A Survey of adaptation systems

Keling Da, Marc Dalmau, Philippe Roose

▶ **To cite this version:**

Keling Da, Marc Dalmau, Philippe Roose. A Survey of adaptation systems. International Journal on Internet and Distributed Computing Systems, 2011, 2 (1), pp.1-18. hal-00689773

HAL Id: hal-00689773
https://hal.science/hal-00689773v1

Submitted on 20 Apr 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Survey of adaptation systems

Keling DA #1, Marc DALMAU #2, Philippe ROOSE #3

UPPA, LIUPPA, IUT de Bayonne
2, Allee du Parc Montaury 64600 Anglet FRANCE

[1] kda; [2] dalmau; [3] roose@univ-pau.fr

***Abstract*—Development of ubiquitous applications is inherently complex. Adaptation system is a solution for ubiquitous computing. It enhances the efficiency of application by the adaptation of software, facilitates application development, and offers a good user experience. Adaptation system is faced with challenges of different research domain including context modeling, situation identification, context reasoning, and adaptation decision. In this paper we discuss the architecture design of adaptation system and the taxonomy of its key technologies in details including communication middleware, context management middleware, adaptation middleware, adaptation platform, application model and software engineering, we analyze and introduce these technologies with the most well-know ubiquitous projects. At the end, we introduce future research directions according to structural adaptation.**

***Index Terms*—Ubiquitous computing, Adaptation, Context-aware Middleware, Dynamic reconfiguration**

I. Introduction

"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it." Mark Weiser [112]

Ubiquitous computing[1] was proposed by Mark Weiser in September 1991. "Ubiquitous computing is the method of enhancing computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user" [113].

According to the definition of ubiquitous computing, which must be pervasiveness, convenience and adaptable. The future applications of ubiquitous computing face with a heterogeneous and dynamic environment. They must be able to adapt and react dynamically to the environment (heterogeneous hardware and software environment) and context [84] (user and environment context).

Currently, increasing numbers of personal smart-devices (e.g. iPhone, Android smartphone and tablets etc.), home smart-devices (e.g. Smart-TV, Cook robot, etc.) and small smart-devices. These devices have powerful computing resources and network connectivity (e.g. 3G or 4G network, Urban Wi-Fi network). We now stand at the beginning of the "one person, many computers" age, which is called the third wave[2]. At the software level point of view, these smart-devices are still individual and their resources didn't have an efficient high-level management. According to the requirements of ubiquitous computing, the future software must be adapted to the environment and to the user needs. Adaptation systems are one solution to adapt software to the environment in order to achieve ubiquitous computing.

Adaptability raises complex scientific problems and new challenges to the software development and execution. For example, how to collect the context information in a highly dynamic and distributed environment, how to adapt application to a mobile environment in order to provide good performances, how applications can cope with heterogeneous infrastructures and fully benefit from the environments capabilities and so on. An adaptation system for ubiquitous computing should be able to provide proper solutions to the above questions. Adaptation systems in ubiquitous computing have four different roles: Context manager role, Middleware role, Adaptation plan provider role and Decisional role. Each role has different challenges that will be developed in the next section. Adaptation system acts on two levels: application level adaptation and system level adaptation. Application level adaptation means that the adaptation will happen only on applications. The adaptation system provides adaptation services to the application but the system itself cannot react to the environment. Web service-based systems act often at the application level adaptation. The problem is if a mobile device looses the network connection, it also looses all the adaptation services. System level adaptation means that the system is able to adapt itself as applications do. It is called Adaptable system or Autonomous platform like is the MUSIC platform [85]. The system level adaptation allows the system itself to run on each device not just on a server. It can provide more powerful and more flexible adaptation services, because it already is inside the execution environment.

The goal of this article is to do a survey of the challenges of adaptation platforms for Ubiquitous Computing. The paper is organized in five sections. We begin with presenting an overview of adaptation systems. Next, we present an adaptation systems state-of-the-art. Section 4 discusses the future requirements of adaptation platforms, and concluding remarks are in section 5.

II. Overview of adaptation system

This section describes the main principles of software adaptation for ubiquitous computing. At the beginning of this adaptation platform's survey, we will present what means adaptation in pervasive computing. Next, we will discuss adaptation platforms and their role in ubiquitous computing.

[1] Ubiquitous computing: now also called Pervasive Computing or UbiComp. In this article we use the terms 'ubiquitous' and 'pervasive' interchangeably.

[2] The wave in computing: the first ware is one computer, many people; the second wave is one person, one computer.

At the end of this section, we will present some related approaches.

A. What is adaptation?

The word adaptation means "Any change in the structure or functioning of an organism that makes it better suited to its environment."[3]. Therefore, according to this definition, the software adaptation has two levels of action: changing the structure and changing the function. These changes aim to make the application better suited to its evolving context. The context means the operating environment and user's utilization environment. The operating environment includes any information observable by the software system, such as end-user input, external hardware devices and sensors, program instrumentation, and network infrastructure [76]. The user's utilization environment means anything about the user, which includes user's intents, user's context information.

1) Why and when we need adaptation?: There are three raisons to adapt application [25]. Each raison defines an adaptation type. These three raisons are: Reactive adaptation, Evaluative adaptation and adaptation for integration. Reactive adaptation accommodates applications to changes in their environment. They change the behavior of the application according to execution environment changes due to context or user preference change. Evaluative adaptation aims to extend the functionality of an application, to correct its errors or to increase its performance. When the QoS (Quality of Service) change, it may launch an evaluative adaptation. The user's preferences changes also can launch such adaptation. Adaptation for integration deals with the problem of integration of incompatible services or components (i.e. different hardware or software interfaces, different protocols and so on). This type of adaptation is often launched after an adaptation decision has been done. For example, the first adaptation is launched by context changes. Due to the execution of this adaptation, a component can no more communicate with another one because they have different security protocol. In this case, the system will launch an adaptation for integration.

Moreover, there are three ways to decide when to launch an adaptation: Context-driven, QoS-driven and User-driven. They often work together, but with different priorities in different situations (depending on adaptation motivations and application intents). Context-driven analyzes all the context information and takes the decision when it receives a context-changing event. Reactive adaptation aims context-driven analysis. QoS-driven works on QoS events and it is more concerned by the quality of the whole system. In the evaluative adaptation, QoS-driven has priority. User-driven adaptation only concerns about users' modifications of system options or users' preferences. It generally has the highest priority.

2) Adaptation loop : The general adaptation loop of an adaptive software system includes the environment observation, the selection of adaptations and their execution. We call it the CADA loop (Collection, Analysis, Decision and Action) in this article. Figure 1 describes the adaptation management.

[3]The adaptation definition comes from the Oxford Dictionary of Science.

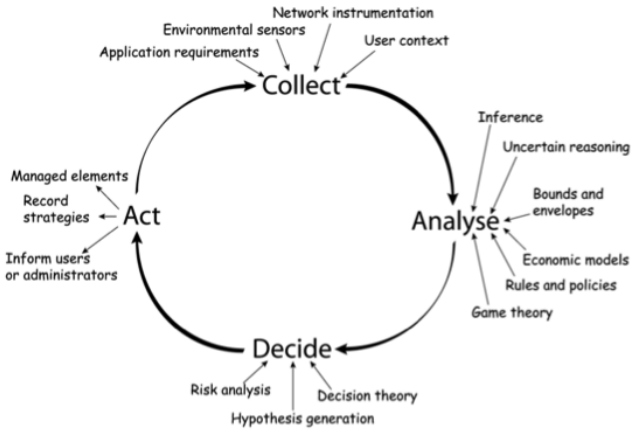

Fig. 1. CADA Adaptation loop [33]

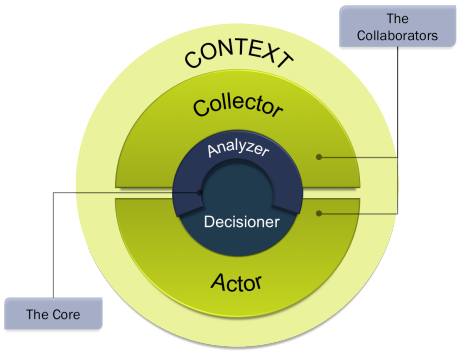

Fig. 2. CADA general Architecture

In this feedback loop, human can be involved or the system can be fully autonomous.

The context collector first collects information on the operating environment from the operating system and the user context. This information is called Meta context information. Then, the collector exploits this Meta context information to build the high-level context model. This high-level context model is a highly abstract model that represents the context information. The analyzer evaluates the context information and produces an adaptation plan. It can produce a list of plans or just one plan. The decider analyses the risk of each plan and takes the adaptation decision. Finally the Actor executes the adaptation as described by the chosen adaptation plan. The core of adaptation is the Analyzer and the Decisioner. Collector and Actor are only collaborators. The core is like the human brain, and the collaborators are like the sensory nerves and limbs. (see Figure 2)

In next two sections, we discuss different types of adaptation at different points of view. The first is the level of adaptation point of view and the second is the realization of adaptation point of view.

3) Self-self vs Supervised Adaptation: At the level of adaptation point of view (i.e. adaptable software architecture), there are two types of adaptation.

Self-self-adaptation: application manages the adaptation. The core and all the collaborators are included into the application. This kind of autonomous applications adapts itself

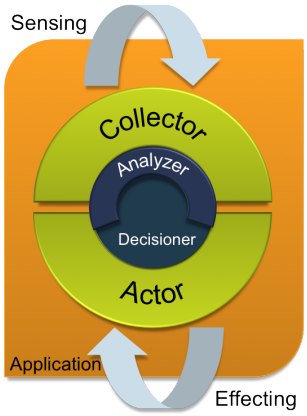

Fig. 3. Self-self approaches of adaptation

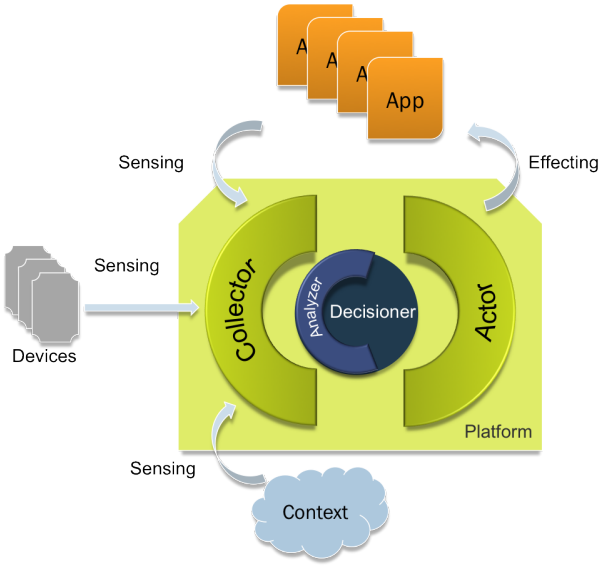

Fig. 4. Supervised approaches for adaptation

without external support. (see Figure 3) It needs neither adaptation platform nor adaptation middleware (usually the middleware supports a part of the adaptation service, like the context collector, and the adaptation actor,). The self-self-adaptation application developer cannot easily reuse its adaptation code. Each application proposes its own adaptation solution. Therefore, in this article we don't consider such adaptation.

Supervised adaptation: the platform manages adaptations. (see Figure 4) Adaptation is transparent for applications. Platform may include the core and all collaborators or just the core (a Decisioner and an intelligent analyzer) then the core uses the collaborators that are offered by third part providers. The core provides intelligent analysis and decision. And usually, the collaborators are provided as services. It can be a combination of collaborators. For example, the platform can have one local context collector, which uses different context collectors to achieve the users' context collection. These collaborators take care of the mechanism part of the context collection, communication, adaptation and so on. For example, middleware adaptation actor is in charge of all the transparency of distributed adaptation operations and of all cross platform/OS/language adaptation operations. It doesn't have any intelligence. The Supervised adaptation has the advantage that application developers do not need to be concerned by adaptation; they can focus on the application's business logic. The challenge of such platforms is about how to make the right decisions for all different applications in all situations.

4) Structural and Behavioral adaptation: The adaptation platform distinguishes two adaptation categories at the realization of adaptation point of view: structure and behavior changes [6]. Structural Adaptation: The structure of the program itself changes (program includes application, platform and middleware). For Component-based system, the change can concern the composition of components by adding or removing components. In addition the migration of components is included: changing the running host of a component and dynamically linking it to others components. The migration operation needs to keep the coherency of the components states before and after. For Service-based systems, the change concerns the configuration of services. Like SAMProc [96], some platform or middleware support service migration. Structural adaptation distinguishes two kinds of abilities: configurability and reconfigurability. The configurability is the ability to modify the structure of a system in order to place it into a variety of configurations [49]. Some researchers highlighted the runtime aspect of configurability by using the term reconfigurability [19].

Behavioral Adaptation: It concerns the change of the behavior of the program. That means adding or removing some functions or changing their QoS. In addition it may be possible to use equivalent functions of services/components to replace existing services/components. The behavior change is also called functional adaptation. In the thesis of An P.K. [60] he use the term "functional adaptation" when the system functionality is modified by the adaptation; otherwise the adaptations are called non-functional adaptations.

As pointed out by Oreizy et al. [76], "Changes can include the addition, removal, or replacement of components and connectors, modifications of the configuration or parameters of components and connectors and alterations in the component/connector network's topology." All adaptation mechanisms (both structure change and behavior change) are based on four basic operations: add, delete, link and unlink. For example to replace a component by another, we need to execute four operations: unlink - delete - add - link. Therefore a system that can dynamically accomplish these four basic operations is a system supporting adaptation. However we will add two operations: saving and loading states, which are indispensable to achieve component or service migration.

As we mentioned above the adaptation system is one solution for ubiquitous computing. Next we will present what are an adaptation system and its roles in ubiquitous computing.

B. What is an adaptation system?

An adaptation system is a system providing application adaptation according to the environment. In an adaptation

system, there is a middleware and an adaptation platform. An adaptation middleware is a middleware providing some mechanisms to achieve adaptation tasks. It is also a layer that manages heterogeneity, communication and context collection. In addition an adaptation platform is also a system providing intelligent analysis, planning heuristics and smart decision services. In an infrastructure point of view, there are two different adaptation systems.

1) Local and distributed adaptation system: Local: a local-based system running on a single computer. Applications running on the system can be distributed, but all the components of the system are deployed on one computer. It looks and acts like a centered adaptation server, which provides context collection service, adaptation plan service and so on. The main disadvantage of such a system is that it cannot adapt itself to the changing of the environment in a distributed network. A centered adaptation server can have a very limited number of clients because of bottlenecks such as network bandwidth, computing resources and storage resources. Such adaptation services are very frequently used operations and they are high resources consumer services.

Moreover, they need powerful computing resources to provide real-time performance. In this condition, because of such bottlenecks, a single server is not suitable to guarantee good performance. Therefore, it is not a good solution for ubiquitous computing.

Distributed: a distributed system supports system level adaptation. The autonomous system must be a distributed one. A distributed system can deploy its components over a network. It is called the adaptation domain on the MUSIC platform [85]. They defined a notion of system core. The core has a minimum functionality that allows components' life-cycle control and components' deployment. The MUSIC platform deploys the core into the adaptation domain on each device. Moreover the system can adapt itself as applications do.

Local or centralized adaptation systems have the same roles in pervasive computing that the next section will present.

2) Role of the adaptation system: In ubiquitous computing adaptation systems have four roles. As mentioned above the four roles are Context Manager, Planner, Decisioner and Middleware (see Figure 2). There are two types of role: collaborator role and core role. Just like in the adaptation loop's steps that we described above. Collaborator role is not an imperative ones. On the contrary, according to the definition of an adaptation system, the Planner role and the Decisioner role are imperative. They are the core roles, which mean they are a part of the system and the system implicitly uses them. The others parts can be replaced by the middleware.

Context Manager: this is a collaborator role. System needs to collect context information and to build high-level context models. Context information may come from different devices or different software systems and even from some social networks. Hence, the system may use a context middleware to achieve the collection and the model construction. However, the system can also deploy the context manager as a component (or service) and distribute it in the adaptation domain.

Planner: the system needs to dynamically produce adaptation plans at runtime. A planning heuristic is used to determine

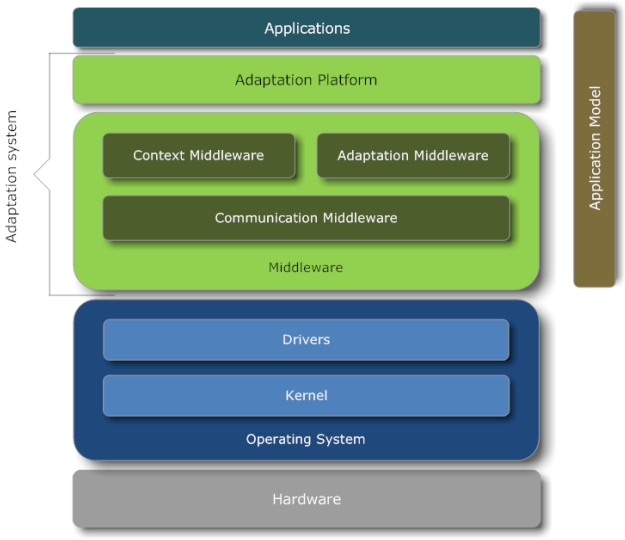

Fig. 5. Architecture of an adaptation system for ubiquitous computing

the best configuration of an application. The reasoning is based on context information.

Decisioner: with the list of adaptation plans produced by the heuristic, the system analyzes the risk of each plan, takes the decision and asks the actor to execute the changes. Role of the Middleware: adaptation system hides the heterogeneity of the network and the software environment; it supports advanced coordination models among distributed entities; and finally it makes the distribution as transparent as possible [53].

Next section we will present technologies that are used to adaptation systems.

III. Technologies of Adaptation Systems

According to the literature, we propose general adaptation system architecture (see Figure 5), which is n layers architecture. From bottom to top, there are the hardware layer, the operating system layer, the adaptation layer and the applications layer. The hardware layer present mobile devices, computers network equipment and so on. On top of hardware layer is operating system layer (OS layer), it presents operating systems like Windows, Linux, iOS [8], and Android [40].

The adaptation platform layer provides intelligent analysis and decision-making whereas the middleware contains a different composition of elements; both of them constitute the adaptation layer. At the top, it is the application layer. These applications will be supervised and manipulated by adaptation system. The Application Model concerns the applications layer and the adaptation layer. It can be different or same model for application and adaptation system. In this chapter we will focus on the adaptation system and application model describing their responsibility, their composition, the technologies applied and the relationship between each other.

According to this general architecture, we propose the taxonomy of adaptation systems' technologies shown in Figure 6. And we will present each technology in this chapter from left to right by the order that we show in the figure. Next presents begin with Communication Middleware.

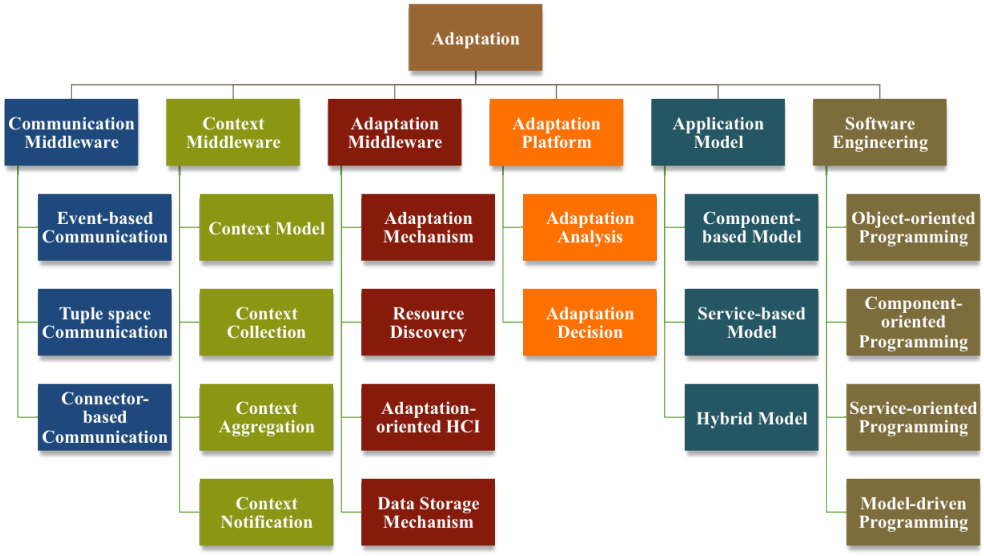

Fig. 6. Overview of adaptation systems' technologies

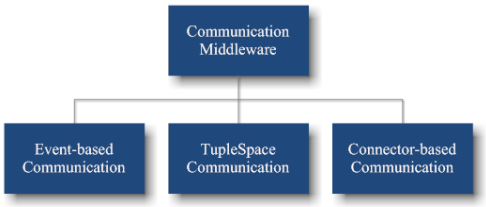

Fig. 7. Communication Middleware technologies

A. *Communication middleware*

Communication middleware is the base of the distributed systems. Adaptive application and upper layer's of the middleware use communication middleware to achieve the intra-application communication and the communication between applications. Communication middleware has already been discussed in the traditional middleware domain. Therefore we will only discuss three popular communication mechanisms: event-based communication, TupleSpace-based communication and connector-based communication. (see Figure 7)

Event-based mechanism allows communication by events. Events can be filtrated, can set up timeout and can be classified by topics. Event-based mechanism decouples the communication from the time-space and the control flow. Event-based mechanism supports Multi-points communications, intermittent networks and Ad hoc networks. Gaia [81] , DREAM [63], Milan [80], [36] and Siena [24] are event-based approaches.

TupleSpace mechanism allows communicating with shared memory. TupleSpace is like a white board. Message producer put messages into the tuple space and consumer reads them from the tuple space. Tasks can generate asynchronous requests. TupleSpace provides a common public data structure to support communication, which is called Tuple. Often it is used as key-value pair. The ubiquitous middleware in TOTA [68] and MESHMdl [51] use TupleSpace mechanism for communication.

Connector-based mechanism allows communication with connectors. On a component-based system, connector act like data containers. Connectors provide a unified I/O interface for components. So application components are not concerned by remote communications. The connector takes care of this part of operation. Connectors can also offer some advanced functions. For example, a user (player 1) uses his cell phone to play a game with another user (player 2) though a Bluetooth connection. Next, because player 2 moves the Bluetooth connection is lost. The connector can use the SRD service (Service Resource Discovery service) to find a computer that can connect to player 2 though Bluetooth and to player 1 through Wi-Fi. Then the connector uses this computer as a proxy to connect the two players. Of course, the whole reconnection operations are transparent for the users. Kalimucho [32] uses such type of communication middleware, which is called Korrontea [22]. It is a distributed connector-based communication middleware.

Moving towards the next row of Communication Middleware, section 3.2 will present these Context Middleware technologies.

B. *Context Middleware*

Integrating physical space and information space into application can make application's complexity and developing

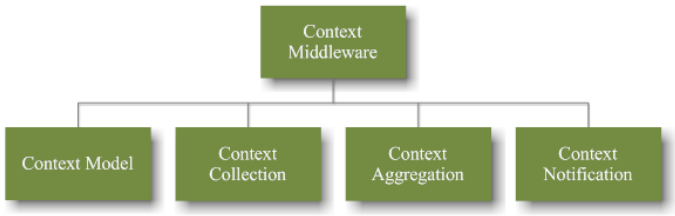

Fig. 8. Context middleware technologies

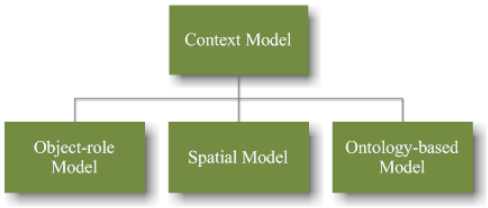

Fig. 9. Context model

difficulty increase. In addition, it compromises code reuse. The context middleware needs to take care of the context collection, aggregation, and analysis operations and to provide a mechanism for context notification to application and/or adaptation platform.

In this section we will present technologies involve in Context Middleware: Context Model, Context Collection, Context Aggregation and Context Notification. (see Figure 8)

1) Context Model: Context model is the format used by the context middleware to provide context information for the upper layer. This model must be structured, consistent, decomposable, assemblable and extensible. In [99], in the point of view of data structure, they divide the context model into 6 categories (Key-value models, Markup Scheme models, Graphical Models, Object Oriented Models, Logic Based Models and Ontology Based Models). In [16], they describe 3 types of current approaches of context modeling: Object-role, Spatial, Ontology-based and hybrid. These three types of context model are the most popular in nowadays ubiquitous computing. (see Figure 9)

Object-role model: ”Common to object oriented context modeling approaches is the intention to employ the main benefits of any object oriented approach - namely encapsulation and reusability - to cover parts of the problems arising from the dynamics of the context in ubiquitous environments” [99]. It makes the dynamics of the early context modeling approaches, such as key-value pairs, inadequate. Such approaches are the cues [94] (It is developed within TEA project.), Active Object Model [30] and CML (Context Modelling Language) [46] etc. Combined with graphical models like ORM (Object-Role Model), ORM supports both analysis and design of the context requirements and relational representations and grammars for high-level context abstractions. It provides more comprehensive support for capturing and evaluating imperfect and historical information than many of other context modeling approaches [16].

Spatial model: In real world, people often ask these questions on a mobile phone or use their smart phone to figure out answers of these questions: ”Where are you?” ”Where can we meet?” ”Where am I?” etc. In ubiquitous computing location and nearby resources are very important. Thus, space and location is a vital factor for context. Schilit, Adams and Want define context as ”Where you are, who you are with and what resources are nearby”[93]. The spatial model concerns location information. There are two kinds of coordinate systems to present location information: Geometric coordinates and Symbolic coordinates. Geometric coordinates represents points or areas in a metric space, such does the GPS system. Usually the GPS sensor is the main equipment in this aim. Symbolic coordinates represent the location by an identifier, such as the location of a sensor, the location of a wireless hotspot, a room number, a building id etc. A camera, a cell phone or something else can be the collecting equipment. Augmented Word Model is developed in the Nexus project [73] this model that both have a geometric coordinate system and a symbolic location system. Spatial model needs modeling both location and relationships of objects. The location information needs to include a Point (Point of object) and a Range. That allows locating other objects in the range that is called Nearest Neighbor [16]. As mentioned before, the spatial models are well suited for ubiquitous computing. The weakness of these kinds of models is that they only concern the location emphases. Thus, researchers have mixed this model with others to build hybrid models, such as Spatial + Object-based model [73] and Spatial + Ontology-based model [38].

Ontology based model: it offers good ability and facilitates the realization of reasoning; recently, it has been extensively used. The concept of Ontology comes from the philosophy. It refers to the explanation of an objective existent system. Studer et al. defined Ontology, as: ”An ontology is a formal, explicit specification of a shared conceptualization” [100]. Ontology could be represented by several methods, such as natural or logical language. In the web semantic domain, there are mainly two description languages to represent ontologies: RDF (Resource Description Framework) and OWL (Ontology Web Language). Nowadays Ontology-based context middleware often use OWL for the context model, such do COSMOS [23], CoBrA [27], SOCAM [44]. CoBrA while SOCAM middleware adopts respectively the SOUPA [28] and CONON [122] ontologies. Gaia [81] middleware uses the DAML+OIL ontology language (a predecessor of OWL) to construct the context model. This model is used to represent high-level context information for context reasoning. The context collector collects raw data from sensors from which complex context data will be constructed by the context aggregator with OWL, then provided to the context analyzer.

Before discussing hybrid model we present a comparison table of the three models we discussed above. Partial satisfaction of the requirements is shown as ’∼’ in the table. Full satisfaction is shown as ’+’, and none satisfaction is ’–’.

Hybrid model: Why do we need a hybrid model? The most common answer to this question is when existing solutions are not satisfactory. Thus, mixing these unperfected solutions allows finding a better representation. These context models have different emphases and weaknesses as shown in Table I. There are some ways to mix these models, such as: Henricksen

	Object-role	Spatial	Ontological
Heterogeneity	+	∼	+
Mobility	∼	+	−
Relationships	∼	∼	+
Timeliness	+	+	−
Imperfection	∼	∼	−
Reasoning	∼	−	+
Usability	+	∼	∼
Efficiency	∼	+	−

TABLE I
COMPARISON OF CONTEXT MODELING APPROACHES WITH CONTEXT REQUIREMENTS [16]

	Context Model			Context Aggregation
	Object-role	Spatial models	Ontology based models	
Active Object Model	X			
CARE	X	X		
CoBrA			X	
Context Toolkit	X			X
COSMOS	X		X	X
A. Frank		X	X	
GAIA	X			X
Nearest Neighbor	X			
SOCAM			X	X

TABLE II
COMPARISON OF CONTEXT MIDDLEWARES

et al. [50], CARE [1] framework, Spatial + Object-based model [73] and Spatial + Ontology-based model [38] and COSMOS [23].

2) Context Collection: The goal of context collection is in any equipment, any OS to effectively collect the information about resource's running state, work performance and so on. Furthermore, for user-centric context collection we also need to collect user geographical location, surrounding resources and equipment and time through a variety of ways. Context collection provides a software abstraction layer of sensors to the upper layer or directly to the application, such as Context Toolkit [88] does. Context Toolkit abstracts sensors as Widgets in the middleware; all Widgets provide a unified interface. Similar approaches are SOCAM's Context Provider [43], CoBrA's Context Acquisition Components [26] and EEM's Context Representer [64].

Dealing with actual context definition, context collection also includes the collection of information about how users utilize their environment, such as information on social networks: e.g., what kind of color they like, what are their hobbies, and their favorite food. In [120] and [91] a technology to resolve these problems is presented.

The role of context collection in adaptation system is to provide meta-context information for every parts of the system.

3) Context Aggregation: The goal of context aggregation is to filter, analyze and build the high-level context information. After that it will translate this low-level context information (called raw data) into the context model as high-level context information. Such high-level context information will help to simplify the task of context analysis and reasoning in the adaptation middleware. Ontology based context middlewares transform raw data to a unified OWL described context model. To eliminate the inaccuracy and conflicts of raw data, the context aggregator will have many interactions with the context reasoner. The context reasoner, in the general architecture that we mentioned before, is located in the adaptation platform layer. However, in a system without adaptation platform, the context reasoner is usually included in the independent context-aware middleware.

4) Context Notification: The ways to notify the upper layer about context changes are generally divided into active and passive notification. In the active notification when context change the upper layer is notified immediately, generally by using an event-based notification mechanism. In the passive notification when the upper layer needs to check the context changes it calls the context API. In that aim it requests a database or read a context log file or uses a "white board" communication mechanism such as TupleSpace which provides a logical shared memory with a good performance.

Context-driven systems generally use the event-based notification. The event-based mechanism provides good extensibility, scalability and low coupling. For example Context Toolkit and Gaia [81] use this mechanism.

In a distributed adaptation system using event-based notification the context middleware is concerned by what messages must be notified and what must not be; and also to whom these messages have to be sent. Of course, different types of application have different emphasis; this implies that message selection and message destination will be different. When using event-based notification Context middleware will provide some pre-analyze mechanism to suit to these different emphasis of applications. When using white board mechanisms, the communication middleware needs to be able to decide what messages will be deleted after a certain delay and what messages must be persistent.

In next section, moving forward to Adaptation middleware. It has four technique categories: Adaptation mechanism, Resource discovery, Adaptation-oriented Human-Computer Interface, and Data storage mechanism.

C. Adaptation middleware

Adaptation middlewares should provide some basic adaptive-oriented services and adaptation mechanisms. These basic adaptive-oriented services must be able to be highly configurable for QoS and dynamically removable and addable. In addition, the management of these services is the under the responsibility of the adaptation platform and the application needs not to be concerned by that. Adaptation mechanisms are core services that the adaptation middleware must provide. Adaptation mechanisms directly relate to the application model and the development method. For example, if the adaptation middleware provides a micro-kernel mechanism for adaptation, the application model will be a component-based or a hybrid model. If the mechanism is AOP based (Aspect-oriented programming) the development method must be AOP too.

Figure 10 presents these adaptation mechanisms and adaptive-oriented services. We will describe them in detail next. We begin with Adaptation Mechanism.

1) Adaptation mechanism: This part of technology is relatively mature. The adaptation platform supervises their application relies on reflection technologies. Reflection and

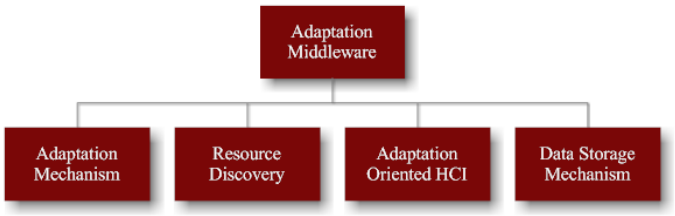

Fig. 10. Adaptation middleware technologies

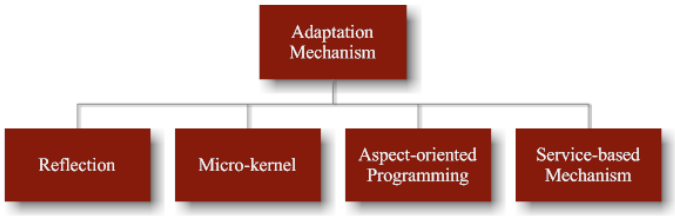

Fig. 11. Adaptation mechanism

dynamic module loading technologies are the most popular mechanisms for adaptation systems. As defined by Bobrow et al. [20] Reflection implies the ability of a software system to observe and reason about its own state and to modify its own execution. There are two categories of Dynamic module loading technologies: Micro-kernel and AOP. (see Figure 11)

Micro-kernel technology uses the experience of micro-kernels OS. In [18] is presented the concept of "Operating systems as application programs". Only the basic functions are loaded into the kernel, other functions will be loaded at runtime as independent modules. Dealing with this concept there are several micro-kernel dynamic module-loading platforms, such as OSGi [3]. For ubiquitous computing, several platforms are based on this technology, such as MUSIC [85] platform, BASE [15] and MundoCore [2].

AOP [61] technology is a software engineering approach that enables separation of crosscutting, such as QoS, security, fault tolerance, and logging. AOP development obliges developers to predefine join-points, cut-points and advices. A join-point is a point in a program where additional behavior can be joined. A Cut-point is a description of a join point. An Advice is some additional behavior, a piece of code that can run before, after, and around join-points. Advices are weaved into the program when a join-point is matched with a cut-point. The weaving is the mechanism for integrating advices. It can happen at either compile or run time. WComp [108] is a typical AOP-based adaptive middleware; it used a concept specific named Aspect of Assembly for adaptation.

On the other hand, Service-based technology is a popular technology to realize adaptation middlewares. Service-oriented systems provide flexibility in handling and dynamicity and suitability for the integration of new devices. These systems always support cross-languages, cross-software-platforms and cross-hardware abilities. A service description only concerns what this service uses, never how this service realized. These abilities exactly suit Ubiquitous-computing requirements, such as heterogeneity, scalability, security, mobility, and discovery [108]. Service-based technology is used for adaptation middlewares with several different implementations, such as Web services, REST (Representational State Transfer) or CORBA services and customized services. CAPUCCINO [83], VieDAME [72], SAMProc [96] and WComp are Web-service based middlewares. DoAmI uses CORBA services only. Gaia [81] , Aura [98] uses customized services.

	Micro Kernel	AOP	Service-based
Aura			X
AxSel			X
BASE	X		
CAPUCCINO		X	X
DoAmI			X
Gaia	X		
Kalimucho	X		
Mundo Core	X		
MUSIC	X	X	X
RCSM	X		
SAMProc			X
VieDAME			X
WComp		X	X

TABLE III
COMPARISON OF ADAPTATION MIDDLEWARE WITH ADAPTATION MECHANISMS

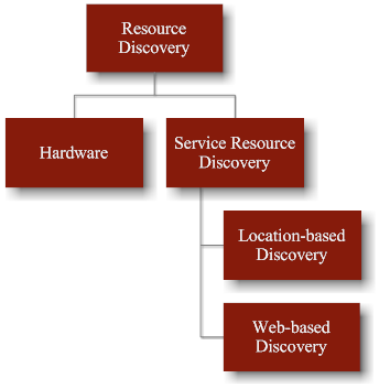

Fig. 12. Resource Discovery

However, there are several hybrid solutions such as SCA [75]. CAPUCCINO is based on SCA, and WComp is based on a SCA-like architecture, which is named SLCA [108] and mixes SCA and AOP.

2) Resource Discovery: An adaptive oriented application is a location-based and distributed application. A lot of code will be distributed on remote devices during the run time of such applications. In addition, location-based application means that the user location's changing will trigger the changing of the application context and it will change the environment resources. Therefore the resource discovery becomes a necessary service. The resource discovery includes: Hardware resource discovery and Service resource discovery. (see Figure 12)

Hardware resource discovery services (HRD services) are mainly use by Wi-Fi or Bluetooth (SDP Service Discovery Protocol) to discover the actual environment hardware resources.

Service resource discovery services (SRD service) use the surrounding hardware and the Internet to discover the services.

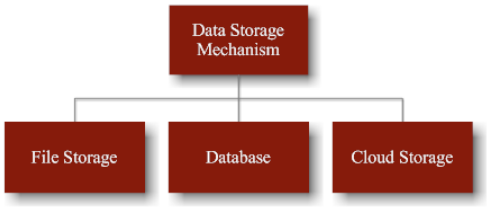

Fig. 13. Data storage

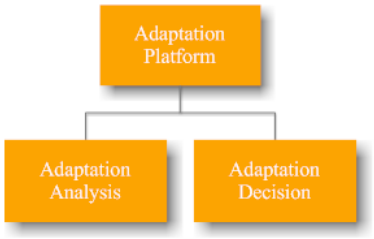

Fig. 14. Adaptation Platform

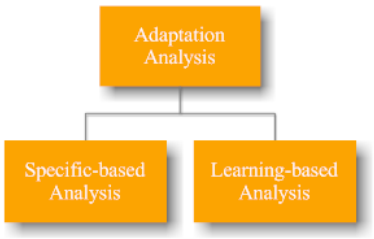

Fig. 15. Adaptation Analysis

When a user changes its geographical position, the SRD service will discover the services that are provided by the surrounding hardware or use the surrounding hardware to connect the Internet to achieve this discovery. Therefore the SRD service needs a users' geographical location service to find the user's position. This service can use (or combine) multiple technologies (like Wi-Fi, GPS, 3G Network, etc.) in order to find the user's position. It's called location-based discovery. On the other hand, SRD services exploit the Internet to discover Web services, which is called web-based discovery.

3) Adaptive oriented Human Computer Interface (HCI): Users' moving will bring an uncertainty of context during application runtime. Therefore the platform needs to provide a HCI service that will decouple the application's business logic code and the HCI code. In order to better support migration of application between different devices, the HCI service needs to use cross OS and cross device HCI technologies. The most commonly used HMI technology is HTML.

4) Data storage mechanism: Applications need to store and share their data and to search into stored data. In a distributed environment, how to guarantee that applications can get fast access to data at any time and everywhere? Generally there are tree possible ways: File system, Database and Cloud storage. File system is used in Gaia's CFS [81]. Databases are the best solution for different purposes because they include access and privacy mechanisms. On the contrary Cloud storage needs Internet access and data privacy is always a problem. However, Cloud storage can suit to self-adaptive platforms. (see Figure 13)

In this section we discussed the four technique categories of Adaptation middleware, next we will present Adaptation platform and its techniques.

D. Adaptation Platform

This is the core of an adaptation system. It can support several types of autonomy, from fully automatic self-contained to human-in-the-loop [76]. It can be distributed or centralized. It is responsible for analyzing and taking adaptation decisions. It provides intelligent analysis, planning heuristics and smart decision services. According to its responsibilities and its definition, we summarized the technologies of adaptation platforms into 2 categories, which are Adaptation analysis and Adaptation decision. (see Figure 14)

We will introduce techniques are used in Adaptation analysis in section 3.4.1 and in section 3.4.2 we will present Adaptation decision techniques.

1) Adaptation Analysis (Situation reasoning): What is the current situation? It is the basic information we need to know and that will guide us to take a decision. Before taking a decision, the human brain will analyze the environment, objects, etc. It is called problem analysis, which is the process to identify the situation. It helps our brain to take a decision [59]. The adaptation platform simulates this process to help the system to take adaptation decisions. Thus, situation reasoning is a very infancy research area. Researchers defined a situation as an external semantic interpretation of sensor data in context-aware applications [33]. To deal with this definition, there are two kinds of categories of techniques: Specification-based and Learning-based techniques (see Figure 15), which allow constructing situation models that can be used toby the adaptation system (see Figure 16).

Specification-based techniques In early times, the relationships between raw data context information and situations were easy to establish because there were a few sensors. Specification-based approaches are mainly situation identification techniques. There are several logic rules based approaches, such as formal logic model [65] and spatial-temporal logic [31], which support efficient reasoning. They have been widely applied, thanks to their powerful representation ontologies and reasoning capabilities. Recently Ontology reasoning is extensively used, as we mention before. Formal logic coordinates with ontology context models to provide a standard vocabulary of concepts and semantic relationships to infer situation. To model and identify the real world the uncertainty must be considered. To deal with this uncertainty, traditional logic-based techniques need to be incorporated with other probabilistic techniques [45]. Fuzzy logic and Dempster-Shafer theory have been applied to solve this problem [5], [52], [69].

Learning-based techniques There are four branches for learning-based techniques: Bayesian derivative models, Grammar-based, Information entropy, and Pattern Mining.

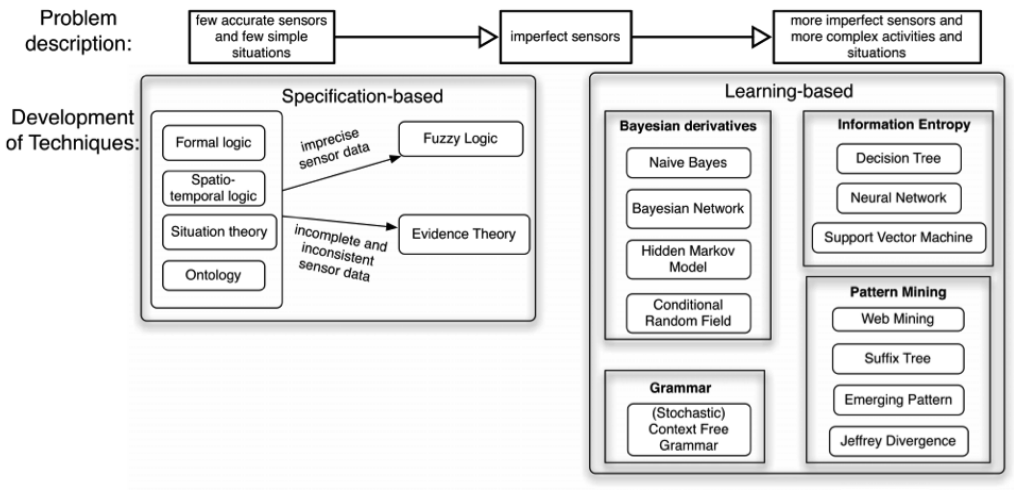

Fig. 16. Development of the main situation identification techniques corresponding to the increasing complexity of problem descriptions [120]

Bayesian derivative models are very famous and include two kinds of approaches. One is an encoding causal relationships model, such as Nave Bayes [78], [104] and Bayesian networks [42]; the other is the an encoding temporal relationships model, such as Dynamic Bayesian network [55], Hidden Markov Models [116], [48] and Conditional Random Fields [110], [56]. Grammar-based approaches are applied for representing the complex structural semantics of processes into hierarchical situations [71], [87], [109]. Decision trees [14], Neural Networks [119] and Support Vector Machines [77], [54] are built on information entropy. Whatever actual Learning-based techniques need a lot of training data to set up a model and estimate its parameters [107].

2) Adaptation decision: Adaptation decision is generally divided into static approach, dynamic approach and Artificial Intelligence approach. The static approach needs to predefine and implement the adaptation logics. Different kinds of application have different particular emphasis; therefore these self-adaptation logics are widely divergent. The adaptive middleware cannot be able to predefine all self-adaptation logics for all applications. Due to the kind of middleware, using a static way to generate adaptation logic often implies that application manages its own self-adaptation logic. That means that the application developer needs to create self-adaptation logic for each specific application. The middleware just provides the context-awareness and the adaptation mechanisms. It runs as a framework. According to our readings, we found six different approaches which are: 1) Algorithm-based adaptation logic, 2) Policy-based adaptation, 3) Planning-based adaptation logic, 4) Automata-based adaptation logic, 5) Graph-based adaptation logic, and 6) Artificial Intelligence. (see Figure 17)

Algorithm-based is a logic that is realized by a chain of algorithms. After computing, the algorithm will provide a solution of decision or an adaptation plan in a given time. The algorithm can be dynamically changed during runtime; Dynaco framework provides this dynamic adaptability. Andr et al. proposed an algorithm to dynamically make master-slave adaptation based on the Dynaco framework. This algorithm is based on the description of the behavior of patterns, which

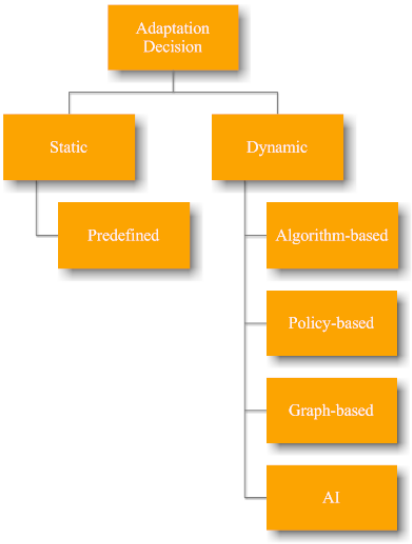

Fig. 17. Adaptation decision

depends on the state of the pattern and the distributed system and on a QoS objective [7].

Policy-based adaptation logic provides guidance in decisions and actions. The policy management services normally consists of a policy repository, a set of Policy Decision Points (PDP) for interpreting the policies, and a set of Policy Enforcement Points (PEP) for applying the policies [114]. Kephart and Walsh discussed different policy types that can be exploited in adaptation systems, such as goal policy, action policy (in the ECA* form) and utility policy [58]. Due to the dynamic changes of situation in pervasive computing, the policy also needs to dynamically change. Hence, Lutfyyia et al. have proposed a control-theoretic technique to deal with this requirement for adaptation systems. Shi et al. proposed an ECA-based XML description policy management technique for their UbiStar adaptation system, which supports runtime changes and online updates [97]. There are a number of adaptation systems that adopt policy-based management for

adaptation planning and decision, for example, [57], [12], [89].

Planning-based In 1995, Russell and Norvig pointed out that a planning-based adaptation engine needs continuous planning via contingency planning or re-planning [86]. Planning has also two aspects in adaptation systems: Observation planning and Adaptation planning [76]. Observation planner decides when and where to adapt depending on observed information (high-level context information). Adaptation planner determines which adaptation plan needs to be made and when it must be executed depending on the identified situation. "It refers to the reconfiguration capability of an application to changing operating conditions by exploiting knowledge about its composition and Quality of Service (QoS) metadata associated to its constituting services". MADAM [39] is a typical planning-based middleware [35]. MUSIC project adopts this middleware and extends it.

Graph-based technologies is the use of mathematic graph to resolve problems such as component deployment dependence, which can help to decide what nodes can be loaded in one device depending on available resources. For example, AxSeL [47] is a graph-based adaptation middleware. AxSeL's application is represented as a global, bi-dimensional and flexible graph of dependencies where nodes represent services and components. AxSeL uses a graph coloring of dependencies algorithm with contextual information on nodes of the graph to proceed for a deployment decision under certain constraints.

A.I. -based is a learning-based technology. It may be useful in an adaptation deciding process thanks to its rich possibilities in planning, reasoning and learning. However, actual AI-based systems encounter some common obstacles such as quality assurance. Nevertheless, some approaches are already used into adaptation systems, such as A*-based algorithm, heuristics, Nave Bayes, Bayesian networks and so on. For example, CADeComp [11] uses an A* based algorithm to reduce complexity of the components' placement problem. This A* based algorithm allows approaching the optimal solution. Arshad et al. use AI-planning for self-healing system [9]. Maes proposed goal-based model for action selection [67]. Tesauro et al. [106], Weyns et al. [115] work on Intelligent-Agent based Multi-Agent Systems. Some adaptation systems also adopt some techniques from Machine Learning and Soft Computing, such as Genetic algorithms and different online learning algorithms. Several researchers highlight reinforcement Learning (RL). Amoui et al. considers that RL can be a promising option for dynamic action selection [4]. Dowling believes it can be use to decentralize collaborative self-adaptation software [34]. Tesauro thinks that RL has the potential to achieve better performance in comparison which traditional methods while requiring less domain knowledge [105]. There are several methods adopted by adaptation systems to help decision and adaptation planning, for example, decision theory, utility theory, and Markov Decision Process (MDP) and Bayesian network.

3) Summary: Adaptation decision is the most important part of the system. Adaptation decision deals with the adaptation plan problem, the observation-planning problem, the components placement problem and the making adaptation decision problem. It needs adaptation analysis to identify the actual situation and decide adaptation based on the situation. (see Table IV)

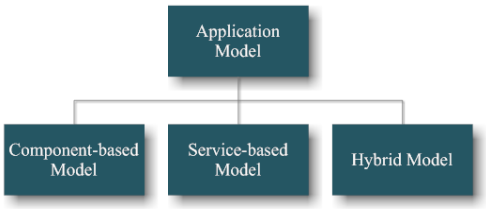

Fig. 18. Application models of adaptation

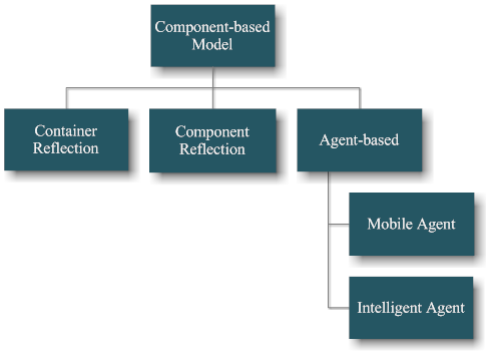

Fig. 19. Component-based application model

E. Application Model

There are three categories of application models which are Component-based, Service-based and Hybrid (see Figure 18). Section 3.5.1 will present Component-based application model, Service-based model will be introduced in section 3.5.2, and in section 3.5.3 we will discus Hybrid model. We will summary in the end of this section.

1) Component-based: Szyperski et al. [103] defined the component concept as "a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties". A component can be composed with other components by well-defined interfaces that specify what it requires and provides. A software configuration is a composition of the software components. In pervasive computing, to adapt component-based software means to change the software configuration. This application model provides several features such as binary reusability, dynamic configurability and dynamic deployment. Thus it is a popular application model for pervasive software. Depending on the reflection technique, there are two kinds of component-based model (see Figure 19):

Container-based reflection: Components are included in containers, and containers supervise them. The Component encapsulates the business logic and the container provides non-functional proprieties. This model can provide a loose coupling between business logic and non-functional proprieties as, for example, QoS control and network control. Kalimoucho's Osagaia model [21] is a typical Container-based model, see figure 20.

	Static	Algorithm-based	Policy-based	Planning-based	Graph-based	A.I.-based
AxSel					X	
CADeComp						X
CAPUCCINO			X			
Dynaco framework		X				
GAIA			X			
Kalimoucho	X		X			
MUSIC				X		
UbiStar			X			
WComp	X		X			

TABLE IV
SUMMARY OF ADAPTATION DECISION

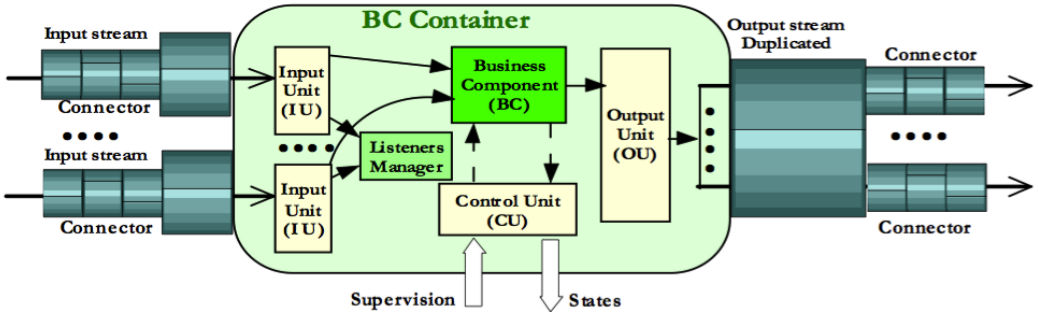

Fig. 20. Osagaia Component-based model

Component-based reflection: System provides a unique container for all the components of system. This container supervises the components thought component reflection technique. Application is constituted by a composition of components. Non-functional properties cannot be managed globally by the system. EJB [101], and .Net [70] are typical approaches of component-based reflection platforms.

Agent-based model: It's a component model based on the agent theory model. Russell and Norving defined an agent as: "anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors". According to this definition, an Intelligent Agent (IA) is an agent that has an AI behavior, and that can autonomously interact with its environment. A mobile agent is an agent that has the ability to transport itself from one system to another through a network [62]. These agent-based models are often used to construct Multi-Agent Systems (MAS) [117]. They encounter one common obstacle: quality assurance. There are several ubiquitous-oriented agent-based projects such as MDAgent [121], SpatialAgent [92], UbiMAS [13] and G-net [118].

2) Service-based: Service in IT area means an entity (e.g. a component, an application) providing some functionalities that is used by other entities of the system or other systems and, maybe, by human beings. Service-based model emphasizes interface-oriented programming. It provides a loose coupling between service interfaces and service implementations. There are several ubiquitous projects that used custom service model, such as SOCAM, Gaia, and Aura [98]. Web-services are actual popular services models. It is a W3C standard that uses WSDL as a description language and SOAP (Simple Object Access Protocol) to communicate. This kind of models allows dynamically changing services' implementation during runtime.

Researchers proposed several autonomic service-oriented processes. In 2005 Verma and Sheth [111] proposed Autonomic Web Processes (AWP), which are web-service-based processes. SAMProc is a middleware for self-adaptive mobile processes [96]. This middleware allows the application changing of location and behavior during its lifecycle. They use a BPEL-like description language to describe their SAM-WS web-service model.

3) Hybrid: Hybrid approach uses component-based model to provide services, it's a good way to construct adaptation systems compatible with all types of application. It that means the system is based on a service model and the implementation of service uses component-based models. Component-based models can provide fine adaptation granularity and service-based models can provide good heterogeneity and scalability. WComp's SLCA model [108] and Plastic model [10] are such service-centric approaches.

4) Summary: Between platform and application level, we can use different application models. For example the MUSIC platform is constructed by an OSGi component-based model and provides SOA services for applications. So applications can be constructed on any model. (see Table V)

Next section will introduce the last raw of taxonomy, Software Engineering. It will present different technique that we use today to develop adaptation system.

F. Software Engineering

Adaptation systems can be implemented by different software engineering techniques. Object-Oriented Programming (OOP) is the most popular basic technique for software

	Component-based	Service-based	Agent-based	Hybrid
Aura		X		
AxSel	X			
CAPUCCINO	X	X		X
DoAmI		X		
G-net			X	
GAIA	X			
Kalimoucho	X			
MAS			X	
MDAgent			X	
MUSIC	X	X		X
Plastic model				X
SAMProc		X		
SOCAM		X		
SpatialAgent			X	
UbiMAS			X	
VieDAME		X		
WComp	X	X		X

TABLE V
APPLICATION MODELS

	Supervised	Centric	Distributed
Aura	X	X	
AxSel	X	X	
CADeComp	X		X
CAPUCCINO	X		X
DoAmI		X	
GAIA	X	X	
Kalimoucho	X		X
MUSIC	X		X
SAMProc		X	
SOCAM	X	X	
VieDAME		X	
WComp	X	X	

TABLE VI
ADAPTATION SYSTEMS CLASSIFICATION I

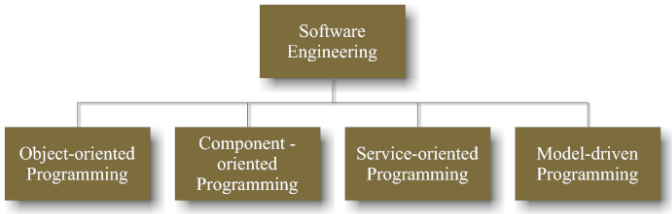

Fig. 21. Software Engineering

engineering; GoF design pattern is a general methodology for OOP. Component-oriented programming and service-oriented programming are actual popular programming techniques; they provide several advanced features for software reusability, scalability, and extensibility. Model-Driven Engineering (MDE) refers to the systematic use of models as the primary artifact for the engineering of systems [37]. (see Figure 21)

1) OOP (Object-Oriented Programming): The birth of Object-Oriented Programming can be traced to 1960s. It is a basic programming paradigm. OOP coordinates with design patterns that can constitute flexible and reusable solutions for developing adaptation systems. Design patterns are a general software engineering methodology for representing well-known solutions to common and recurring problems in software design [95].

2) Component-oriented programming: Component-oriented programming (COP) is also called Component-based software engineering (CBSE). COP can be coordinated with component-based model design - or its variants - for implementing adaptation systems. It is a well-known programming paradigm that can be exploited for developing flexible and extensible software systems [103]. Component-based system can adapt itself by changing components composition. As we mentioned before, this technique can be composed with SOA to provide good flexibility and heterogeneity. OSGi is the most popular component-based dynamic framework for component-oriented software.

3) AOP (Aspect Oriented Programming): In ubiquitous computing, AOP technique can be used to encapsulate the adaptation described by the aspects. During runtime, aspects can be dynamically waved into the application code in order to achieve adaptation. AOP allows implementing fine-grained adaptation actions at a level lower than components [41], [90]. AspectJ is the most popular framework for Java programming. JAC is a dynamic AOP framework, which uses wrapping to dynamically modify or add join-points [82].

4) SOP (Service Oriented Programming): SOP coordinates with service-based model design or its variants for implementing adaptation system. ”Much of this technology focuses on the concept of discovery by location: being able to discover and interact with the objects around you” [17]. SOP can be implemented by Java with Jini [102] framework, Microsoft .Net framework [70], Web service and so on. Web service technology provides flexibility for composition, orchestration, and choreography [79].

5) MDD (Model Driven Development): Schmidt defined Model-Driven Engineering, as ”A promising approach to address platform complexityand the inability of third-generation languages to alleviate this complexity and express domain concepts effectivelyis to develop Model-Driven Engineering (MDE) technologies”. MDE provides a high-level abstraction of system that can be directly migrated to cross-platforms. Model-Driven Architecture (MDA) is an MDE approach proposed by the Object Management Group (OMG) [74]. The abstracted model is platform independent; it is call PIM (Platform-independent Model). PIM can be translated to one or more platform-specific models (PSMs). PSM is specified for one specific platform.

In the last section of this chapter, we will summary the different related approaches.

6) Classification of related approaches: We described above actual technologies in Ubiquitous computing, in this section we will present several famous related approaches in Ubiquitous computing with three different classifications.

According to the classification we mentioned before in section 2.1.3 and section 2.2, we draw the adaptation system classification table, shown as Table VI. Self-self adaptation is out of this article; we just classified the supervised adaptation. But some approaches neither supervised nor the self-self like SAMProc. It’s because of these approaches are not a complete adaptation system, i.e. they are just adaptation middleware.

Adaptation systems can be classified according to adaptation

	Architecture-based	Parametric-based	Aspect-oriented-based
Aura	X		
AxSel	X		
CADeComp	X		
CAPUCCINO	X	X	X
DoAmI		X	
GAIA	X	X	
Kalimoucho	X		
MUSIC	X	X	X
SAMProc		X	
SOCAM		X	
VieDAME		X	
WComp	X	X	X

TABLE VII
ADAPTATION SYSTEMS CLASSIFICATION II

	Context middleware	Adaptation middleware	Adaptation platform
Aura	X	X	
AxSel		X	X
CoBrA	X		
COSMOS	X		
CADeComp	X	X	X
CAPUCCINO	X	X	X
DoAmI		X	
GAIA		X	X
Kalimoucho	X	X	X
MUSIC	X	X	X
SAMProc		X	
SOCAM	X	X	
VieDAME			X
WComp		X	X

TABLE VIII
ADAPTATION SYSTEMS CLASSIFICATION III

as: Architecture-based adaptation, Parametric-based adaptation and Aspect-oriented-based adaptation. (see Table VII) "(m) Any types of adaptation techniques have been developed: architecture-based adaptation that is mainly concerned with structural changes at the level of software components, parametric-based adaptation that leverages policy files or input parameters to configure the software components, aspect-oriented-based adaptation that changes the source code of a running system via dynamic source-code weaving, and so on" [29].

Adaptation system can also be classified according to middleware as: Context middleware, Adaptation middleware, and Adaptation platform. (see Table VIII)We propose this classification that because these actual projects are usually called middleware for ubiquitous computing. But they have so many different. Context middleware is the middleware we described in section 3.2, Adaptation middleware is described in section 3.3 and Adaptation platform that we presented in section 3.4.

IV. CONCLUSION AND PERSPECTIVES

We described technologies that are involved in adaptation systems and tried to classify and summarize recent adaptation systems and projects. Engineers design adaptation policies and adaptation rules at system design time. Most adaptation systems provide adaptation support service, but they can't adapt themselves. Front to these challenges, how to integrate existing technologies to create a flexible, scalable and extensible platform architecture that will satisfy ubiquitous adaptation system's requirements? This is obviously an interesting research direction. We presented a lot of technologies but it's not possible to mix all these ones to build an ubiquitous adaptation system because several techniques need powerful computation support. And we cannot use such techniques into embedded systems.

Adaptation doesn't only mean dynamic loading/replacement of software components, I also means satisfying the requirements of an application adaptation. The system needs to dynamically provide some common services, such as encryption/decryption service or user-centric security strategy services. The system itself and common services will be dynamically distributed via networks and adaptable (see Figure 22).

When an adaptation system is based on a micro-kernel technique, the micro-kernel provides the minimum runtime services and is deployed on each device. Adaptation service includes adaptation core and adaptation collaborators, as mentioned in section II. The relationship between core and collaborators looks like the relationship between human brain and human body. Adaptation core corresponds to the previously mentioned adaptive platform that provides intelligent analysis and smart decision. Adaptation core can be a distributed network service or can be a single local service running on a computer. Adaptation collectors mainly correspond to the previously mentioned context middleware and the various context collection programs running on different devices. Collection programs collect raw-data (low-level context information). Context middleware combined it into a high-level context model that will help the adaptation core to analyze/identify the situation and makes an adaptation decision. Adaptation collectors are distributed via network and run on different devices. Adaptation actors mainly correspond to the previously mentioned adaptation mechanism that is included in the adaptation middleware. The adaptation system micro-kernel directly acts to manipulate application's component or service. It also needs to provide negotiation mechanisms in order to find the adaptation plan realization's balance point between adaptation core and application during adapting the planning process. Adaptation actors can be local or fully distributed depending on what adaptation is required. Communication middleware supports all communications though all the levels of the adaptation system providing a unified communication interface.

Pervasive computing involves many technologies. It has been proposed 20 years ago, however, it is still full of challenges. These first 20 years are just only the beginning of pervasive computing. We need a long time to achieve the pervasive computing that was proposed by Mark Weiser [112]. This is a long-term research area.

Adaptation system is a basic part of pervasive computing. This paper tried to find common definitions of middleware and adaptation platform in adaptation systems and proposed taxonomy of adaptation-oriented technologies. We introduced and analyzed these technologies mainly through communication, context, middleware, adaptation platforms, application models and software engineering point of view. Furthermore we make a comparison and summary of some existing pervasive computing projects. Finally we sketched out the trend of adaptation systems and our future research priorities and directions.

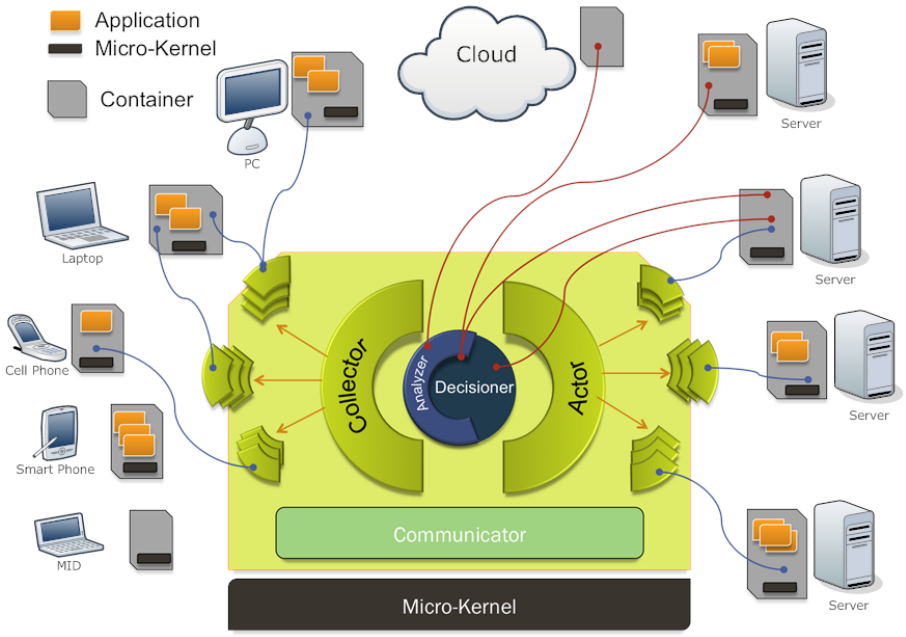

Fig. 22. Dynamic adaptation distributed system

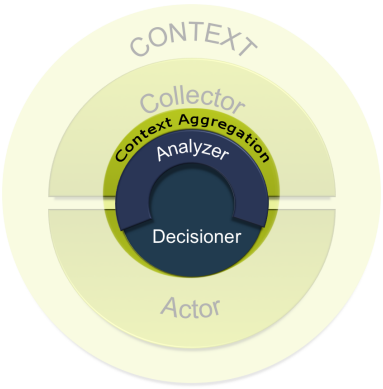

Fig. 23. Our future researches

As figure 23 point out, our future researches are the adaptation platform and context aggregation. We have already the communication middleware: Korrontea, adaptation middleware: Kalimucho and a part of context middleware are proposed in [66].

REFERENCES

[1] Agostini, A., Bettini, C., Riboni, D.: Hybrid reasoning in the care middleware for context awareness. Int. J. Web Eng. Technol. 5, 3–23 (May 2009)

[2] Aitenbichler, E., Kangasharju, J., Mühlhäuser, M.: MundoCore: a lightweight infrastructure for pervasive computing. Pervasive and Mobile Computing 3(4), 332 – 361 (2007), middleware for Pervasive Computing

[3] Alliance, O.: OSGi Technology. OSGi Alliance (2011), http://www.osgi.org/About/Technology

[4] Amoui, M., Salehie, M., Mirarab, S., Tahvildari, L.: Adaptive action selection in autonomic software using reinforcement learning. In: Proceedings of the Fourth International Conference on Autonomic and Autonomous Systems. p. 175–181. IEEE Computer Society, Washington, DC, USA (2008)

[5] Anagnostopoulos, C., Ntarladimas, Y., Hadjiefthymiades, S.: Situational computing: An innovative architecture with imprecise reasoning. Journal of Systems and Software 80(12), 1993 – 2014 (2007), selected papers from the International Conference on Pervasive Services (ICPS 2006)

[6] Anastasopoulos, M., Klus, H., Koch, J., Niebuhr, D., Werkman, E.: DoAmI- a middleware platform facilitating (Re-)configuration in ubiquitous systems. Irvine (2006)

[7] Andre, F., Gauvrit, G., Perez, C.: Dynamic adaptation of the Master-Worker paradigm. In: Proceedings of the 2009 Ninth IEEE International Conference on Computer and Information Technology - Volume 02. p. 185–190. CIT '09, IEEE Computer Society, Washington, DC, USA (2009)

[8] Apple: About iOS Application Design. Apple (2011), http://developer.apple.com/

[9] Arshad, N., Heimbigner, D., Wolf, A.L.: Deployment and dynamic reconfiguration planning for distributed software systems. Software Quality Control 15, 265–281 (Sep 2007)

[10] Autili, M., Cortellessa, V., Marco, A.D., Inverardi, P.: A conceptual model for adaptable context-aware services. Web Services-Modeling and Testing (WS-MaTe 2006) (June 2006)

[11] Ayed, D., Taconet, C., Bernard, G., Berbers, Y.: CADeComp: context-aware deployment of component-based applications. Journal of Network and Computer Applications 31(3), 224 – 257 (2008)

[12] Badr, N., Reilly, D., Taleb-Bendiab, A.: Policy-based autonomic control service. In: Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'04). p. 99–102. IEEE Computer Society (2004)

[13] Bagci, F., Petzold, J., Trumler, W., Ungerer, T.: Ubiquitous mobile agent system in a P2P- network. In: in UbiSys-Workshop at the Fifth Annual Conference on Ubiquitous Computing. p. 12–15 (2003)

[14] Bao, L., Intille, S.S.: Activity recognition from user-annotated acceleration data. p. 1–17. Springer (2004)

[15] Becker, C., Schiele, G., Gubbels, H., Rothermel, K.: BASE " a Micro-Broker-Based middleware for pervasive computing. In: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications. p. 443–. PERCOM '03, IEEE Computer Society, Washington, DC, USA (2003)

[16] Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D.: A survey of context modelling and reasoning techniques. Pervasive and Mobile Computing 6(2), 161–180 (Apr 2010)

[17] Bieber, G., Architect, L., Ci, I.: Introduction to Service-Oriented programming. In: Openwings, URL = http://www.openwings.org (2001)

[18] Black, D., Golub, D.B., Julin, D.P., Rashid, R.F., Draves, R.P., Dean,

R.W., Forin, A., Barrera, J., Tokuda, H., Malan, G., Bohman, D.: Microkernel operating system architecture and mach. In: In Proceedings of the USENIX Workshop on Micro-Kernels and Other Kernel Architectures. p. 11–30 (1992)

[19] Blair, G.S., Coulson, G., Robin, P., Papathomas, M.: An architecture for next generation middleware. In: Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing. p. 191–206. Middleware '98, Springer-Verlag, London, UK (1998)

[20] Bobrow, D.G., Gabriel, R.P., White, J.L.: CLOS in context: the shape of the design space. p. 29–61. MIT Press, Cambridge, MA, USA (1993)

[21] Bouix, E., Dalmau, M., Roose, P., Luthon, F.: A multimedia oriented component model. AINA 2005 - The IEEE 19th International Conference on Advanced Information Networking and Applications (2005)

[22] Bouix, E., Roose, P., Dalmau, M.: The korrontea data modeling. In: Proceedings of the 1st international conference on Ambient media and systems. p. 6:1–6:10. Ambi-Sys '08, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium (2008)

[23] Bouzeghoub, A., Taconet, C., Jarraya, A., Do, N., Conan, D.: Complementarity of process-oriented and ontology-based context managers to identify situations. In: ICDIM. pp. 222–229 (2010)

[24] Carzaniga, A., Rosenblum, D.S., Wolf, A.L.: Design and evaluation of a wide-area event notification service. ACM Trans. Comput. Syst. 19, 332–383 (Aug 2001)

[25] Chefrour, D.: Developing component based adaptive applications in mobile environments. In: Proceedings of the 2005 ACM symposium on Applied computing. p. 1146–1150. SAC '05, ACM, New York, NY, USA (2005)

[26] Chen, H., Finin, T., Joshi, A.: Using OWL in a pervasive computing broker. Workshop on Ontologies in Agent Systems, AAMAS-2003 (Jul 2003)

[27] Chen, H., Finin, T., Joshi, A.: Semantic web in the context broker architecture. In: Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04). p. 277–. PERCOM '04, IEEE Computer Society, Washington, DC, USA (2004)

[28] Chen, H., Perich, F., Finin, T., Joshi, A.: SOUPA: standard ontology for ubiquitous and pervasive applications. In: In International Conference on Mobile and Ubiquitous Systems: Networking and Services. p. 258–267 (2004)

[29] Cheng, B.H., Lemos, R., Giese, H., Inverardi, P., Magee, J. (eds.): Software Engineering for Self-Adaptive Systems. Springer-Verlag, Berlin, Heidelberg (2009)

[30] Cheverst, K., Mitchell, K., Davies, N.: Design of an object model for a context sensitive tourist GUIDE. In: Computers and Graphics. p. 883–891 (1999)

[31] Cook, D.J., Augusto, J.C., Jakkula, V.R.: Ambient intelligence: Technologies, applications, and opportunities. Pervasive and Mobile Computing 5(4), 277–298 (Aug 2009)

[32] Dalmau, M., Roose, P., Laplace, S.: Context aware adaptable applications - a global approach. CoRR abs/0909.2090 (2009)

[33] Dobson, S., Denazis, S., Fernández, A., Gaïti, D., Gelenbe, E., Massacci, F., Nixon, P., Saffre, F., Schmidt, N., Zambonelli, F.: A survey of autonomic communications. ACM Trans. Auton. Adapt. Syst. 1, 223–259 (Dec 2006)

[34] Dowling, J.: The decentralised coordination of self-adaptive components for autonomic distributed systems. Ph.D. thesis, Department of Computer Science, Trinity College Dublin (2004)

[35] Eliassen, F., Gj\orven, E., Eide, V.S.W., Michaelsen, J.A.: Evolving self-adaptive services using planning-based reflective middleware. In: Proceedings of the 5th workshop on Adaptive and reflective middleware (ARM '06). p. 1–. ARM '06, ACM, New York, NY, USA (2006)

[36] Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.: The many faces of publish/subscribe. ACM Comput. Surv. 35, 114–131 (Jun 2003)

[37] France, R., Rumpe, B.: Model-driven development of complex software: A research roadmap. In: 2007 Future of Software Engineering. p. 37–54. FOSE '07, IEEE Computer Society, Washington, DC, USA (2007)

[38] Frank, A.U.: Tiers of ontology and consistency constraints in geographical information systems. International Journal of Geographical Information Science 15(7), 667 – 678 (2001)

[39] Geihs, K., Barone, P., Eliassen, F., Floch, J., Fricke, R., Gjorven, E., Hallsteinsen, S., Horn, G., Khan, M.U., Mamelli, A., Papadopoulos, G.A., Paspallis, N., Reichle, R., Stav, E.: A comprehensive solution for application-level adaptation. Softw. Pract. Exper. 39, 385–422 (Mar 2009)

[40] Google: What is android. Google (2011), http://developer.android.com/guide/basics/what-is-android.html

[41] Greenwood, P., Blair, L.: Using dynamic Aspect-Oriented programming to implement an autonomic system. Tech. rep., Proceedings of the 2003 Dynamic Aspect Workshop (DAW04 2003), RIACS (2003)

[42] Gu, T., Pung, H.K., Zhang, D.Q., Pung, H.K., Zhang, D.Q.: A bayesian approach for dealing with uncertain contexts (2004)

[43] Gu, T., Pung, H.K., Zhang, D.Q.: A service-oriented middleware for building context-aware services. J. Netw. Comput. Appl. 28, 1–18 (Jan 2005)

[44] Gu, T., Wang, X.H., Pung, H.K., Zhang, D.Q.: An ontology-based context model in intelligent environments. In: IN PROCEEDINGS OF COMMUNICATION NETWORKS AND DISTRIBUTED SYSTEMS MODELING AND SIMULATION CONFERENCE. p. 270–275 (2004)

[45] Haghighi, P.D., Krishnaswamy, S., Zaslavsky, A., Gaber, M.M.: Reasoning about context in uncertain pervasive computing environments. In: Proceedings of the 3rd European Conference on Smart Sensing and Context. p. 112–125. EuroSSC '08, Springer-Verlag, Berlin, Heidelberg (2008)

[46] Halpin, T.: Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design. The Morgan Kaufmann Series in Data Management Systems (2001)

[47] Hamida, A.B., Mouël, F.L., Frénot, S., Ahmed, M.B.: Déploiement adaptatif d'applications orientées services sur environnements contraints. Technique et Science Informatiques 30, 59–91 (2011), d.2.12.1: Distributed objects, D.: Software/D.4: OPERATING SYSTEMS/D.4.2: Storage Management/D.4.2.0: Allocation/deallocation strategies, C.: Computer Systems Organization/C.2: COMPUTER-COMMUNICATION NETWORKS/C.2.4: Distributed Systems/C.2.4.1: Distributed applications, C.2.1.10: Wireless communication

[48] Hasan, K., Rubaiyeat, H.A., Lee, Y., Lee, S.: A reconfigurable HMM for activity recognition (2008)

[49] Heimbigner, D., Wolf, A.L., Heimbigner, D., Wolf, E.L., Wolf, E.L.: Intrusion management using configurable architecture models. Tech. rep. (2002)

[50] Henricksen, K., Livingstone, S., Indulska, J.: Towards a hybrid approach to context modelling, reasoning and interoperation. In: Proceedings of First International Workshop on Advanced Context Modelling, Reasoning And Management (2004)

[51] Herrmann, K., Mühl, G., Jaeger, M.A.: MESHMdl event spaces – a coordination middleware for self-organizing applications in ad hoc networks. Pervasive and Mobile Computing 3(4), 467–487 (Aug 2007)

[52] Hong, X., Nugent, C., Mulvenna, M., McClean, S., Scotney, B., Devlin, S.: Evidential fusion of sensor data for activity recognition in smart homes. Pervasive and Mobile Computing 5(3), 236–252 (Jun 2009)

[53] Issarny, V., Caporuscio, M., Georgantas, N.: A perspective on the future of middleware-based software engineering. In: 2007 Future of Software Engineering. p. 244–258. FOSE '07, IEEE Computer Society, Washington, DC, USA (2007)

[54] Kanda, T., Glas, D.F., Shiomi, M., Ishiguro, H., Hagita, N.: Who will be the customer?: a social robot that anticipates people's behavior from their trajectories. In: Proceedings of the 10th international conference on Ubiquitous computing. p. 380–389. UbiComp '08, ACM, New York, NY, USA (2008)

[55] Kasteren, T.v., Krose, B.: Bayesian activity recognition in residence for elders. In: Intelligent Environments, 2007. IE 07. 3rd IET International Conference on. pp. 209–212 (2007)

[56] Kasteren, T.v., Noulas, A., Englebienne, G., Kröse, B.: Accurate activity recognition in a home setting. In: UbiComp '08: Proceedings of the 10th international conference on Ubiquitous computing. pp. 1–9. ACM, Seoul, Korea (2008)

[57] Keeney, J., Cahill, V.: Chisel: a policy-driven, context-aware, dynamic adaptation framework. In: Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on. pp. 3–14 (2003)

[58] Kephart, J., Walsh, W.: An artificial intelligence perspective on autonomic computing policies. In: Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop on. pp. 3–12 (2004)

[59] Kepner, C.H., Tregoe, B.B.: The Rational Manager: A Systematic Approach to Problem Solving and Decision-Making. McGraw-Hill (1965)

[60] Khac, A.P.: A Model-driven Feature-based Approach to Runtime Adaptation of Distributed Software Architectures. Ph.D. thesis, Computer Science Department, TELECOM Bretagne (2010)

[61] Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J., Irwin, J., Akşit, M., Matsuoka, S.: Aspect-oriented programming. In: ECOOP'97 — Object-Oriented Programming, vol. 1241, pp. 220–242. Springer-Verlag (1997)

[62] Lange, D.B., Mitsuru, O.: Programming and Deploying Java Mobile Agents Aglets. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edn. (1998)

[63] Leclercq, M., Quéma, V., Stefani, J.: DREAM: a component framework for the construction of resource-aware, reconfigurable MOMs. In: Proceedings of the 3rd workshop on Adaptive and reflective middleware. p. 250–255. ARM '04, ACM, New York, NY, USA (2004)

[64] Lim, S., Helal, A.: Encapsulation and Entity-Based approach of interconnection between sensor platform and middleware of pervasive computing. In: Ubiquitous Computing Systems, Lecture Notes in Computer Science, vol. 4239, pp. 500–515–515. Springer Berlin / Heidelberg (2006)

[65] Loke, S.W.: Incremental awareness and compositionality: A design philosophy for context-aware pervasive systems. Pervasive and Mobile Computing 6(2), 239–253 (Apr 2010)

[66] Louberry, C., Roose, P., Dalmau, M.: Kalimucho: Contextual deployment for qos management. DAIS'11 - 11th IFIP International Conference on Distributed Applications and Interoperable Systems (June 2011)

[67] Maes, P.: Situated agents can have goals. Robotics and Autonomous Systems 6(1-2), 49–70 (Jun 1990)

[68] Mamei, M., Zambonelli, F.: Programming pervasive and mobile computing applications with the TOTA middleware. In: Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04). p. 263–. PERCOM '04, IEEE Computer Society, Washington, DC, USA (2004)

[69] Mckeever, S., Ye, J., Coyle, L., Bleakley, C., Dobson, S.: Activity recognition using temporal evidence theory. J. Ambient Intell. Smart Environ. 2, 253–269 (Aug 2010)

[70] Microsoft: Microsoft .NET Framework. Microsoft (2011), http://www.microsoft.com/net/default.aspx

[71] Moore, D., Essa, I.: Recognizing multitasked activities from video using stochastic context-free grammar. In: Eighteenth national conference on Artificial intelligence. p. 770–776. American Association for Artificial Intelligence, Menlo Park, CA, USA (2002)

[72] Moser, O., Rosenberg, F., Dustdar, S.: VieDAME - flexible and robust BPEL processes through monitoring and adaptation. In: Companion of the 30th international conference on Software engineering. p. 917–918. ICSE Companion '08, ACM, New York, NY, USA (2008)

[73] Nicklas, D., Mitschang, B.: The nexus augmented world model: An extensible approach for mobile, Spatially-Aware applications. In: Proceedings of the 7th International Conference on Object-Oriented Information Systems. pp. 392–401 (2001)

[74] OMG: Model Driven Architecture (MDA) Specification. OMG (2010), http://www.omg.org/mda/specs.htm

[75] OpenSOA: Service component architecture specifications (2007), http://www.osoa.org/display/Main/Service+Component+Architecture+Specifications

[76] Oreizy, P., Gorlick, M., Taylor, R., Heimbigner, D., Johnson, G., Medvidovic, N., Quilici, A., Rosenblum, D., Wolf, A.: An Architecture-Based approach to Self-Adaptive software (1999)

[77] Patel, S., Robertson, T., Kientz, J., Reynolds, M., Abowd, G.: At the flick of a switch: Detecting and classifying unique electrical events on the residential power line (Nominated for the best paper award). In: UbiComp 2007: Ubiquitous Computing, pp. 271–288 (2007)

[78] Patterson, D., Liao, L., Fox, D., Kautz, H.: Inferring High-Level behavior from Low-Level sensors. In: UbiComp 2003: Ubiquitous Computing, pp. 73–89 (2003)

[79] Peltz, C.: Web services orchestration and choreography. Computer 36(10), 46–52 (2003)

[80] Picco, G.P., Cugola, G., Murphy, A.L.: Efficient Content-Based event dispatching in the presence of topological reconfiguration. In: Proceedings of the 23rd International Conference on Distributed Computing Systems. p. 234–. ICDCS '03, IEEE Computer Society, Washington, DC, USA (2003)

[81] Ranganathan, A., McGrath, R.E., Campbell, R.H., Mickunas, M.D.: Use of ontologies in a pervasive computing environment. Knowl. Eng. Rev. 18, 209–220 (Sep 2003)

[82] Renaud, P., Lionel, S., Laurence, D., Gérard, F.: Jac: A flexible solution for aspect-oriented programming in java. Lecture notes in computer science (2001)

[83] Romero, D., Rouvoy, R., Seinturier, L., Chabridon, S., Conan, D., Pessemier, N.: Enabling Context-Aware web services: A middleware approach for ubiquitous environments. In: Sheng, M., Yu, J., Dustdar, S. (eds.) Enabling Context-Aware Web Services: Methods, Architectures, and Technologies, pp. 113–135. Chapman and Hall/CRC (2010)

[84] Roose, P.: De la réutilisation à l'adaptabilite. Tech. rep., Université de Pau et des Pays de l'Adour (2008)

[85] Rouvoy, R., Beauvois, M., Lozano, L., Lorenzo, J., Eliassen, F.: MUSIC: an autonomous platform supporting self-adaptive mobile applications. In: Proceedings of the 1st workshop on Mobile middleware: embracing the personal communication device. p. 6:1–6:6. MobMid '08, ACM, New York, NY, USA (2008)

[86] Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach (2nd Edition). Prentice Hall (Dec 2002)

[87] Ryoo, M., Aggarwal, J.: Recognition of composite human activities through Context-Free grammar based representation. Computer Vision and Pattern Recognition, IEEE Computer Society Conference on 2, 1709–1718 (Oct 2006)

[88] Salber, D., Dey, A.K., Abowd, G.D.: The context toolkit: aiding the development of context-enabled applications. In: Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit. pp. 434–441. CHI '99, ACM, New York, NY, USA (1999)

[89] Salehie, M., Li, S., Asadollahi, R., Tahvildari, L.: Change support in adaptive software: A case study for Fine-Grained adaptation. In: Proceedings of the 2009 Sixth IEEE Conference and Workshops on Engineering of Autonomic and Autonomous Systems. p. 35–44. IEEE Computer Society, Washington, DC, USA (2009)

[90] Salehie, M., Li, S., Tahvildari, L.: Employing aspect composition in adaptive software systems: a case study. In: Proceedings of the 1st workshop on Linking aspect technology and evolution. p. 17–21. PLATE '09, ACM, New York, NY, USA (2009)

[91] Santos, A.C., Cardoso, J.M., Ferreira, D.R., Diniz, P.C., Cha?nho, P.: Providing user context for mobile and social networking applications. Pervasive and Mobile Computing 6(3), 324–341 (Jun 2010)

[92] Satoh, I.: Physical mobility and logical mobility in ubiquitous computing environments. In: Proceedings of the 6th International Conference on Mobile Agents. p. 186–202. MA '02, Springer-Verlag, London, UK, UK (2002)

[93] Schilit, B.N., Adams, N., Want, R.: Context-Aware computing applications. In: IN PROCEEDINGS OF THE WORKSHOP ON MOBILE COMPUTING SYSTEMS AND APPLICATIONS. p. 85–90. IEEE Computer Society (1994)

[94] Schmidt, A., Beigl, M., w. Gellersen, H.: There is more to context than location. Computers and Graphics 23, 893–901 (1998)

[95] Schmidt, D., Stal, M., Rohnert, H., Buschmann, F.: Pattern-Oriented Software Architecture Volume 2: Patterns for Concurrent and Networked Objects. Wiley (2000)

[96] Schmidt, H., Hauck, F.J.: SAMProc: middleware for self-adaptive mobile processes in heterogeneous ubiquitous environments. In: Proceedings of the 4th on Middleware doctoral symposium. p. 11:1–11:6. MDS '07, ACM, New York, NY, USA (2007)

[97] Shi, D., Ding, B., Yin, G., Feng, J., Wang, H.: Research on policy-driven software self-adaptation mechanism. Journal of Frontiers of Computer Science and Technology 4(2), 115–123 (2010)

[98] Sousa, J.a.P., Garlan, D.: Aura: an architectural framework for user mobility in ubiquitous computing environments. In: Proceedings of the IFIP 17th World Computer Congress - TC2 Stream / 3rd IEEE/IFIP Conference on Software Architecture: System Design, Development and Maintenance. pp. 29–43. WICSA 3, Kluwer, B.V., Deventer, The Netherlands, The Netherlands (2002)

[99] Strang, T., Popien, C.: A context modeling survey. In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing (2004)

[100] Studer, R., Benjamins, V.R., Fensel, D.: Knowledge engineering: principles and methods. Data Knowl. Eng. 25, 161–197 (Mar 1998)

[101] Sun: JSR 220:Enterprise JavaBeansTM. Sun, version 3 edn. (2005)

[102] Sun: Jini framework. Sun (2007)

[103] Szyperski, C.: Component Software: Beyond Object-Oriented Programming. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edn. (2002)

[104] Tapia, E.M., Intille, S.S., Larson, K.: Activity recognition in the home using simple and ubiquitous sensors. In: In Pervasive. p. 158–175 (2004)

[105] Tesauro, G.: Reinforcement learning in autonomic computing: A manifesto and case studies. IEEE Internet Computing 11(1), 22–30 (2007)
[106] Tesauro, G., Chess, D.M., Walsh, W.E., Das, R., Segal, A., Whalley, I., Kephart, J.O., White, S.R.: A Multi-Agent systems approach to autonomic computing. In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1. p. 464–471. AAMAS '04, IEEE Computer Society, Washington, DC, USA (2004)
[107] Thomson, G., Terzis, S., Nixon, P.: Situation determination with reusable situation specifications. In: Proceedings of the 4th annual IEEE international conference on Pervasive Computing and Communications Workshops. p. 620–. PERCOMW '06, IEEE Computer Society, Washington, DC, USA (2006)
[108] Tigli, J., Lavirotte, S., Rey, G., Hourdin, V., Cheung-Foo-Wo, D., Callegari, E., Riveill, M.: WComp middleware for ubiquitous computing: Aspects and composite event-based web services. Annals of Telecommunications 64(3), 197–214–214 (Apr 2009)
[109] Turaga, P., Chellappa, R., Subrahmanian, V., Udrea, O.: Machine recognition of human activities: A survey. Circuits and Systems for Video Technology, IEEE Transactions on 18(11), 1473–1488 (2008)
[110] Vail, D.L., Veloso, M.M., Lafferty, J.D.: Conditional random fields for activity recognition. In: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems. p. 235:1–235:8. AAMAS '07, ACM, New York, NY, USA (2007)
[111] Verma, K., Sheth, A.: Autonomic web processes. In: Benatallah, B., Casati, F., Traverso, P. (eds.) Service-Oriented Computing - ICSOC 2005, Lecture Notes in Computer Science, vol. 3826, pp. 1–11. Springer Berlin / Heidelberg (2005), 10.1007/11596141_1
[112] Weiser, M.: The computer for the Twenty-First century. Scientific American 265(3), 94–104 (1991)
[113] Weiser, M.: Some computer science issues in ubiquitous computing. SIGMOBILE Mob. Comput. Commun. Rev. 3(3) (Jul 1999)
[114] Westerinen, A., Schnizlein, J., Strassner, J., Scherling, M., Quinn, B., Perry, J., Herzog, S., Huynh, A.N., Carlson: Terminology for policy-based management (2000)
[115] Weyns, D.: An architecture-centric approach for software engineering with situated multiagent systems. Ph.D. thesis, Katholieke Universiteit Leuven (2006)
[116] Wojek, C., Nickel, K., Stiefelhagen, R.: Activity recognition and Room-Level tracking in an office environment. In: Multisensor Fusion and Integration for Intelligent Systems, 2006 IEEE International Conference on. p. 25–30 (2006)
[117] Wooldridge, M.: An Introduction to MultiAgent Systems. John Wiley & Sons (Jun 2002)
[118] Xu, H., Shatz, S.M.: A framework for Model-Based design of Agent-Oriented software. IEEE Trans. Softw. Eng. 29, 15–30 (Jan 2003)
[119] Yang, J., Wang, J., Chen, Y.: Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers. Pattern Recognition Letters 29(16), 2213–2220 (Dec 2008)
[120] Ye, J., Dobson, S., McKeever, S.: Situation identification techniques in pervasive computing: A review. Pervasive and Mobile Computing In Press, Corrected Proof (2011)
[121] Yu, Z., Xiao-xing, M., Jian-nong, C., Ping, Y., Jian, L.: Software Agent-Virtualized application mobility in pervasive environments (2008)
[122] Zhang, D., Gu, T., Wang, X.: Enabling context-aware smart home with semantic web technologies. International Journal of Humanfriendly Welfare Robotic Systems 6(4), 12–20 (2005)