An adaptation platform for multimedia applications

CSC (Component, Service, Connector)

M. Derdour*[1], P. Roose[2], M. Dalmau[2], N. Ghoualmi Zine[1]

[1] University of BM Annaba, Computing Department, Algeria.
E-mail: {m.derdour, goualmi}@yahoo.fr
*Corresponding author

[2] LIUPPA – IUT of Bayonne, Computing Department, France.
E-mail: {roose, dalmau} @ iutbayonne.univ-pau.fr

Abstract: The trend toward ubiquitous services and any multimedia, the proliferation of mobile devices and the widespread use of wireless networks imply changes in the design, the implementation and the execution of software applications. Ubiquitous systems are dynamic systems that change their behavior according to user's needs and hardware capabilities at runtime. As it is not desirable to develop these systems from scratch every time, a specific software architecture providing opportunities for dynamic adaptation of systems is necessary. It must be able to create adaptations at runtime in order to provide a dynamic and adaptive behavior for users according to the evolving context. In this paper we present a supervised adaptation platform for applications based components.
The CSC platform (Component, Service and Connector) is based on a component/service model that allows adaptation of component-based applications and use service-oriented architecture for providing adaptation services to be embedded in adaptation connectors.

Keywords: Component; Connector; Software architecture; Multimedia; Adaptation.

1. Introduction

Ubiquitous systems are designed to make communication possible anytime, anyhow and anywhere. These systems must be used in different contexts depending on the environment of the user, his profile and his device. The mutualisation of the means of communication and the tendency towards all multimedia caused a major problem of heterogeneity of the multimedia flows exchanged between the components of such a system. The future multimedia ubiquitous systems must have capacities of adaptation, and thus being able to modify the system configuration and/or the multimedia contents at any time.

As part of ubiquitous computing, the environment of an application is composed of heterogeneous machines (PCs, PDAs, Smartphone, etc.) with various hardware resources (screen size, interaction modes, memory, battery, network interfaces, etc.) belonging to users with different needs and handling media from various types (video, sound, image, text). These characteristics require structuring of application in an organization of independents entities which cooperate and interact in order to facilitate its adaptation to the context of use.

Unfortunately this structural and behavioral organization does not make it possible to solve all heterogeneity problems. According to the respect of functional constraints, replacing a component with another one claims the satisfaction of several conditions and verifying several properties (homogeneity of the components, coherence of the assembly, stability of the application, traceability of adaptation choices, etc). That requires a reflection on the design (in particularly on the configuration of applications). This reflection consists in separating the functional aspects from those of the adaptation and to provide dynamic and reconfigurable mechanisms during the life cycle of the application.

The development of an adaptation system raises two questions: how to design a platform to ensure dynamic adaptations? How to design adapters to ensure the adaptations? The important aspects for the design of content adaptation solution are the diversity of content, description of the environment, context management and adaptation. As for heterogeneous environments, the adaptation occurs at several levels: the user environment (noisy, dark, geographical, etc.), the terminal (screen size, codec, etc.) and the components of application (communication interfaces, client-server, RPC, etc.). That is why the design and the implementation of such a system imply efforts in various fields.

There are two main axes for adaptation of component-based applications, that of the functional adaptation like MUSIC [19] and MADcAR [20] based on the reassembly or the reconfiguration, and that of the non-functional adaptation like SCL [21] based on the behavior of components. In our approach we worked on non-functional adaptations of flows exchanged between components in order to resolve the data interactions heterogeneity problem. We do not seek modification or replacement of features, but adapt them to the runtime context. This may result in adaptations of assemblies (call of different service, redeployment, replacement of components/services, etc.) in order to preserve the same functionality but with different quality of service.

The massive introduction of multimedia data with ubiquitous/pervasifs systems leads to handle various media types involving several problems influencing the interaction of the components such as the heterogeneity of data flows exchanged between those components. These problems deals with to the size of data (video streams are difficult to manage depending on the type of connection), the encoding of data (formats, codec, containers, encoding quality), modality (text received while the person is blinded, etc.). The adaptation services are a solution to solve this problem which represents one of the major challenges of such applications. They propose mechanisms allowing and ensuring the adaptation of multimedia data flows exchanged between heterogeneous components. According to this point of view, CSC proposes an executing adaptation platform of multimedia applications.

The main entity of our proposal is the connector entity (having a non-functional character), which proposes solutions to answer the problems of adaptation in the business components and allows resolution of heterogeneous data without changing the functionality of such system. It is represented by a component of first class. It is the first class because he does not just have the traditional roles associated with the communication but also supports the adaptation of data (in a unit way or by assembly of connectors). This type of connector is becoming reconfigurable and able to adapt the multimedia data flow according to the situation.

2. Motivation

Our main motivation is to provide a platform to maintain the consistency of configurations implemented by assembling heterogeneous components using MMSA (Meta-model for Multimedia Software Architecture) [1] which provides for new types of graphical interfaces and connectors with a richer semantics. The use of these interfaces

allows the automatic detection of heterogeneity points between components, while the use of adaptation connectors allows resolving these heterogeneities.

The multimedia communication needs services able to face heterogeneity on several levels: the context, the access devices, the communication network, the user, etc. It is necessary to integrate capacities to deal the heterogeneity problem, and to answer the changes of the context caused by the user, the application, the network or the access device. It is necessary therefore to develop platforms capable of ensuring the execution and the monitoring of multimedia applications.

In most adaptation and self-adaptation platforms [19, 20, 21] we find that:

a) The assembly management does not take into account the behavioural heterogeneity for the components of the software architecture;

b) Few platforms allow defining new connectors with ad hoc processing providing non-functional needs of components (adaptation, security, communication, conversion, etc.);

c) There is no direct and automatic match between architectures (models) and applications built using these architectures (instances).

In order to answer to these lacks, we propose CSC, a self-adaptation platform based on MMSA to describe software architectures for multimedia oriented application and providing adaptation capabilities. The platform is based on services and offer architecture with three layers particularly adapted to adaptation of multimedia flow (types, formats, properties) allows solving the heterogeneity problems of components.

3. MMSA Approach

The highlight of incompatibilities of data flows exchanged between components is a necessity in such approaches based components. Indeed, software architectures validate the functional aspects, which is not enough to ensure a realistic assembly and addressing the problems of heterogeneity of data flow exchanged. To detect these incompatibilities and to make possible a solution, a model-based approach called MMSA was proposed in [1]. It allows the description of software architectures using an UML-profile dedicated to express a software system as a collection of components which handle various types and formats of data and which interacts between them via connectors including adaptation connectors. MMSA provides a coherent configuration which can be used for the configuration and the execution of the application. Nevertheless, control and monitoring of application is necessary in order to answer to any dynamic change of context and preserved the coherence of the application.

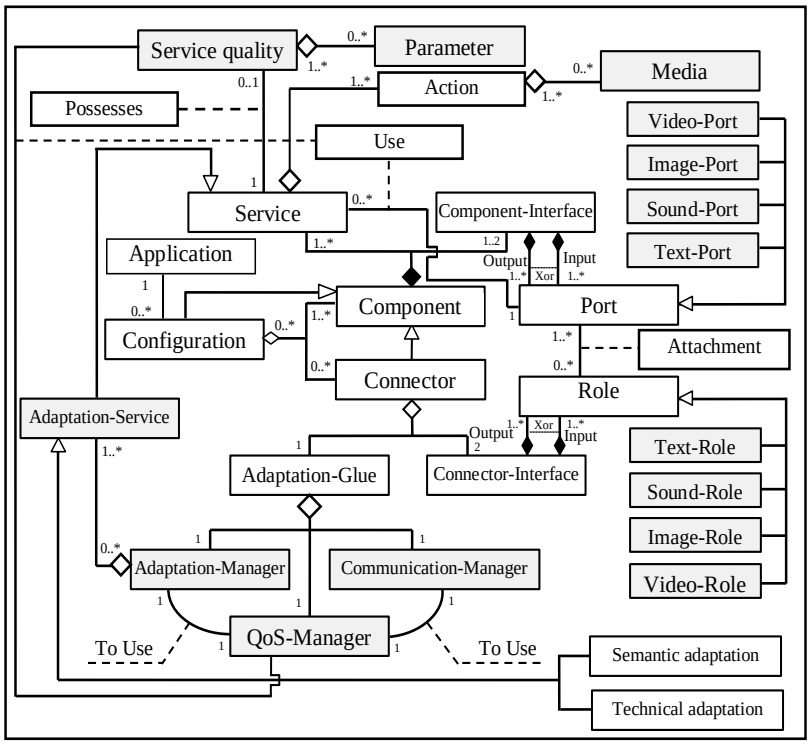


Figure 1. Class diagram of MMSA

MMSA is a generic meta-model for describing multimedia software architecture integrating concepts related to multimedia data and quality of service. This enabled him to present separately the data flow parameters and media in that they have a very important aspect in configurations and assemblies of components. Main characteristic of MMSA is the multimedia aspect and the separation between functional and non-functional aspect of components.

MMSA is a meta-model of multimedia software architecture integrating properties of data flow. The adaptation of data flow is dedicated to connectors called adaptation connectors. It integrates the adaptation services necessary as well as qualitative extensions of these services to provide a QoS measure reflecting the evolution of the data flow following the adaptation.

Connectors are used in order to link the components. A connector is composed of glue and two interfaces (required and provided). The types of adaptation connectors were proposed in [2] for ensuring the adaptation of multimedia flows exchanged between components of an application.

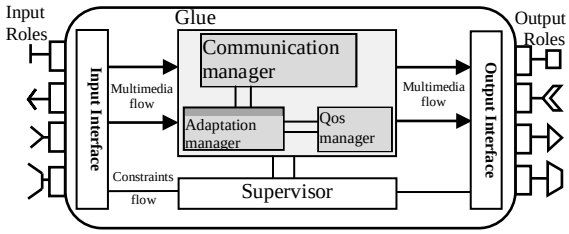


Figure 2. Model of adaptation connector

Allowing to heterogeneous components to interact is a significant task. The adaptation task is a non-functional aspect; the adaptation process is delegated to connectors. The role of the adaptation connector is to receive the data, to adapt them according to the QoS managers requirements (which makes it possible to modify the parameters of the adaptation services in order to manage the quality provided by the latter according to the needs) and to forward them to the component or the connector recipient according to the format and the type desired by this last. It should be noticed that it is possible to chain connectors where adaptation requires several operations provided by multiple connectors including when it is necessary to change network interface (Wifi -> Zigbee).

We propose to ensure the coherence of the applications, according to the context changes, a dynamic adaptation platform. The dynamic adaptation is the process by which a software application is modified in order to take into account a context changes. This platform monitors and controls the execution of multimedia applications in order to detect any change of the context. When a change event occurs, the platform seeks the possible solutions and makes the adequate decision for the adaptation of the application to the new context. Then, the platform seeks and selected the necessary adaptation services in order to integrate them in adaptation connectors and reassembled with the business components of the application.

4. Adaptation process in the CSC platform

The adaptation process is a sequence of steps allowing selecting the adaptation services needed and to make the assembling of connectors in order to ensure both communications of data between heterogeneous components.

The service term is perhaps one of the terms most used and most ambiguous in the software industry [3]. Usually, services are defined as features provided by a software system for others or for a human user [4]. In the context of SOA, services are provided by independent service providers that instantiate the software on their computers and publish the services that it provides using standardized mechanisms so they can be discovered and dynamically related to the components that need. A service is a behaviour defined by contract which can be produced and provided by any component to be used by other on single basis of contract [5].

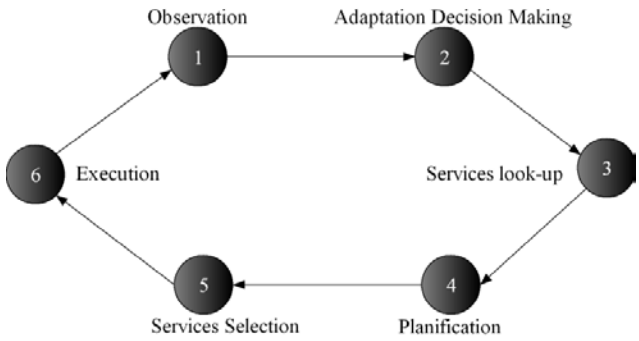


Figure 3. Process of dynamic adaptation

The dynamic adaptation is the process by which a software application is modified in order to take into account a change that it is on the environment level or on the application. It is about a process in six stages. It must first (1) observe the execution

environment, (2) decide appropriateness of the adaptation and the strategy appropriate to the detected situation, then (3) search the adaptation services necessary and (4) plan the activities to achieve in order to adopt the decided strategy, then (5) select the adaptation services capable of providing the requested adaptation and finally (6) realize the decided treatments. The figure3 presents an explanatory diagram of this process.

For each adaptation phase, several techniques are used. A summary of these techniques is presented in the following table:

Adaptation phase	Techniques used
Observation	− *Sensor (sensors)* − *Manual (observation)* − *Monitoring*
Decision	− *Systems of rules* − *Diagnosis based model* − *Optimization under constraints* − *Probabilistic Models* − *Machine learning*
Search services	− *Service discovery, UDDI (Universal Description Discovery and Integration)*
Planning	− *Graph Theory* − *Network Pert* − *Gantt* − *Planning AI (Artificial Intelligence Planning)*
Selection of services	− *System of rules;* − *Logic programming;* − *Finding a path in the adaptation graph*
Implementation	− *Web Services Invocation SOAP (Simple Object Access Protocol) ;* − *AOP;* − *Alternative Programming ;* − *Programming with Components.*

Table 1. ADAPTATION TECHNIQUES

The planning phase provides a graph of adaptation constitutes of adaptation services and connections between these services and the selection phase provides the most adequate path of adaptation.

To ensure the self-adaptation of the applications, we need three levels: the description, the supervision and the adaptation.

The figure 4 presents a functional view of the configuration and adaptation platform. This view is composed of a set of descriptors and functions. The functions are distributed on three levels: Description, supervision and Adaptation. The context descriptors and the manifests of components provide application requirements for components and adapters in order to find the right configuration of the application. The supervision allows to track changes in context and to update the application requirements. These changes can affect the configuration. The adaptation provides a reconfiguration of the application taking into account the problem of heterogeneity of components.

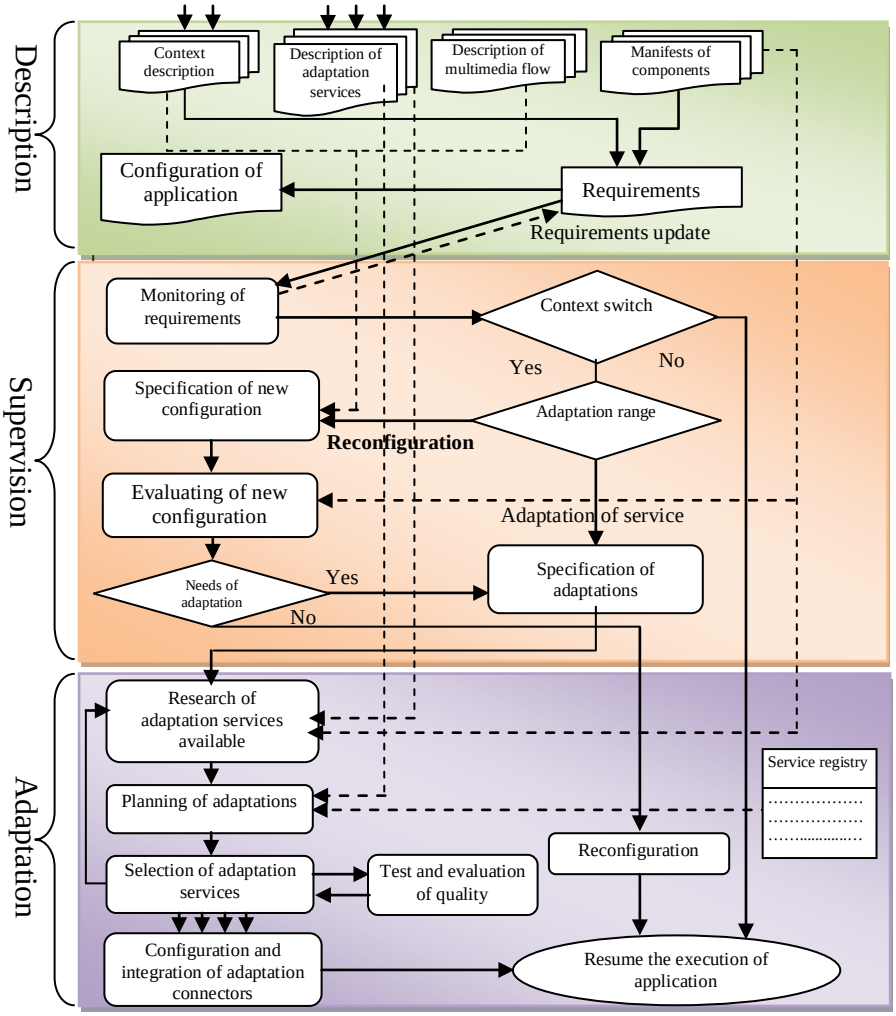


Figure 4. Process of configuration/reconfiguration

5. An adaptation approach for ubiquitous computing

The CSC adaptation platform is based on concepts of component, connector and services. Services are used by connectors to ensure the tasks of adaptation, while the connectors are used for communication, exchange and adaptation of multimedia data between components.

In MMSA [1] we presented the architectural aspect of multimedia applications and the mechanisms to verify the configurations of these applications. The CSC platform focuses on the behavioral and dynamics aspects of adaptation mechanisms. We will discuss the process of adaptation and self-adaptive of multimedia applications and the mechanisms of selection and integration of adaptation services.

In the CSC platform, we try to separate the concern of adaptation of components and the self-adaptation of services from the functional concerns of applications, which gives the possibility to delegate the additional complexity related to adaptation and self-

adaptation on the platform. The adaptation process is applied to the component model produced by MMSA to detect the heterogeneities points between components.

To ensure the data flow adaptation during the interactions between components, we propose a platform with a three tiered architecture (Figure 5). Each layer provides a dedicated task, namely:

a) ***Configuration layer*** *: it ensures dynamic reconfiguration of the application, detection of data flow heterogeneity problems inter-component and chek changes in the context;*

b) ***Adaptation layer****: it ensures the planning, the negotiation and the selection of adaptation services;*

c) ***Application layer*** *: it is responsible for running applications. It contains all the elements necessary for running applications such as components, services or relationships between services and connectors. The discovery of dynamic change of context, the supervision of the execution of the application and the detection of contracts failures are tasks ensured by the QoS manager.*

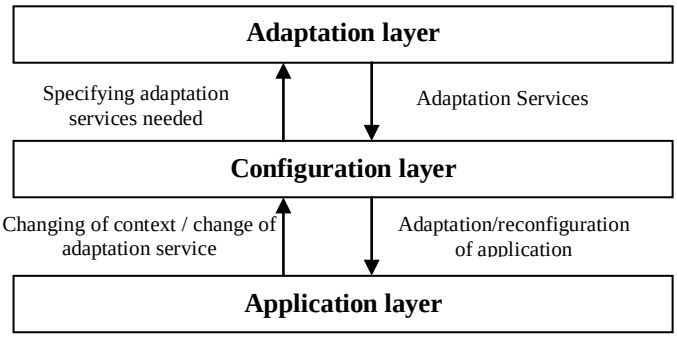


Figure 5. Different layers of CSC platform

Two interesting factors must be taken into account when implementing such a platform: the range of adaptation and the place of execution. The adaptations can be changes of interfaces (non-functional aspect), services or components (functional aspect). At the adaptation connector mechanism and the service parameters proposed in MMSA, we add the manipulation and change of service parameters that can influence the data produced by the adaptation services. Several approaches can be investigated in the adaptation at runtime: based-client, based-server, based-proxy and hybrid. We prefer the hybrid approach that give more flexibly to select freely the adaptation emplacement. Consequently, we follow to execute adaptation connectors in the client part, the server part or in its relied post, depending on the capabilities of the client and the context application.

A. CSC Platform

Managing service quality is an important task in the CSC platform. So, two QoS managers have been proposed to ensure the quality of service at the moment of application configuration and quality of service during the execution of the application and adaptation services. The first are interacts with the platform to monitor QoS changes at the application run-time phase. It provides also application adaptations for the new context. The second one is integrated at each adaptation connector. It manages service quality

within connector by services adaptation parameters. This manager is related to the QoS manager of platform in order to request a change of adaptation services if necessary.

The adaptation Platform is used for building architectures based components that ensure their adaptations under continuous context changes. It provides a monitor mechanism to control and pilot applications. Most of applications include: different types of media, discrete (text, images) or continuous (video, audio), several mobile devices with variety of capabilities and users moving profiles in ubiquitous computing environments. This involves additional difficulties related to self-adaptation.

The CSC platform provides adaptation by checking the application configurations coherence. It proposes adaptation system based services to increase flexibility in discovery service adaptation. The execution and monitoring of adaptation services are provided by adaptation connectors.

To illustrate our CSC platform objectives, we propose the following example:

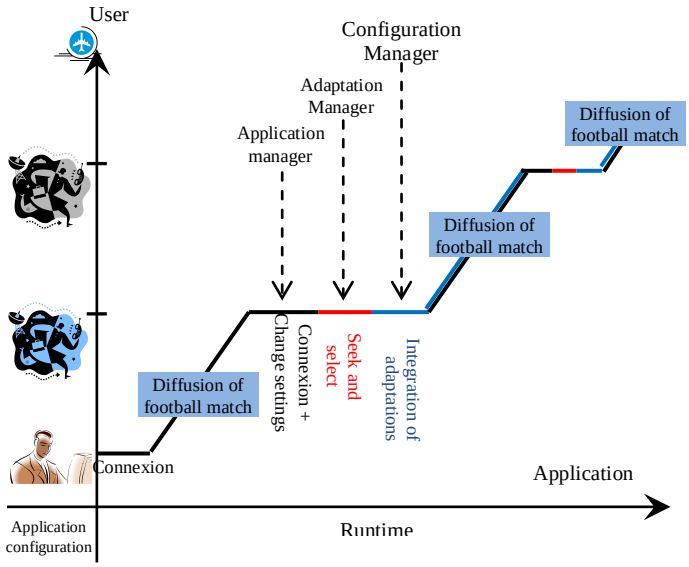


Figure 6. Scenarios of the user login

A user has a PC with a wireless connection (Figure 6). He wants to watch a football match from an internet site which provides the game in the Real Player format (.rm) which has a subscription. After 25 minutes of game, he received a phone call to join a friend at the airport. He needs to quit home but still want to watch the game. Since, he has a PDA with a 3G wireless connection he can do that with its. After obtaining the connection, the application detects its new PDA parameters and the absence of real media codec and therefore is unable to receive the video in the same format as with its PC due to physical characteristics (screen size, strain energy) and quality of connection (bandwidth, connection type). For preserving the connection, the application needs to adapt the video game to meet new requirements. So, it needs two adaptation services: the first deals with transcoding the video format from .rm codec to MP4 format which is supported by the PDA and the other one reduce its resolution to meet the bandwidth reduction to be compatible with the 3G. After finding the adequate services and after its integration, the user can follow the football match while driving to the airport.

At the end, the PDA battery is very low; instead of having no thing, he prefers to follow the match on his mobile phone, because he do not have a 3G subscription, he can only receive the sound. The same scenario as with the previous case, we need also another adaptation service (change of media type) from video to audio transformation in order to let the user continue to watch the game until the end.

B. The architecture of CSC platform

The key idea of CSC (see figure 7) is to use the services provided by the components available in the platform and services available to solve the heterogeneity problem between application components. It investigates on adaptation services of multimedia data to achieve a good configuration and to improve components interoperability. The connector mechanism provides services parameters adaptation to ensure quality changes or dynamic context.

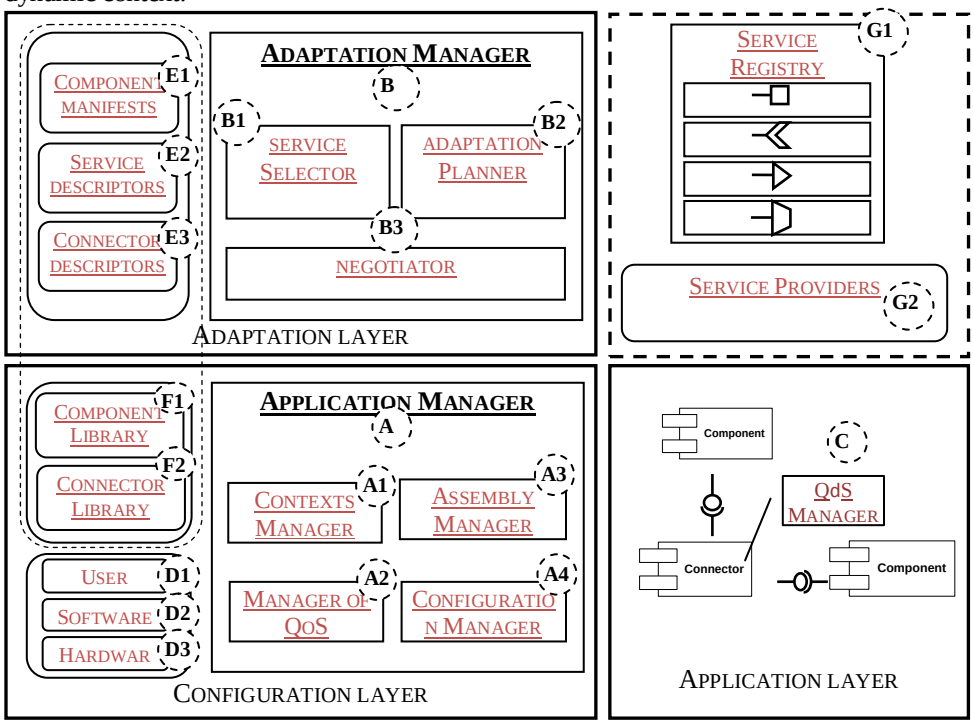


Figure 7. CSC Platform

The configuration layer encapsulates a library of components (F1) and connectors (F2) used for configuring applications, a database to store context information's: users (D1), software (D2) and hardware (D3) and an application manager that is the engine of "reasoning" of this layer. The Application Manager (A) is composed of a context manager (A1), QoS manager (A2), assembly manager (A3) and a configuration manager (A4). The application manager uses the model to describe the MMSA application architecture which allows it to detect the need for adaptation between the components of a configuration. Heterogeneity points are detected from an analysis of the manifestos of the components (E1) and context elements. The context manager (A1) is responsible for updating the context after a detection of change announced by the QoS manager (A2). Then, the need for adaptation is transferred to the adaptation layer as a specification of adaptation. This

layer encapsulates the descriptors of components (E1), connectors (E3) and services (E2) and an adaptation manager (B). The Adaptation Manager is composed of a service selector (B1), a planner (B2) and a service negotiator (B3). At the reception of adaptation request, the scheduler adaptation transforms specifications into a graph of adaptation that contains all possible paths of adaptation depending on the available adaptation services from descriptions of connectors, descriptors of services and service registry (G1). The service selector uses the adaptation process to find the best way of adaptation by constructing a list of adaptation services classified by type and quality. This list will be used thereafter if needed to change the adaptation services. Then, the adaptation service requests the negotiator to negotiate and to establish of contracts with service providers (G2). Finally, the assembler ensures the assembly of components and connectors after their integration that depending on the selected configuration.

6. Layers description of CSC platform

The CSC platform (Figure 7) is divided into three layers: the configuration layer, the adaptation layer and the application layer. The configuration manager at the configuration layer is responsible for selecting suitable components and configurations; it is also responsible for operations of assembly and reassembly of components. We use MMSA to describe the application architecture which allows it to identify the adaptation needs of application components and build the adaptation connectors. These needs will be forwarded to the Adaptation Manager. At the configuration layer, the QoS manager provides conflict resolution between the possible configurations and the adaptation services installed, and also for monitoring applications and ensuring the good adaptation of the application by checking the context changes. The context manager exploits profiles changes.

At the adaptation layer, the adaptation manager ensures the selection of adaptation services. It is handled by three services: the planner, the selector and the negotiator. The first provides the adaption plan for each adaptation need; the second ensures the selection of adaptation services, while the last provides the negotiation and establishment of contracts with service providers.

A. Application layer

The application needs to detect the context changes, but also the selection of configuration of the application that maintain the sufficient quality to meet new context. So, it is necessary to discover dynamically the adaptation services as soon as they are useful and their disappearance in order to ensure their replacement. This work is provided by the configuration layer.

A QoS manager exists at each adaptation connector. It informs the QoS manager of configuration layer of any change need of adaptation service due to unavailability or insufficiency of quality.

B. Configuration layer

This layer is composed of the application manager, the context information (users, software and hardware) and a repository (components and connectors). The application manager is composed of four managers:

- The configuration manager provides all possible configurations for an application. It's also capable to detect components incompatibilities in a given configuration (heterogeneity data flow level) by checking components manifest that contain inputs/outputs details of components. A manifest can describe components as an

abstract model (without implementation details). It separates the abstract description of the functionality offered by a component, as well as concrete details of the component such as "how" and "where" functionality is available and describe data handled by components (data type, format , time constraints : high/low , etc..).

- The context manager ensures the context updating during supervised changes by the QoS manager. It provides further information about the environment (user, software, hardware) to the configuration manager. For example: for each web application: the type and version of browser, the navigation terminal, preferences and physical characteristics of the user, etc. to better select the configuration and its components.
- The QoS Manager provides control and monitor applications by checking all possible context changes that affect the good execution of the application. It cooperates with QoS manager at each level of connector adaptation.
- The assembly manager ensures the assembly/reassembly of components and connectors.

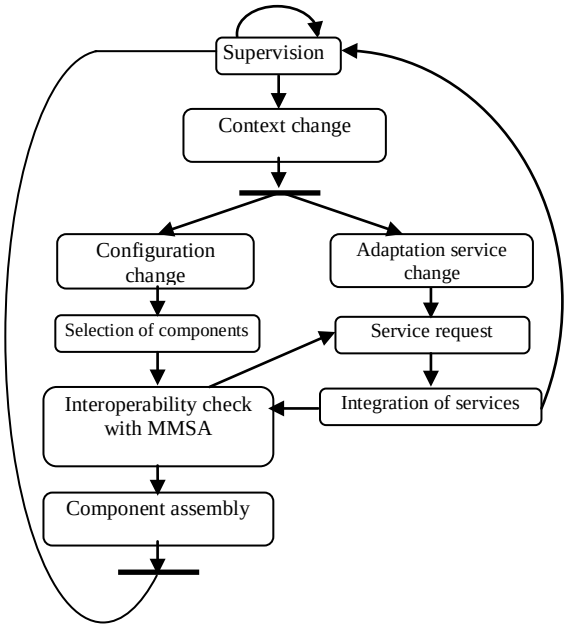


Figure 8. Activity diagram of configuration layer

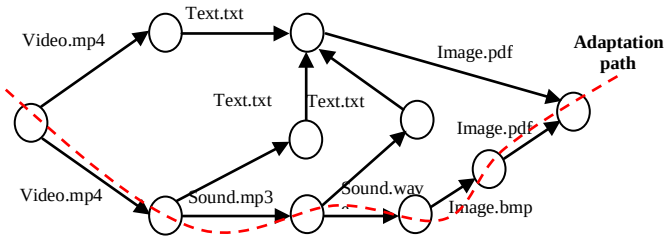
After detecting a context changes, the platform begins by a replacement of adaptation services to respond to this change, if the replacement of adaptation services is not sufficient and does not meet the new context, the platform precedes the replacement of some components followed by reassembling or reconfiguring of application if necessary. In the case of reconfiguration, we must select carefully new components of the new configuration and then checking the consistency of the configuration using the MMSA approach [1]. Then, the adaptation layer provides the necessary services and integrates them into the connectors (Figure 8).

C. Adaptation layer

This layer is composed of the adaptation manager and a set of descriptors: services, components and connectors. The adaptation manager is composed of three components:

- The adaptation planner specifies the required adaptations in terms of adaptation services. Next, it consults the services registry to find available services. Finally, it builds an adaptation graph whose vertices are adaptation services and edges are links between them.

Example: We consider an adaptation of conference oral presentation video into PDF file. This adaptation goes through several steps: sound extraction, transmoding (sound to text and transmoding text to image as PDF format). This makes a total of three adaptation services. By consulting the registry service, we may find the following graph:



- The service selector provides the best adaptation way from a graph, in the case of several services of the same natures or several paths providing the same adaptation. We use graph theory to find the best adaptation path (Fig. 9).

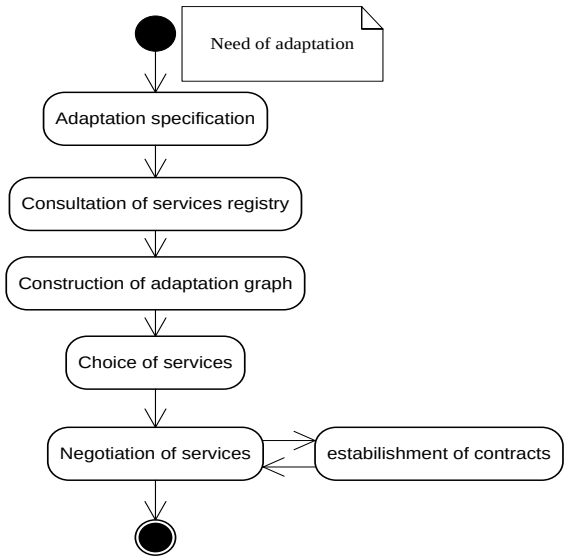


Figure 9. Activity diagram of adaptation layer

- The designer provides weights for each criteria of quality (e.g. resolution, compression ratio, etc…) according to their needs. Then it calculates QoS for each service in the graph and QoS for each adaptation path. The comparison of QoS adaptation paths allows us to select the best path [22].

There are two kinds of service quality: static and dynamic. The static QoS management is achieved through a process of selecting between several services those providing different qualities, while the dynamic QoS management is provided by the connector that handles the adaptation as service parameters changes to satisfy context execution.

- The service negotiator negotiates with services provider for selecting and establishing services contracts using the protocol SLA (Service Level Agreement).

A key concept for service-oriented architecture is the service contract standard [6] which is used for expressing services. QoS properties are generally negotiated between the provider and the service consumer, and are described as SLA contract. A service level is used to describe the expected performance (e.g. response time and availability) and properties (fracturing, conditions termination and penalties) for the SLA contract violation [7]. In our system we focus only on expected performance.

Example: The Service Level Agreement can contain several performance measures (service) matches with service objectives. We consider for example, images adaptations, the parameters that we measure in this case are:

- TA (Adaptation Time): average time needed to deliver service adaptation.
- TT: (Transfer Time): average time needed to transfer the adaption image.
- QA: (Adaptation Quality): degree of output quality compared to input quality.

An SLA can be created after fixing a selection level of service among several predefined or, in more complex cases, after customization via a negotiation process. An SLA may be valid for a limited period (for example the adaptation of a set of images) or may be explicit (for example the end of direct diffusion). During the SLA, the service provider monitors and adapts its resources to avoid SLA violations.

The negotiator Manager is responsible for negotiating and implementing SLAs with selected providers by the Adaptation Manager. After establishment of service contracts, the QoS manager of the connector adaptation takes control and supervision of adaptation services. If the adaptation service cannot meet the needs of this connector components related to adaptation, the latter applies to the replacement of service from the QoS manager of the configuration layer.

7. Discussion

Adaptations of component-based applications refer to the capacity of a system to adapt to the evolving needs of the users and the context by exploiting knowledge on its configuration and the characteristics of QoS of its constitutive components. The adaptation based planning [8, 9, 10, 11 and 12] is one of the adaptation approaches of component-based applications. In MUSIC [13], this knowledge is provided in the form of a model directed by QoS which describes the abstracted composition, relevant dimensions of QoS and how they are affected when there alternatives of components configuration exist. This model is exploited by the adaptation middleware to select, connect and deploy a configuration of components providing the best utility. The utility is measured by the degree of achievement of user preferences while optimizing the use of device resources [14, 9].

Adaptive Service Grids (ASG) [15] and VieDAME [16] provide the dynamic services composition and services attachments for the adaptation. AGC propose the adaptation life-

cycle for delivering adaptation services. It consists of three sub-cycles: the planification, the attachment of the semantic specification to a concrete service and the integration. The delivery cycle is an application that describes the service semantics and its functionalities but does not describe the concrete services to be executed. VieDAME offers a monitor system that observes BPEL (Business Process Execution Language) process efficiency and automatically replacing the service that caused the efficiency degradation. Compared to our approach, ASG and VieDAME are based only on the planning of services composition on demand that describe semantic properties of the request service. Thus, both approaches not guarantee a coherent configuration of components and services but our platform provides such consistency for the ubiquitous applications based components and separates the adaptation and the application concerns that ensured by the adaptation connectors.

Menasce and Dubey [17] propose a QoS approach in SOA. Client send services demand to a QoS broker that selects an appropriate service provider that maximizes the utility function of the client under cost constraints. The approach assumes that service providers register with the broker that provides solutions for each service resource and for each service costs. The QoS broker uses analytic models to predict the values of QoS for various services that could be chosen under varying conditions of work. This is an interesting approach for both customer and supplier. The client is discharged to lead service discovery and negotiation, the supplier determines the management support of QoS. This approach requires that the client device allows access to the broker, which may not be possible in ubiquitous environments. This is different from our approach, we consider the proposed properties as solutions to select the best configuration of the application and allow the connectors to adapt to the components needs.

CARISMA is a peer-to-peer mobile middleware exploiting the reflection principle to support the construction of adaptable context-aware applications [18]. The services and the adaptation policies are installed and uninstalled on the fly. CARISMA can automatically trigger adaptation of applications deployed during detection of changes in context. CARISMA uses the service to select the application profiles that are used to determine the appropriate action to a particular event context. If there are conflicts in the profiles, it uses a bidding process to resolve them. Unlike our approach, CARISMA does not address to discovery of remote services that can trigger the reconfigurations of application and does not allow the search of new configurations.

The two conceptual models SeCSE (http://www.secse-project.eu/) and PLASTIC (http://www.ist-plastic.org/) focus on service-oriented systems. PLASTIC model is an extension of the SeCSE model which introduces new concepts such as context, location and level of service credibility. Our approach and PLASTIC model share the SOA approach and the components-based software development. However, the conceptual model of our approach is focused component while that of PLASTIC is centered service.

The MUSIC model describes the abstract composition as a set of roles in collaboration with ports that represent the functionality provided or required by collaborator components. The properties and predictive functions associated with ports define how the QoS properties and the needs of components resource are influenced by the QoS properties of components on which they depend. This middleware of adaptation proposes a typing service in order to change it in the case of absence of this service or in the case of context change. Our platform provides a data typing in order to detect and resolve the problem of heterogeneity between components which handling different media types. Typing services used in our approach to classify adaptation services by provided adaptation that gives us the opportunity to choose the best services in terms of quality and change the adaptation service in the case of disappearance or reduction in quality.

The CSC platform can address the problem of context changing by proposing a set of services that can recover the system stability. The use of techniques and services offered by service-oriented architecture (such as: service publication, service discovery, service replacement) can provide a solution to the CSC platform.

8. Conclusion

The increasing use of the divide goes both with an increasing in multimedia information production/consumption make necessary the transformation and the adaptation of contents. This need is justified by an increasing demand of access and exchange of information in any place and on heterogeneous platforms.

CSC is a platform for execution and adaptation of multimedia component-based applications, it ensures the adaptation of applications to the context, offering adaptation connectors to adapt and manage changes caused by a context variation. Next, it provides a reconfiguration of applications according to the new context and using the MMSA approach to validate the new configuration.

One of the important benefits of the services is the ability to compose them in order to build services more complex with higher semantic level. In the composition process, the selection services step allows to choose the specific services that will be participating entities in the composition of complex adaptation service.

The proposed solution is a platform for distributed adaptation, which, from an abstract component called manifest, is able to detect the incompatibility points between components according to MMSA approach. Then it built the adaptation connectors from the adaptation services provided by the environment. Then it can choose and incorporate these services in adaptation connectors. Finally, it ensures the assembly / reassembly of components and adaptation connectors. By using this design and adaptation process, our system remains consistent.

Other research includes the description of services and the manifest, and the discovery and the composition. Other research areas include the selection of services and the criteria for distinguishing between services.

9. References

[1] Derdour, M., Roose, P., Dalmau, M., Ghoualmi-Zine, N., Alti, A. MMSA: Metamodel Multimedia Software Architecture. Advances In Multimedia, Hindawi Ed. - vol. 2010, Article ID 386035, 17 pages, 2010. doi:10.1155/2010/386035.

[2] Derdour, M., Roose, P., Dalmau, M., Ghoualmi-Zine, N. Typing of Adaptation Connectors in MMSA Approach Case Study: Sending MMS. International Journal of Research and Reviews in Computer Science (IJRRCS) - pp. 39-49, Vol. 1, No. 4, 12/2010 - ISSN: 2079-2557.

[3] Baida, Z., Gordijn, J., Omelayenko, B. A shared service terminology for online service provisioning. In: 6th Int. Conf. on Electronic commerce, 2004.

[4] Sassen, A., Macmillan, C.: The service engineering area: An overview of its current state and a vision of its future. European Commission. Network and Communication Technologies, Sof Technologies, 2005.

[5] Zoran Stojanovic , Ajantha Dahanayake, « Service-Oriented Software System Engineering: Challenges and Practices », IDEA Group, 2005, ISBN 1-59140-426-6.

[6] Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall, Englewood Cliffs, 2006.

[7] Dan, A., Ludwig, H., Pacifici, G.: Web service differentiation with service level agreements. IBM White Paper (2003)

[8] Rouvoy, R., et al.: Composing Components and Services using a Planning-based Adaptation Middleware. In: Pautasso, C., Tanter, É. (eds.) SC 2008. LNCS, vol. 4954, pp.52–67. Springer, Heidelberg, 2008.

[9] Geihs, K., et al.: A comprehensive solution for application-level adaptation. Software: Practice and Experience, 2008.

[10] Brataas, G., et al.: Scalability of Decision Models for Dynamic Product Lines. In: Int. Work. on Dynamic Software Product Line, DSPL, 2007.

[11] Floch, J., et al.: Using Architecture Models for Runtime Adaptability. IEEE Software, 2006.

[12] Lundesgaard, S.A., et al.: Construction and Execution of Adaptable Applications Using an Aspect-Oriented and Model Driven Approach. In: Indulska, J., Raymond, K. (eds.) DAIS 2007. LNCS, vol. 4531, pp. 76–89. Springer, Heidelberg (2007)

[13] Romain Rouvoy, Paolo Barone, Yun Ding, Frank Eliassen, Svein Hallsteinsen, Jorge Lorenzo, Alessandro Mamelli, and Ulrich Scholz. MUSIC: Middleware Support for Self-Adaptation in Ubiquitous and Service-Oriented Environments. Springer-LNCS 5525, pp. 164–182, 2009.

[14] Mascolo, C., Capra, L., Emmerich, W.: Mobile Computing Middleware. In: Gregori, E., Anastasi, G., Basagni, S. (eds.) NETWORKING 2002. LNCS, vol. 2497, pp. 20–58. Springer, Heidelberg, 2002.

[15] Kuropka, D., Weske, M.: Implementing a Semantic Service Provision Platform — Concepts and Experiences. Wirtschaftsinformatik Journal (1), 16–24, 2008.

[16] Moser, O., Rosenberg, F., Dustdar, S.: Non-intrusive monitoring and service adaptation for WS-BPEL. In: 17th Int. Conf. on World Wide Web (WWW). ACM, New York, 2008.

[17] Menasce, D., Dubey, V.: Utility-based QoS Brokering in Service Oriented Architectures. In: Int. Conf. on Web Services (ICWS), 2007.

[18] Capra, L., Emmerich, W., Mascolo, C.: CARISMA: Context-Aware Reflective Middleware System for Mobile Applications. IEEE Trans. on Software Engineering, 2003.

[19] Romain Rouvoy, Paolo Barone, Yun Ding, Frank Eliassen, Svein Hallsteinsen, Jorge Lorenzo, Alessandro Mamelli, and Ulrich Scholz. MUSIC: Middleware Support for Self-Adaptation in Ubiquitous and Service-Oriented Environments. Springer-LNCS 5525, pp. 164–182, 2009.

[20] G. Grondin, N. Bouraqadi, and L. Vercouter. MaDcAr: an Abstract Model for Dynamic and Automatic (Re-) Assembling of Component-Based Applications. In Proceedings of the 9th International SIGSOFT Symposium on Component-Based Software Engineering (CBSE 2006), LNCS 4063, pages 360-367, June 2006, Västerås, Sweden. Springer.

[21] Luc Fabresse, Christophe Dony, and Marianne Huchard. Foundations of a Simple and Unified Component-Oriented Language. Journal of Computer Languages, Systems & Structures, editor Elsevier, Volume 34/2-3 (July-October 2008), p. 130-149.

[22] Makhlouf Derdour, Nacira Ghoualmi-Zine, Philippe Roose, Marc Dalmau - Toward a dynamic system for the adaptation multimedia fluxes in the P2P architectures - FINA, helded in AINA-09, ISSN : 978-0-7695-3639-2/09. DOI 10.1109/WAINA.2009.28, pp 67-72.