Toward a Generic Framework for Ubiquitous System

Amara Touil, Etienne Pardo, Jean Vareille, Philippe Le Parc

Toward a generic framework for ubiquitous system

Amara Touil, Etienne Pardo, Jean Vareille, Philippe Le Parc
Université Européenne de Bretagne, France
Université de Brest
EA3883 LISyC - Laboratoire d'Informatique des Systèmes Complexes
20 av. Victor Le Gorgeu, BP 809
29285 Brest Cedex - France
amara.touil; etienne.pardo; jean.vareille; philippe.le-parc@univ-brest.fr

Abstract

In this paper we present a beginning work about industrial applications using WEB technologies. Systems to study are, for example, robot arms in factories, Heating Ventilation and Air-Conditioning (HVAC) systems for commercial center or buildings, water distribution networks or power management consumption systems of corporate.

WEB technologies give us new opportunities to collect the data, to analyze correlations of signals and external events, and finally to change in "soft real-time" the parameters of the managed system. But these applications can be strongly influenced by the behaviour of the communication network and its reliability. We describe the key points that we will explore in our further work.

1. Introduction

Ubiquitous computing is generally sump up as "any time anywhere". It means that, in a close future, we would like to be able to perform remote actions to control different systems using networks, computers and software.

Although the underlying technologies are now quite mature, although a lot of experimentations have demonstrate the feasibility of remote operating, scientists are now facing a new challenge: transforming the dream of ubiquity to reality, moving from demonstrations to mass production tools and also making it possible to more people to use these new ways of interacting.

In this paper, after a description of related works (section 2), we will describe the way we are considering ubiquitous computing and the interaction between Humans and Machines (section 3). We will then propose three topics that will have to be studied to improve applications: security and safety aspects of networked control (section 4), use of virtual or augmented reality to provide high level interface to users (section 5) and definition of high level models to be able to simply specify ubiquitous systems to automate their development (section 6). Last section will conclude this work and propose future directions (section 7).

2. Related works

Distributed computing technologies, networking control and supervision, are extensively used as in industrial process as in ubiquitous systems. In literature, many concepts raised in ubiquitous systems conception and deployment. We are interested to four items: generic frameworks for ubiquitous systems conception and implementation, virtual reality paradigm for these systems, security and data management.

Concerning the first item, there are many existing approaches for generic frameworks. Those frameworks depend on specific requirements and operational functions. They are then limited in their genericity and reusability and blocked by categorical aspect. For example, in the field of robotics, there are many attempts to provide frameworks that will propose the ability to describe the most important robotics categories[13] [15]. In Yuan [25] and Amoretti [1], a real-time software framework is presented to improve the scalability and reusability of software modules. This framework is CORBA middleware design-based and includes several reusable services. Avraam [3] gives a generic framework for tele-control applications experimented on two applications: telemedicine and home security control. This work gives some important aspects like high level models for this category of ubiquitous systems.

About the second item, use of virtual reality, Sabater [22] paper presents a dynamic virtual environment to test teleoperated system with time delay communication. This approach makes it possible to verify and simulate the ubiqui-

tous systems architecture coupled with virtual reality. And as pointed by [10] and [19], virtuality and ubiquitous devices will affect day-to-day behaviours. Ubiquity means (virtual) presence somewhere else in space, but also in time, a concept still hard to manipulate. More generally, ubiquity can be interpreted as the consequence of user-device's interaction, and how it feeds back control. Therefore the medium should not only be adapted to its user, but also to its use. Nevertheless, simulations are closed to specific domains and dedicated first of all to training goal.

About the third item, security, some works trend to give adequate approaches to safety and security of ubiquitous systems. Sauter [23] studies a method based on system structural analysis to provide fault detection and isolation conditions. The goal is to make it possible the use of tolerant fault control using compensation mechanisms: Fault Detection and Isolation, and Fault Detection Control (FDI/FDC). Furthermore, robust fault detection where studied in [16] using soft computing techniques. The main idea is to show how to use bounded error approaches to determine the uncertainly of soft computing model like neural and neuro-fuzzy networks. Ruan He [12], study ubiquitous environments protection and security requirements diversity. He proposes an integrated solution for self-protected system and presents some preliminary results concerning an end-to-end architecture, and its corresponding implementation. More some validation tools like model checking [24] may be used and generic ubiquitous system frameworks validation is under research.

The last item, focus on data management in ubiquitous systems. In literature we find some related works dealing this item. For example, Christopher [5] presents a file system for ubiquitous computing applications that is context-aware (context is: time, location, and situation) in order to facilitate the use of the computational environment. His system is evaluated as part of an ubiquitous computing infrastructure deployed in a seminar room to investigate issues of performance, scalability, and usability. Rafi [21] thinks that the main limitations in ubiquitous systems projects are the ad hoc specificity of data management, and gives a model helping to describing strengths and weaknesses of ubiquitous systems and to compare it to other systems.

3. Ideal ubiquitous system definition

We consider mostly the interaction between humans and machines, and secondary the interaction between humans or between machines. For us a machine is a device situated at the end of the system, acting and remotely controlled. A human is situated in general far from the machine, but he could be near, and in some case we can have humans near the machine and other far from it.

So we can describe four cases of interactions as in the table 1.

Interactions	Humans	Machines
Humans	H2H	H2M
Machines	M2H	M2M

Table 1. Humans and Machine interactions

The interaction H2H (humans to humans) are like telephony or videoconferencing when humans are far but could be a face to face discussion when humans are near. Obviously when humans are far, their interaction needs devices to communicate. So we should write H2D2N2D2H where D means one device like a phone and N a communication network. Now we can introduce our first target: how to reduce the difficulties to use the devices and the network and to reach the transparency of the usage for the ordinary user. When the transparency is achieved then we can diminish the expression in H2H, and forget the D2N2D in between. In the same way we reduce the H2D2N2D2M in H2M and so on.

The M2H case corresponds to the feedbacks sent by the machines to the user or by cameras, microphones or other sensors situated in the environment of the machines. It is the case of the supervisory control.

The H2M case is the most important case, it corresponds to the human control of the machine. In this case the first device near the user displays the human machine interface.

The M2M case corresponds to the communications between the machines.

In the real world it is impossible to send data without delays of transmission [15]. The most important difference between the data management in distributed systems and the remote control of machines is that the real world around the machines continues to change [18] when the communication is delayed or broken. If we would like to design an ubiquitous system first we have to evaluate where the delays are and what are their scales. After that we have to know, in term of delays, what are the limits of the human perception and reaction.

The limits of round trip time (RTT) easily accepted by the human [4] are:

- less than 350 ms for phoning,
- 200 ms for teleoperation with force feedback,
- 20 ms to 40 ms to play music simultaneously depending on the frequency and the power.

The physical delays available for digital transmissions are:

- in optic fibbers the speed is near 200 000 km/s, it means 5 s / km, or 1 ms / 200 km

- in the air the speed is near the light speed within 300 000 km/s (299 792 458 m/s exactly) it means 1 ms / 300 km

- in wires the speed is less than the light speed in the vacuum but can reach this limit.

We must add the physical delays, the delays of commutation, the delays of the computations, and the delays of the conversions analog/digital of the signals collected by the sensors.

Commonly we suppose that the knowledge of the global behaviour of the whole system is a composition of the behaviours of each part. But in the case of the ubiquitous system using wide area networks of communication the behaviour of the communication through the network cant be controlled neither by the user nor by the controlled machine. The whole long distance communication is supported by an external access provider, the obligations of the provider are written in the contract. In our case a permanent connection is needed for the ideal system without delay to repair because the quality of experience (QoE) is the principal target in H2M interactions. Commonly the provider propose services like VoIP, access to Internet, network games, etc. When one of these services is the VoIP we can suppose that the RTT will be less than 350 ms on the whole earth, and that the jitter is compatible with the real-time codec used for the phones. Frequently we can read in our contract that the VoIP communications are guaranteed for a maximal duration of 2 hours, and in the case of failure of the network that the minimal time to repair is two days. For an ubiquitous system, interruptions of the service or an arise of the delay could have bad effects on the machines or the environment around, because the stability of the process in some situations can't be insured when the delays exceed a threshold value. The conclusion here is that when we use real networks without quality of service (QoS) and when the communications use external links, the ubiquitous system can have behaviour of a complex system.

During the design process of an ubiquitous system we have to take into account this aspect and to introduce safety modes. This aspect will be developed in the next section.

Another important point is that the difference between the ideal system and the real system can't be avoided. The result is that the needs of formation before first use and to simulate the behaviour of the whole system are very important. To do that, we can use virtual models of the system usable in a virtual reality environment. This aspect will be developed in a further section.

The growth of the Web 2.0 develops today some pre-existing modes of continuously refreshed communications like for example the "push" technology to display videos or RSS streams to broadcast news. These modes have some similarity with the usage of tags in the supervisory control. But they can't be directly used for ubiquitous systems because they don't integrate information sent back from receivers to senders about the delays of the transmission.

4. Safety and security of ubiquitous systems

Ubiquitous systems require strong and flexible protection, due to their highly dynamic character, and to the diversity of their security and safety requirements. Looking to this issue, we will focus on two aspects. The first one is how to secure ubiquitous systems access and data management against unauthorized people? And how to preserve a good operation mechanism itself? The second one concerns how ubiquitous devices react and recover non specific medium failure? We mean by specific medium all general infrastructural networks like Internet, WIFI, Zigbee... that we can't control.

For the first issue, there are some approaches like making security space flexible itself and automating reconfiguration of the protection mechanisms. In this case system will be self-protected without any user intervention. In [12], Ruan He uses a component-based software paradigm to design and implement such systems. This work focus on self-protected pervasive systems in terms of a 3-level architecture containing two control loops at the network and node levels. He applies this approach to some scenarios like military surveillance. In [9], Fukase study the environment demands on security, speed, and power in processing huge amount of multimedia information. His solution is to construct and implement a safety aware. To secure authentification and unfriendly usage, Fukase study systems composed of a hardware cryptography-embedded multimedia mobile architecture and its software support. The two works focus on specific network characteristics and low layers.

The second issue concerns how to guarantee safety and security according to authentification, data management and communication reliability throws non dedicated networks (like Internet for example). Although communication medium are often considered without failure while building and using remote applications, possible defaults have to be taken into account because we may be sure that they will happen. That's why, we have to build a methodology including predictive and preventive strategies ; cutting off communication between user and device is an example of frequent scenario that can happen, in this case device must have a recovery procedure for this problem. Other, the user can continue to monitor the functioning device from a virtual reality mechanism. After failure recovering, user and device will be synchronized on real monitoring. Our work is based on a tele-robotic platform [15] [26] we already implemented. Many tests have been realized on network connection reliability and failure recovery. As example, a direct teleoperation architecture with Speed Limit

Module (SLM) and Delay Approximator (DA) were proposed in [26] to avoid unpredictable Internet delays and possible connection rupture, in a case of a mobile robot. Results showed that this approach guarantees the path error of the continuous robot motion. Moreover the current time delay is supervised by DA module, and SLM applies different speed limit rules according to the current time delay situation. Then the robot is always running in a proper speed which meets the path error restriction. This experimentation shows a recovery approach to face Internet network failures between one device and one controller, our future goal will be built around this approach to face those problems in a distributed ubiquitous systems.

5. The paradigm virtual reality, augmented reality, man-machine interface and training

Ubiquitous computing systems purpose —at least in domotic— is to serve mankind. This requires such systems to be able to cooperate efficiently with humans. In [11], authors point the need in humanised device for its presence to be accepted. Thus the use of natural language, gestual communication, standard context-sensitive approximation and such are that much points an interface between devices and users should take in account. Moreover, this interface should be simple (easy to build), natural (easy to use), efficient (it mustn't restraint by its nature the possibilities) and powerful (it should facilitate high-levelled solutions' conception). Although there is no perfect medium responding to all this, there are solutions that cover wide panels of possibilities. Amongst these media, lies Virtuality and its most common usage: Virtual Reality and Augmented Reality.

Freund [8] proves that Virtuality can effectively be used as an interface between humans and machines. In the Projective Virtual Reality the machines are abstracted, while the user's actions are identified and divided. This way, users do not feel the constraint of giving order to specific device, the sub-actions being dispatched adequately. Although it is possible to create it for specific use, more generic non-limiting interfaces are still difficult to conceive. Users have their habits, each user's different from the others. So the medium must adapt itself to the user, so as to simplify specification of standard behaviours (that satisfy most of the future users). Liao [17] presents the use of an adaptive interface, multimedia oriented, based on Augmented Reality. Through training, the device learns to swap between different interfaces, execute actions' sequences... automatically. Such auto-learning systems resolve partly the inner-intelligence of ubi-comp devices. However, no matter their smartness, full automated houses tend to be rather stupid. The main cause is that each evolution in technology induced behaviour has drawback, negative or not, that takes time to be accustomed to.

However, virtuality can not only be used as a remote-device's control medium, but also, as stated by Michel [20], as a remote-device's conception medium. By simulating a "real" environment, from daily standard to least possible situation, and analyzing the behaviour of the "would-be" remote device, it saves money and time. And by allowing simulations with many devices, it simplifies the elaboration and validation of cooperation / communication protocols. This usage as a simulation display medium enables also consequences' expectations of actions. The time factor being manipulable in simulation, the resulting state can be predicted. Therefore, scenarios can be prospected at the users' want, fasting or jumping low interest actions, and slowing important ones.

The conception and design of such systems (be they interface or conception) require adapted, standardized, frameworks. This way, each device can interact with others, and users can use it, efficiently.

6. Ubiquitous model proposal

In the first paragraph of this section, approaches to specify languages and frameworks for ubiquitous systems are presented. In the second paragraph, a first modeling level is given to show our approache. The last paragraph introduces tools and methodologies used to implement instances of an ubiquitous system in order to generate applications (executable code).

6.1. Model Driven Engineering(MDE) approach

For long time, man was looking for a suitable reality presentation according to his mature perception. Earliest civilizations have shown us some models: drawing animals, tree, etc... Others have shown scenes of hunting, war, etc... A little further, according to science development, mathematical models, and even physical models of society appeared. The model paradigm is not a new concept and we have always tended to model and to construct an image of our world, following specific concerns. To integrate complex systems into a development process, a normative framework is needed to specify systems characteristics and mutual dependencies. This trend is supported by many groups like the Object Management Group (OMG), who creates an Unified modeling Language(UML) in order to gather designers around the same approach.

Ubiquitous systems face rapid growth and include more and more devices and components with different internal architectures and from different constructors. To ensure interoperability between these various components, to ensure the ability to reuse components and to ensure rapid development, a generic framework is required.

There are some approaches around the concept of framework for ubiquitous systems, based on Distributed Data Control (DDC) architecture or in Supervisory Control Data Acquisition (SCADA) architecture. These approaches are generally dedicated to specific areas and domains like robotics, tele-operation and home automation. They cover some other ubiquitous system area but component reusability is very limited and their genericity is weak [3]. Moreover, security and safety operation for systems depend on communication mediums quality and characteristics that aren't often taken into account when designing ubiquitous systems.

To build and implement ubiquitous systems, we have to take in account the diversity of used components and communication medium. To manage this diversity we have to focus on high abstraction and generic modeling based on high level model approach that ensures a representation of different ubiquitous systems dedicated to different platforms. Genericity offered by this approach will enable the representation of dependencies between different components together.

The foundation of our approach will be consistent with the emergence of new standards modeling language such as:

- *Architecture Analysis and Design Language*(AADL) [14]. This language employs formal modeling concepts to describe and analyze application system architectures. It includes abstractions of software, computational hardware, and system components for specifying and analyzing real-time and high dependability systems. This duality between software and hardware permits the mapping of software onto computational hardware elements.

- *UML language profiles*. There are three important UML profiles useful for our approach:

  - the first one is *modeling and Analysis of Real-Time and Embedded systems* (MARTE) [7]: it includes software and hardware aspects and provides support for non-functional property modeling and adds rich time and resource models to UML. Moreover, it provides support for quantitative analysis (e.g. scheduling, performance).
  - The second one is TURTLE*Timed UML and RT-LOTOS Environment* (TURTLE) [2], which includes a real-time UML profile with a formal semantics given in terms of translation to RT-LOTOS, and a model validation approach based on this environment.
  - The last one is *Systems Modeling Language* (SysML) [7], which supports the specification, analysis, design, verification and validation of a broad range of complex systems. These systems may include hardware, software, information, processes, personnel, and facilities.

All these standards offer a useful large scale presentation and patterns in ubiquitous systems' specification, and also toolboxes. Turtle is dedicated to LOTOS environmen and deals with the same concepts as SysML or MARTE profiles, in real-time embedded systems hardware and software modeling and specification. Nevertheless, SysML adds some extra concepts like information, personnal, and facilities to design systems.

The current trends in software development have a "model-centered" approach that requires both: advanced in terms of media tools to change consistency models between the modeling languages and applications themselves, and also in defining process based development model. Model Driven Engineering (MDE) [6] responds to these trends in addressing the following locks:

- The transformation of models which includes the development, evaluation, code generation, validation, testing, tracing...

- The definition of models and modeling languages sufficiently precise to be "useful" and expressive enough to meet the application requirements including real-time, embedded and distributed systems.

- The support for collaborative aspects and concerns separation in development and implementation of complex systems such as ubiquitous systems.

The MDE approach makes models contemplative, so they can be manipulate during development process. This facility let us to design a framework that allows passage from meta-models high level independent platform (also called PIM : Platform Independent Model), to instantiated models of ubiquitous systems (known as PSM : Platform Specific Model).

6.2. Framework specification : main concepts

The prototype scheme of our framework (figure 1) is composed of 5 main components (also called packages in the following): device, medium, user, administrator and data. Packages are used according to following relationships:

- *Dependencies relationships* : according to the UML specification, dependency is a weaker form of relationship that indicates one element depends on another because it uses it at some point in time. The use of this relationship kind in our approaches means

that when specifying ubiquitous system packages need each other for the global functionality specification of the system. So the *"User"* need the *"Medium"* to communicate and collaborate with ubiquitous system components. *"Administrator"* and *"Device"* need also the *"Medium"* for the same reasons as *"User"*. We note that *"Administrator"* is a centric element for the system handling and organizing global collaboration and interaction between devices and users on the medium support. Signaling that dependency is at modeling level and not at runtime one, so this kind of need help us to have a generic point of view of global system dependency.

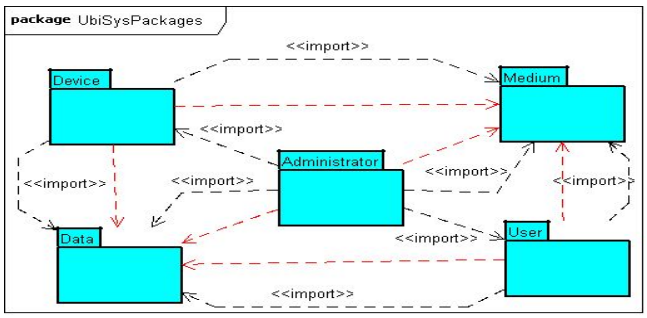

Figure 1. Basic framework packages for ubiquitous system specification

- *Import relationship*: is a kind of dependency that means the use of some elements from one package in an other one. So imported elements could be referenced or created in the important package. *Import relationship* is displayed in the diagram as a dashed line with an open arrow pointing from the imported package to the important one. The keyword *import* is attached to the connector.

To illustrate our approach concept, we give in the following paragraphs, the first modeling level for each package as metamodel. [1] classes design with the help of the Eclipse modeling framework core (ECORE). Metamodels shown present just the main concepts to keep the genericity aspect of our approach.

6.2.1. Device package

This package will describe devices properties, behavioral scheme, and constraints defined in their specification and obtained as much as possible from constructors.

[1] Metamodel si a modelisation concept that leads to define a model according to its artifacts. The model is conform to the metamodel and will be placed in a bottom level than the metamodel.

In the class diagram presented in figure 2, the main element is the *"Device"* one. Other elements are needed to specify devices concepts for ubiquitous system that belong directly to the Device package or imported from other packages.

Starting by imported elements to show dependency degree at this level of modeling, each device could awn a context specification(*"Context"* element) like location, profile, action... imported from *"Data"* package. This generic element is instantiated directly from this package according to the containment type of association *"itsContext"*. The second imported element is the *"MediumConnection"* from the *"Medium"* package. This element is the bridge between device and his external communication environment composed by all ubiquitous system elements.

Looking for the owned elements in device Package, *"Device"* class is the main component for this level of modeling. Depending on its type (mentioned by *"Type"* element), device could perform a computing operations and treatments (that is not shown in this level) related to its general context if it has autonomous working. *"Behavior"*element offers a customized behavior for the device : device behavior uses data and information given by *"Context"*. associations references could be st in a bottom level between contained elements in *"Context"* and *"Behavior"*. Some relations are left for more detailed level to give more conceptual choices.

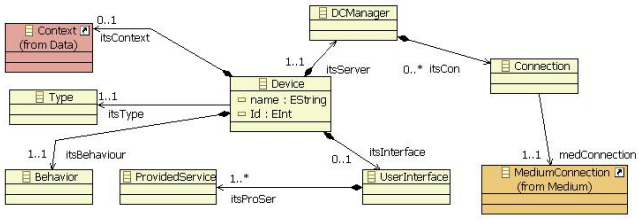

Figure 2. The device package

Device communication and collaboration in ubiquitous system could be specified by a set of several elements: *"DCManager"* witch handles device connection with other devices or users included in the ubiquitous network. This ability is done throw *"itsCon"* roles associated to *"Connection"* element that references imported element *"MediumConnection"*. Device exchanges information and data with its environment through an interface(*"UserInterface"* element) that gives provided services (*"ProvidedService"* element) for user. We mention that in some cases and depends on freedom degree of device autonomous aspects, device could react according to some defined scenarios independently from user commands (M2M concept). In this case designer could not specify a user interface (cardinality for *"itsInterface"* association is zero or one).

All these elements are generic and lead to cover the most ubiquitous devices specification. This metamodel could include other concepts depending on application requirements in lower level modeling.

6.2.2. Medium package

The use of *"Medium"* package in our framework concept has a general view according to the nature of ubiquitous systems. It means the communicating environment where surrounding objects interact under some conditions, or influences. The general purpose of the medium at this modeling level lets the choice for ubiquitous designer to specify what they need. In other words, use a normalized existing network infrastructure and application services, keep infrastructure and specify services... May be in few years, ubiquitous systems will have their own medium normalization that our framework will be able to specify their needs. In the following, we describe mainly components needed at this level of modeling.

To the best of our knowledge, ubiquitous system generally uses heterogeneous existing networks as connection medium. This package include needed specification for those networks like Internet LAN and WAN networks, Bluetooth , WIFI, Zigbee... at a bottom level that not directly mentioned here.

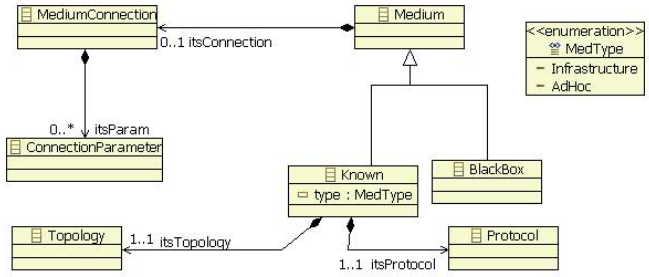

Figure 3. The medium package

Figure 3 shows a metamodel of medium concept with needed artifacts for communication. The *"Medium"* element is associated to *"Connection"* with *"itsConnection"* role to gather devices, users and administrator around ubiquitous system. Each connection has some parameters to deal with actors need. On the other side, the medium could have two inheritants elements :

- *"BlackBox"* : for unknown medium that we don't know communication paths, routing points, physical infrastructure,... like Internet network. We just know how to access and communicate according to provided *"ConnectionParameter"* without possible control.
- *"Known"* : for this kind of medium we know the most important criteria for the network communication such as protocols, topology... and we have access to its properties for customizing like local networks. *"Known"* network element as defined in this modeling level, allow designer to specify his own medium for ubiquitous systems like type choice (infrastructure, ad hoc), protocol, topology...

Medium specification is very important in ubiquitous system, so all medium features are taken in account to implement devices and users strategies. The administrator (presented latter) is the most important actor for handling and organizing medium connection and data exchange.

6.2.3. User package

This package gives users commands and control functionalities. It will make it possible to specify all needed procedures to communicate, to exchange data with other ubiquitous system element. First level metamodel of this package is shown figure 4.

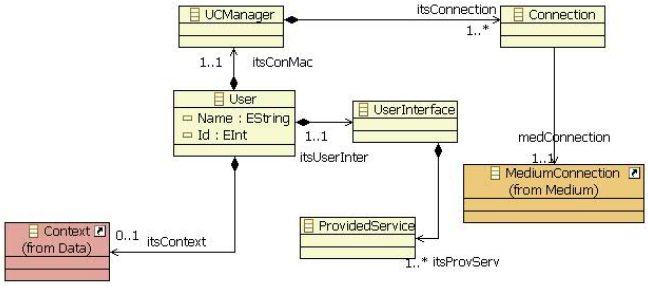

Figure 4. The user package

"User" is the centric element; it has *"Context"* which could include data, strategy, processing informations,... imported from Data package like the *"Device"* in figure 2. The communication and services (*UCManager, Connection, UserInterface, ProvidedService* elements share the same abstract level as *"Device"* elements but their use may be different depending on their use and environment.

A first look to this metamodel can let us mistake user for device, or consider the user as a particular device. Actually, autonomous device is an intelligent one and could behave as a user in a M2M approach. The fact that user is separated from device, is that we need this concept for virtual reality and augmented one. User could use other device to control distant one or interact directly in a cognitive communication. This user concept facilitate pervasive ubiquity concept.

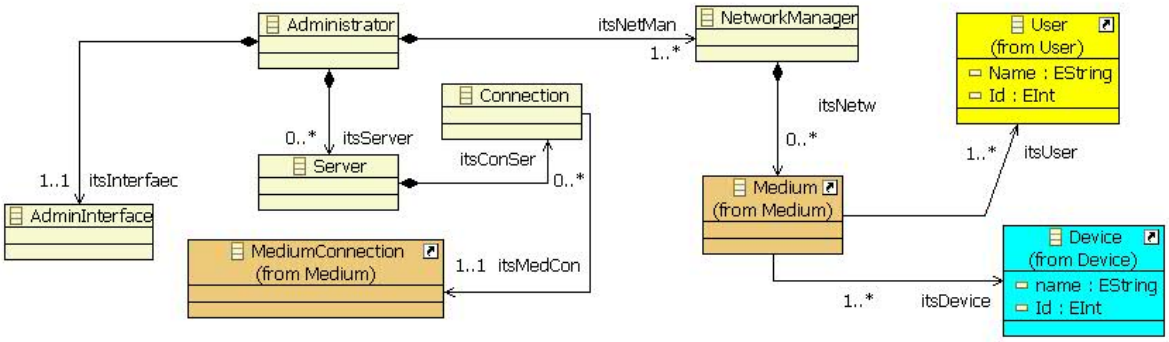

Figure 5. The administrator package

6.2.4. Administrator package

The Administrator package includes network configuration, and management and exchange of data between all actors of an ubiquitous systems. The *"Administrator"* element is the main concept.

The first decomposition level of the metamodel of this package is shown figure 5. *"Administrator"* is a specific user that has extra opportunities to handle the whole ubiquitous system. It has an administration interface *"AdminInterface"* for all operations: management, control, supervisory... for all ubiquitous system actors.

Administrator has the ability to manage the whole network through *"NetworkManager"*. This element allows medium instantiation (*"Medium"*, imported from Medium package). User and device are instantiated separately and referenced through associations (*itsUser, itsDevice*).

Note that the term *"Data"* is hide at this level of modeling. In fact, the administrator use his own data and given data from ubiquitous system actors for processing, storing, exchanging, managing,...

6.2.5. Data Package

Data in an ubiquitous system is like blood in a human body, it presents all the informations circulating in the network. The data package includes data and all the related processing technics like data management profiles that control, protect, deliver and enhance the value of data, and information assets.. Strategies and scenarios that guarantee reliability and maintainability between all ubiquitous systems actors are also included in processing data.

The first level of decomposition of the metamodel of this package is shown in figure 6. At this stage of modeling, security and surety are presented in the *"Strategy"* element. The latter could have some scenarios policies related to devices, users and administrator.

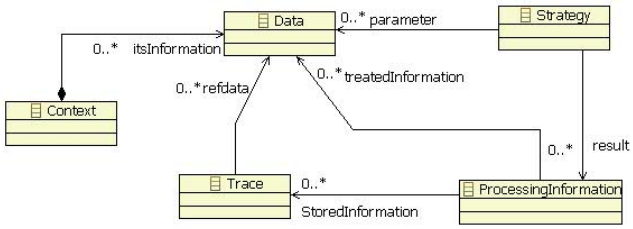

Figure 6. The Data package

The main element of this metamodel *"ProcessingInformation"* that could include data processing and managing tools. *"Data"* element is referenced as *"refdata"* in trace concept(*"Trace"* element) to save information history. Stored informations are referenced by *"ProcessingInformation"* and used as libraries elements.

In pervasive ubiquity, context concept is very important, its strongly related to the data concept. So *"Context"* element is placed in this package and exported to other ones as shown in last paragraphs.

6.2.6. Conclusion

Presented metamodels in this section show just one level of modeling specification. Other ubiquitous system concepts are included at lower level stage, and some could be added directly in framework repository as metamodels or imported as libraries.

The specification of our ubiquitous system and its implementation will be handled in future work, where we explain processing scenarios and different steps adopted according to adequate tools.

7. Conclusion

Ubiquity will be a new way of living in the future: being in one place, acting on systems located in another city, country or continent. Ubiquity will rely on information technologies: electronic devices (computer, phone, haptic systems...), networks (from long distance optical link to short distance wireless link) and software frameworks (managing communication protocol, using distributed software, providing Human-Machines Interfaces...). Ubiquitous systems will be complex systems to create and to manage in order to ensure not only remote action on devices, but safe distant control on complete systems.

In this paper, we tried on section 3 to define a kind of "ideal" ubiquitous systems and we focused after on three concerns that are, according to us, the most important to take into account. As communication failures will happen, we have to design systems that integrate that kind of risk. As these systems are complex, we need to use development methodologies to design systems that may be easy to build, to simulate and to prove. As user's adoption is a key point of the development of ubiquitous systems, we need to provide interface that will reduce the distance from the user to the system, and virtual reality may be used for this purpose

Data management is also a great concern in ubiquitous systems and will have to be included more in our work.

After several experiences in developing remote control systems, from real devices to efficient web based applications, we will try to have a top-to-bottom approach, from models to application.

Acknowledgments

This beginning work is supported by the town community of Brest "Brest Métropole Océane" and performed in partnership with the company Terra-Nova Energy. We thank them for their help.

References

[1] M. Amoretti, S. Bottaazzi, S. Caselli, and M. Reggiani. Telerobotic systems design based on real-time corba. *Journal of Robotic Systems*, pages 183 – 201, April 2005.

[2] P. Apvrille, L. Courtiat, J. Lohr, and C. de Saqui Sannes. Turtle : a real-time uml profile supported by a formal validation toolkit. *IEEE Transactions on Software Engineering*, 30(7):473 – 487, July 2004.

[3] C. Avraam and A. George. Implementing a generic component-based framework for telecontrol applications. *Software Practice*, (37):1087 – 1132, February 2007.

[4] F. Bérard. *Vision par ordinateur pour l'interaction homme-machine fortement couplée*. PhD thesis, Université Joseph Fourier, IMAG, Grenoble, 1999.

[5] K. Christopher and H. C. Roy. A context-aware data management system for ubiquitous computing applications. *IEEE Computer Society*, (3):294 – 303, 2003.

[6] Douglas and C. Schmidt. Model-driven engineering. *IEEE Computer Society*, pages 25 – 31, february 2006.

[7] M. Faugere, T. Bourbeau, D. Simone, and R. Gerard. Marte: Also an uml profile for modeling aadl applications. *12th IEEE International Conference on Engineering Complex Computer Systems*, pages 359 – 364, July 2007.

[8] E. Freund and J. Rossmann. Human-robot collaboration: A literature review and augmented reality approach in design. *ICAR*, 1997.

[9] M. Fukase, H. Takeda, and T. Sato. Hardware/software co-design of a secure ubuquitous system. *2006 International Conference on Computational Intelligence and Security*, 2:1307 – 1310, November 2006.

[10] A. Goradia, N. Xi, and I. Elhajj. Internet based robots: Applications, impacts, challenges and future directions. *IEEE Workshop on Advanced Robotics and its Social Impacts*, pages 73 – 78, 2005.

[11] S. Green, M. Bilinghurst, X. Chen, and J. Chase. Human-robot collaboration: A literature review and augmented reality approach in design. *International Journal of Advanced Robotic Systems*, 5(1):1 – 18, 2008.

[12] R. He and M. Lacoste. Applying component-based design to self-protection of ubuquitous systems. *Proceedings of the 3rd ACM workshop on Software engineering for pervasive services*, pages 9 – 14, february 2008.

[13] A. Hentout, B. Bouzouia, and Z. Toukal. Multi-agent architecture model for riving mobile manipulator robots. *International Journal of Advanced Robotic Systems*, 5(3):257 – 268, 2005.

[14] J. Hudac and P. Feiler. Developping aadl models for control systems: A practitioner's guide. *Technical Report CMU/SEI-20066TR-019*, October 2006.

[15] L. Kaddour el Boudadi, J. Vareille, P. Le Parc, and N. Berrached. Remote control on internet, long distance experiment of remote practice works, measurements and results. *International Review on Computers and Software (IRECOS)*, ISSN 1828-6003, May 2007.

[16] J. Korbicz and M. Witczac. Uncertain soft computing models in robust fault detection. *1st NeCST workshop*, pages 199 – 204, October 2005.

[17] C. Liao, Q. Liu, D. Kimber, and S. Lertsithichai. A telerobot assistant for remote environment management. *IEEE International Conference on Multimedia and Expo*, 2004.

[18] J. Ma. Smart u-things challenging real world complexity. *J. Ma, IPSJ Symposium Series Vol. 2005, No. 19, Proc. of DPSWS13*, ISSN1344-0640:146 – 150, November 2005.

[19] T. Mahler and M. Weber. Mobile device interaction in ubiquitous computing. *Advances in Human Computer Interaction*, pages 313 – 330, October 2008.

[20] O. Michel. Cyberbotics ltd - webotstm: Professional mobile robot simulation. *Advanced Robotic System*, 1(1):39 – 42, 2004.

[21] L. Muhammad, R.and Young Koo and L. Sungyoung. Knowledge management framework for ubiquitous systems. *IEEE International Conference on Management of Innovation and Technology*, 1:412 – 416, June 2006.

[22] J. Sabater, J. Azorin, O. Reinoso, R. Neco, and N. García. Dynamic virtual environment to test teleoperated systems with time delay communications. *Journal of Robotic Systems*, 22(4):167 – 181, April 2005.

[23] D. Sauter, T. Boukhobza, and F. Hamelin. Decentralysed and autonomous design for fdi/ftc of networked control systems. *1st NeCST workshop*, pages 79 – 85, October 2005.

[24] C. Yean Ru, L. Yen-Hung, and H. Pao-Ann. Model checking safety-critical systems using safecharts. *IEEE Transactions on Computers*, 56(5):692 – 705, May 2007.

[25] K. Yuan-hsin and A. MacDonalds. A distributed real-time software framework for robotic applications. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 1964 – 1969, April 2005.

[26] S. Zhong, P. Le Parc, and J. Vareille. Internet-based teleoperation: a case study. *Fourth International Conference on Informatics in Control, Automation and Robotics (INCINCO'07)*, 1:267–270, Mai 2007.