 - 37.

Chen, G. and Kotz, D. (2000). A Survey of Context-Aware Mobile Computing Research. *Technical Report*, Department of Computer Science, Darmouth College.

CMU_CS. (2010). Speech at CMU. from www.speech.cs.cmu.edu.

Coutaz, J., Crowley, J. L., et al. (2005). Context is key. *Communications of the ACM* 48(3): pp. 49-53.

CS_Rutgers. (2010). The Vision, Interaction, Language, Logic and Graphics Environment. from http://www.cs.rutgers.edu/~village/.

DeVaul, R. W. and Pentland, A. S. (2000). The Ektara Architecture: The Right Framework for Context-Aware Wearable and Ubiquitous Computing Applications. *MIT Technical Report***.**

Dey, A. K. (2001). "Understanding and Using Context " *Springer Personal and Ubiquitous Computing* 5(1): pp. 4 - 7.

Dey, A. K. and Abowd, G. D. (1999). Towards a Better Understanding of Context and Context-Awareness. *1st Intl. Conference on Handheld and Ubiquitous Computing*, Karlsruhe, Germany, Springer-Verlag, LNCS 1707.

Dey, A. K., Salber, D., et al. (2001). A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human- Computer Interaction Journal* 16(2-4): pp. 97-166.

Djenidi, H., Lévy, N., et al. (2004). Generic Multi-Agents Architecture for Multimedia Multimodal Software Environment. *International Journal of Object Technology* 3(8,): pp. 147-168.

Djenidi, H., Ramdane-Cherif, A., et al. (2002). Dynamic Multi-agent Architecture for Multimedia Multimodal Dialogs. I*EEE Workshop on Knowledge Media Networking*.

Djenidi, H., Ramdane-Cherif, A., et al. (2003). Generic Multimedia Multimodal Agents Pradigms and their Dynamic reconfiguration at the Architecture Level. Architecture for Software Environment. *Eurasip : European Journal of Applied Signal Processing- Special Issue on Multimedia Apllications*.

Dong, S. H., Xiao, B., et al. (2000). Multimodal user interface for internet. Jisuanji Xuebao/*Chinese Journal of Computers* 23(12): pp. 1270-1275.

Efstratiou, C., Cheverst, K., et al. (2001). An Architecture for the Effective Support of Adaptive Context-Aware Applications. *2nd Int. Conf. in Mobile Data Management* (MDM'01), Hong Kong.

Elrod, S., Bruce, R., et al. (1992). Liveboard: a large interactive display supporting group meetings, presentations and remote collaboration. *ACM Conference on Human Factors in Computing Systems*, Monterey, CA, USA.

Esler, M., Hightower, J., et al. (1999). Next Century Challenges: Data-Centric Networking for Invisible Computing. *5th Annual Intl. Conference on Mobile Computing Networking* (MobiCom'99), Seattle, WA, USA.

Eustice, K., Lehman, T. J., et al. (1999). A Universal Information Appliance. *IBM Systems Journal* 38(4): pp. 575-601.

Garlan, D., Siewiorek, D., et al. (2002). Project Aura: Towards Distraction-Free Pervasive Computing. *IEEE Pervasive Computing, Special Issue on Integrated Pervasive Computing Environments* 21(2): pp. 22 - 31.

Gwizdka, J. (2000). "What's in the Context?". Workshop on Context-Awareness, *CHI 2000, Conference on Human factors in Computing Systems*, The Hague, Netherlands.

Held, A. (2002). Modeling of Context Information for Pervasive Computing Applications. *6th World Multiconference on Systemics, Cybernetics and Informatics* (SCI2002), Orlando, FL.

Hina, M. D., Tadj, C., et al. (2009). Autonomic Communication in Pervasive Multimodal Multimedia Computing System. Book "*Autonomic Communication*". Vasilakos, A. V., Parashar, M., Karnouskos, S. and Pedrycz, W., Springer**:** pp. 251 - 283.

Horn, P. (2001). Autonomic Computing: IBM's Perspective on the State of Information Technology, *IBM Research***:** pp. 1 - 22.

Indulska, J., Loke, S. W., et al. (2001). An Open Architecture for Pervasive Systems. *3rd Intl. Work. Conf. on Distributed Applications and Interoperable Systems* (DAIS 2001), Kraków, Poland.

Indulska, J., Robinson, R., et al. (2003). Experiences in Using CC/PP in Context-Aware Systems. *4th Intl. Conf. on Mobile Data Management*, Melbourne, Australia.

Kallenberg, O. (2002). *Foundations of Modern Probability*, Springer.

Kantarjiev, C. K., Demers, A., et al. (1993). Experiences with X in a wireless environment. Mobile & Location-Independent Computing *Symposium on Mobile & Location-Independent Computing*, Cambridge, MA, USA.

Kephart, J. O. and Chess, D. M. (2001). The Vision of Autonomic Computing, *IBM Thomas J. Watson Research Centre***:** pp. 41 - 50.

Kindberg, T. and Barton, J. (2001). A Web-Based Nomadic Computing System. *Elsevier Computer Networks* 35(4): pp. 443-456.

Kolodner, J. (1993). *Case-Based Reasoning*. San Mateo, CA, Morgan Kaufman.

Lajmi, S., Ghedira, C., et al. (2007). Une méthode d'apprentissage pour la composition de services web. *L'Objet* 8(2): pp. 1 - 4.

Lightfoot, D. (2001). *Formal Specification Using Z*, 2nd ed., McMillan Press.

Mitchell, T. (1997). *Machine Learning*, McGraw-Hill.

Oviatt, S. (2002). *Multimodal Interfaces: Handbook of Human-Computer Interaction*. New Jersey, USA, Lawrence Erbaum.

Pascoe, J. (1998). Adding Generic Contextual Capabilities to Wearable Computers. 2nd *IEEE Intl. Symposium on Wearable Computers* Pittsburg, IEEE Computer Society.

Prekop, P. and Burnett, M. (2003). Activities, context and ubiquitous computing. *Computer Communications* 26(1): pp. 1168-1176.

Razzaque, M. A., Dobson, S., et al. (2005). Categorization and Modelling of Quality in Context Information. IJCAI 2005 *Workshop on AI and Autonomic Communications*, Edinburg, Scotland.

Ringland, S. P. A. and Scahill, F. J. (2003). Multimodality - The future of the wireless user interface. *BT Technology Journal* 21(3): pp. 181-191.

Schilit, B., Adams, N., et al. (1994). Context-aware computing applications. *First Workshop on Mobile Computing Systems and Applications*, WMCSA 1994, Santa Cruz, California, USA, .

Schilit, B. N., Adams, N., et al. (1993). The PARCTAB mobile computing system. *Fourth Workshop on Workstation Operating Systems* (WWOS-IV), Napa, CA, USA.

Schilit, B. N. and Theimer, M. M. (1994). Disseminating Active Map Information to Mobile Hosts. *IEEE Network* 8(5): pp. 22 - 32.

Strang, T. and Linnhoff-Popien, C. (2003). Service Interoperability on Context Level in Ubiquitous Computing Environments. *Intl. Conf. on Advances in Infrastructure for Electronic Business, Education, Science, Medicine, and Mobile Technologies on the Internet* (SSGRR 2003), L'Aquila, Italy.

Truman, T. E., Pering, T., et al. (1998). The InfoPad multimedia terminal: a portable device for wireless information access. *IEEE Transactions on Computers* 47(10): pp. 1073-1087.

Want, R., Hopper, A., et al. (1992). Active badge location system. A*CM Transactions on Information Systems* 10(1): pp. 91-102.

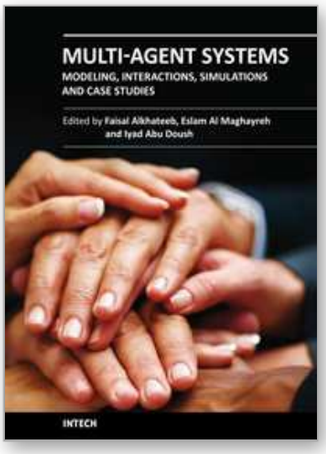
Want, R., Schilit, B. N., et al. (1995). Overview of the ParcTab ubiquitous computing experiment. *IEEE Personal Communications* 2(6): pp. 28-43.

Weiser, M. (1991). The computer for the twenty-first century. *Scientific American* 265(3): pp. 94 - 104.

Weiser, M. (1993). Some computer science issues in ubiquitous computing. *Communications of the ACM* 36(7): pp. 74-84.

Weiser, M. and Brown, J. S. (1996). Designing Calm Technology. *Powergrid Journal* 1(1): pp. 94 - 110.

Witten, I. H. and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. San Francisco, Morgan Kaufmann.




**Multi-Agent Systems - Modeling, Interactions, Simulations and Case Studies**

Edited by Dr. Faisal Alkhateeb

ISBN 978-953-307-176-3
Hard cover, 502 pages
**Publisher** InTech
**Published online** 01, April, 2011
**Published in print edition** April, 2011

A multi-agent system (MAS) is a system composed of multiple interacting intelligent agents. Multi-agent systems can be used to solve problems which are difficult or impossible for an individual agent or monolithic system to solve. Agent systems are open and extensible systems that allow for the deployment of autonomous and proactive software components. Multi-agent systems have been brought up and used in several application domains.

**How to reference**

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Manolo Dulva Hina, Chakib Tadj, Amar Ramdane-Cherif and Nicole Levy (2011). A Multi-Agent based Multimodal System Adaptive to the User's Interaction Context, Multi-Agent Systems - Modeling, Interactions, Simulations and Case Studies, Dr. Faisal Alkhateeb (Ed.), ISBN: 978-953-307-176-3, InTech, Available from: http://www.intechopen.com/books/multi-agent-systems-modeling-interactions-simulations-and-case-studies/a-multi-agent-based-multimodal-system-adaptive-to-the-user-s-interaction-context

INTECH
open science | open minds

**InTech Europe**
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

**InTech China**
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821