An adaptation approach for component-based software architecture

Makhlouf Derdour, Philippe Roose, Marc Dalmau, Nacira Ghoualmi-Zine, Adel Alti

An adaptation approach for component-based software architecture

Makhlouf Derdour[1], Philippe Roose[2]
Marc Dalmau[2], Nacéra Ghoualmi Zine[1], Adel Alti[3]

[1] University of Annaba
Computing Department
Annaba – Algeria
{m.derdour, ghoualmi}@yahoo.fr

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

[3]Université of Setif
Computing Department
Setif – Algeria
altiadel2002@yahoo.fr

***Abstract*- In this paper we propose a meta-model for architectures with heterogeneous multimedia components. Currently, a generic solution does not exist to automatically deploy a distributed architecture based on multimedia components. The description of the incompatibilities between components is a need in such approaches. Indeed, software architectures validate the functional aspects, which are not sufficient to guarantee a realistic assembly. For instance, the problem of heterogeneity related to the exchanged data flows. In order to highlight these incompatibilities and to find solutions, a model-based approach called MMSA (Meta-model Multimedia Software Architecture) is proposed. It enables the description of the software architectures expressing a software system as a collection of components which handle various types and formats of data, and interacts between them via connectors including the adaptation connectors.**

Keywords- component; adaptation; concerns; multimedia; software architecture.

I. INTRODUCTION

The components-based development is an approach widely used to construct complex systems. Basically, requirements are affected to components of a certain type classes, packages, services, etc. While many requirements can be effectively assigned to individual components, there are many requirements that cannot be located to one component and that having repercussions on numerous components (configuration). The requirements express functional and non-functional concerns. The conservation of such concerns during the design and the implementation gives a system difficult to understand and maintain. It is commonly accepted that it is better to separate the functional and non-functional concerns. This facilitates the search of the business components to satisfy the functional concerns and allows the factorization of the use of the components ensuring the non functional concerns. In the MMSA approach (Meta-model for Multimedia Software Architecture), the two types of preoccupations are ensured respectively by the components and the connectors. Thus, the connectors ensure the communication and the connection of components that realize the functional part (business component). Their execution within adequate configurations also requires taking into account of the non-functional aspects.

The component-based design has two fundamental activities: the conception for the reuse and the conception by the reuse. The main objective of design for reuse is to create a complete library of reusable components while the main objective of the design by reuse is to create new products by reusing existing components. In this paper we focus on the second activity in order to construct multimedia applications (applications handling several types of media, such as: text, image, sound and video). Compared to conventional applications, these applications have many drawbacks related to the media variety (type, format and characteristic), and to their adaptations. These difficulties are increased by the pervasive character increasingly ubiquitous in such applications.

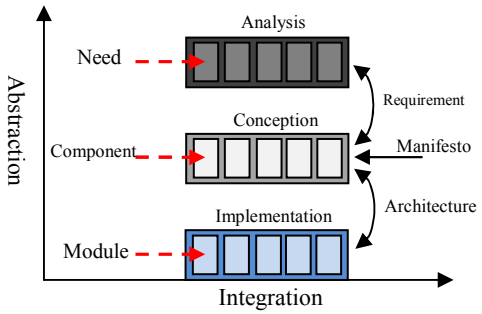

Figure 1. Levels of abstraction in software architecture

The heterogeneity of components regarding embedded sensors, CPU power, communication mechanisms (GPRS, WIFI, Bluetooth, ZigBee, etc.), speed of transmission as well as the media variety (sound, video, text and image) requires taking into account adaptation to an abstract level in order to avoid the ad hoc solutions which are not reusable and/or generalized (see figure1).

0730-3157/10 $26.00 © 2010 IEEE
DOI 10.1109/COMPSAC.2010.24

Therefore, we propose a meta-model of software architecture for multimedia applications incorporating flows properties of multimedia data. The adaptation of the data flows is deported to the connectors, here called adaptation connectors. These connectors include necessary adaptation services as well as qualitative extensions of these services in order to offer a measure reflecting the evolution of data flow after the adaptations.

After the introduction, the motivations of this work are presented in the second section. Furthermore, some related work is presented in the third section; the next section presents our meta-model of multimedia component-based software architecture. The last section details the properties and the characteristics of the meta-model components. Finally some conclusions and perspectives are given.

II. MOTIVATIONS

Our main motivation is to propose a meta-model for maintaining data consistency in configurations constituted of heterogeneous components (multimedia flow, communication protocol, etc.) using new types of graphic interfaces and connectors with a richer semantic.

The use of these graphics interfaces allows the automatic detection of points of heterogeneity between components, while the use of adaptation connectors allows the resolution of these heterogeneities. The systems are constructed by assembling (functional) components and (non-functional) connectors, where each element is correctly placed in the architecture configuration. In most of the ADL (Architecture Description Language) and the existing languages we find that:

- The choice of the available connectors in the environment is limited to the primitive connectors, no compounds connectors.
- The management of the non-functional concerns of the components is ensured after the definition of architecture and configuration of the components.
- The management of assembly does not take into account the behavioral heterogeneity (semantic) of the components of software architecture.
- Few models are able to define new connectors with different treatments that ensure the non-functional concerns of the components (security, communication, conversion, etc.).
- There is no direct and automatic correspondence between architectures (models) and the applications conceived following these architectures (instances).

In order to solve these insufficiencies, we propose MMSA to describe multimedia-components-based software architectures. Based on the definition of four types of interfaces according to the data flow (Image, Sound, Text, and Video) and a strategy of adaptation of the multimedia flows (type, format, property) to three levels, we propose a model to solve the problem of components data exchange heterogeneity. It is developed in order to reach the following objectives:

- Ensure a high level of abstraction for the connectors in order to make them more generic and more reusable, and therefore reconfigurable.
- Take into account the semantics of communication links between components in order to detect points of heterogeneity and insert the adaptation connectors in those points.
- Favor the maintenance and the management of the adaptation QoS and of the communication ensured by the connectors by providing the following possibilities: adding, suppression and substitution of adaptation services.

III. RELATED WORKS

The software components are reusable software entities promising a cost reduction in development, maintenance and in software evolution. Currently, many propositions claim the development mode based on the assembly of software components. Despite the common vocabulary (component, port, interface, service, configuration, connector), these propositions are varied regarding their origins, their objectives, and their concepts and also their mechanisms.

ADLs are used to specify the software architecture. Medvidovic [26] has presented the difference between an ADL and a formal specification to distinguish the ADL from the other modelling notations. The [16] and [11] presented an overview of the ADL. The connector types have been studied by [27]; they presented taxonomy of connectors allowing the support of the non-functional properties (communication, security, conversion, facilitation, coordination, interaction).The models cover all or part of the needs in terms of language, of semantic and of tools. In [27], the authors raise insufficiencies in the specification of non-functional properties of the systems; we notice a lack in semantic foundation for the expression of the constraints and refinement (component, connector and configuration) and a lack of tools for the dynamic reconfiguration and the evolution in real time.

The approaches like [3] [12] [24] [7] allow the separation of the functional concerns. They were proposed in order to capitalize the functional needs in modular entities. Several ideas were proposed within this perspective. We mainly distinguish two categories of approach for software architectures: those inspired on Component-based software engineering (CBSE) and those that are service oriented for service-oriented architecture (SOA). In the first case [37] [3] [12] the accent is put on the static structure of the system: the software elements are components assembled by connectors in configurations. Whereas in the second case [33] [24] [7] [6] the accent is put on the functional structure of the system: the software elements are functionalities (services) linked by relations of collaboration or combination. The model proposed in this paper could be described as hybrid as it includes components and proposed services by these components.

Modern applications which have software preponderance are more and more developed by ADL-based development processes [9]. The ADLs allow analysis and verification of properties early in the development cycle that the future system will have to satisfy, in particular the homogeneity and compatibility properties of components handling various media. Indeed, the current applications (multimedia, embedded systems, communication systems, etc.) consider the media notion as an important characteristic of their behavior [8] [10]. Most of existing ADLs such as SPT-UML [20], MARTE [30], and AADL [34] do not take into account the adaptation and the properties related to multimedia flow during the software construction phase. Some of them, treat the problem of heterogeneity by modification of the configuration parameters (addition, withdrawal or replacement of components) [22] or by a meta-model which verifies the adequacy of service regarding its context and research of the adaptation strategy [23].

A simple component language [21] proposes a comparison of the principal characteristics of the components languages: component, interface, port, service and connector. The main objective of this work is to take into consideration the unforeseen connection of the developed components in an independent way. As a solution, it proposes the production of reusable and configurable connectors through the association of a particular service to the provided ports which will be used in the absence of the requested service at port level. A drawback of this work is the absence of the integration mechanisms of the new communication services which ensures the evolution of architecture towards new needs; it also lacks techniques for checking the quality of architectures and the provided services.

C3 (Component Connector Configuration) [1] is an approach based on software architectures. It makes it possible to describe a view of logical architecture in order to automatically generate physical architecture for all the application instances. The idea is based on the refinement and the traceability of the architectural elements. The software architecture is described in accordance with the first three levels of modeling defined by the OMG [31] [32]. Consequently, to describe logical architecture, three types of connectors are defined: the connection connector (CC) used to connect components and/or configurations, the composition/decomposition connector (CDC) which represents a structural link between a configuration and its constituent (component, connector), and expansion/compression connector (ECC) which establishes a service link between a configuration and its internal elements. Each connector type has its own semantics and its own form. The physical architecture is an image in memory of application instance of the logical architecture. This image is constructed by a graph whose nodes are instances. The nodes of this graph are connected by arcs whose types correspond to specific types of connectors. The connectors proposed do not ensure the connection of the heterogeneous components and do not take into account the semantics of configurations and that of the links between components.

The ADL can be classified in three different categories [1]: ADL without connectors, ADL with a preset set of connectors, and ADL with explicit types of connectors. In the last case, the ADL provides connectors as first order elements of the language such as: Wright [4], [25], ACME C2 [19], xADL [17], AADL [5], etc. All these languages seek to improve the reusability of the components and the connectors by separating the calculation and the coordination. In our approach, we choose the explicit category of connector. Thus, in MMSA meta-model, we present a generic and explicit type of connector that the system can specialize it according to the architecture and the components needs. We will detail this concept in section 5.2.

IV. THE METAMODEL MMSA

The development of multimedia applications requires two complementary models: a multimedia data flow model allowing the representation of various types of media exchanged between components and their relationships, and an architecture model based on the concepts of ADLs extended to multimedia and integrating adaptation connectors. The main idea of this proposal is to take into consideration the standard concepts of multimedia data as well as the nonfunctional concerns (data adaptation, communication protocol, security, etc.) of the components by connectors at the software architecture level. The objective is to propose a generic, clear and complete description. In the following parts we present different concepts represented by models. For each model we detail the relations between its concepts.

A. Data flow model

In pervasive environments (mostly heterogeneous and mobile), the devices can require for any contents type, going from textual contents to the complex and rich multimedia documents. Ensuring the delivery of the adapted data to each peripheral requires adaptation techniques which take into consideration the media and the flows structuring. Therefore, their modeling is necessary. It facilitates the adaptation work between media of the same type (image to image for example) or between different media types (text to sound for example).

The hierarchic structure of media is expressed in UML using a class diagram. The media are classified in two categories: continuous media, such as the video or the sound, which are characterized by temporal dependences and the discrete media such as the image or the text. Each type of media has a set of encoding formats and some specific properties like the resolution (in the case of image or video), the frequency (in the case of the sound), etc. we distinguish three types of structural links between media: temporal (to describe the temporal dependences between units), logic (to describe the logical organization of a flow in hierarchy form

of media) and spatial (to describe the disposition of the multimedia-flow elements).

Currently, the multimedia data flows must be executed on many platforms (smartphones, PDA, Laptop or Desktop PC, etc). This diversification of the uses and the supports requires the adaptation of flows to their execution context, which are sometimes unforeseeable at the time of preparation and design of data.

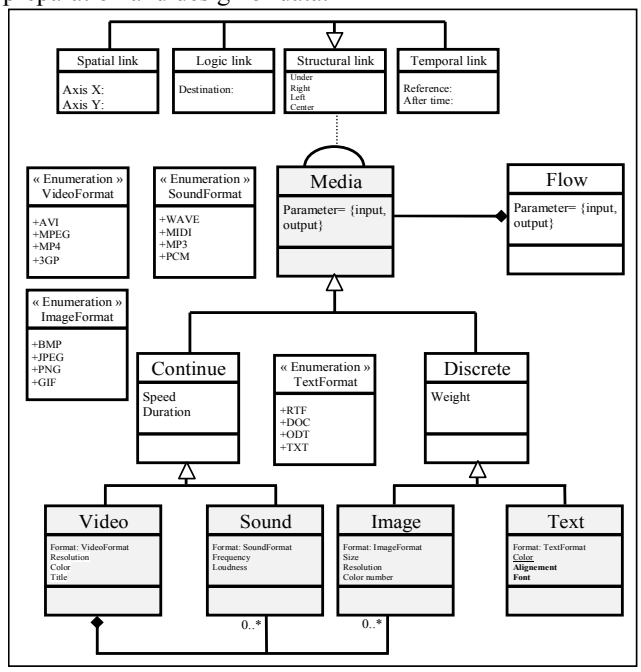

Figure 2. Multimedia flow model for MMSA

B. *Adaptation of data flow*

Each media can undergo three types of adaptation. The first one is known as the format conversion (Transcoding). It allows conversion in the same type according to a different encoding format (BMP to.JPEG for example). The second one allows a handling of the media characteristics (eg. modification of image resolution for example). This type of adaptation (transforming) depends on the media format, since each format authorizes the change of some characteristics in the form of parameters. The third and more complex transformation is called conversion of types (Transmoding). It allows passing from a media type towards another (text to sound for blind people for example). This conversion of the type can also act on media structures by removing the temporal dependences (for example the video to the images). Each adaptation has an impact on the data quality. Thus, the conversion of an image from a JPEG format towards a GIF one implies a reduction in the number of colors to 256, the reverse implies the suppression of component "transparency", which according to the use context can be problematic, even crippling.

The adaptation is a process (cf. figure3) allowing a modification the type of media (transmoding), the format of encoding (transcoding) and/or the media content (transforming) in order to adapt it to the component recipient. The class diagram of figure3 shows the various classes of association allowing the passage of a media type to another, or of a media format to another format.

The following table (*Table 1*) presents taxonomy of possible adaptations between media:

TABLE I. ADAPTATIONS OF MEDIA

Category	VIDEO	SOUND
Transcoding	Format conversion	Format conversion
transforming	-frame rate reduction -spatial resolution reduction -temporal resolution reduction -color depth reduction	Change sampling
Transmoding	-video to image -video to text -video to audio	Audio to text

Category	Text	Image
Transcoding	Format conversion	Format conversion
transforming	-font size reduction - change of police, color, etc.	-data size reduction -dimension reduction -color depth reduction -color to grayscale
Transmoding	Text to Audio Text to Image	Image to Text

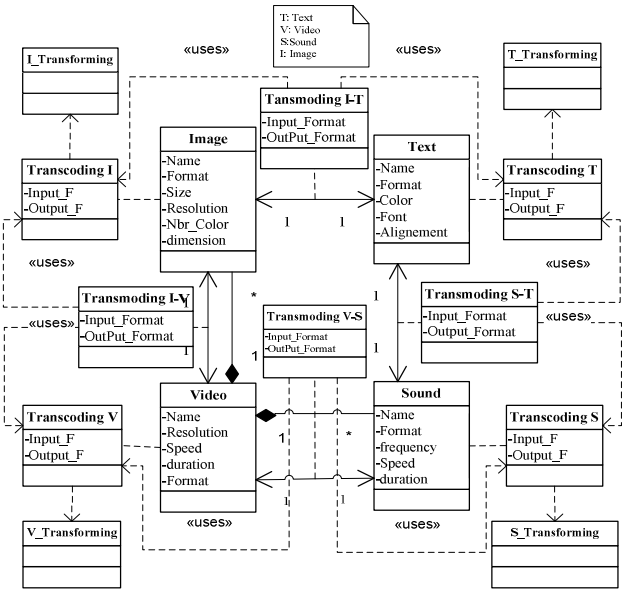

Figure 3. The transformation relationship between different media

The relation between association classes of transmoding with the association class of transcoding explains that the transcoding class can be called upon by the transmoding class to participate in achieving the task of the latter.

Although the relation between transcoding class and the transforming is a relationship of dependence, this relationship explains that each format has a set of parameters to manage the various qualities of media. The transforming is a particular type of transcoding which keeps the same format of media with changes of characteristics (for example: conversion of a color JPEG picture to a black and white one).

C. *The adaptation in MMSA*

During the process of architecture creation, in order to solve the heterogeneity problem of architectural elements (component, connector and configuration), the adaptation is made in three successive stages: (I) adaptation of the types (II) adaptation of the formats (III) adaptation of the properties.

The data flow is a main constituent of the functional components, it is often specified as a constraint to associate with a functionality of communication involving several components.

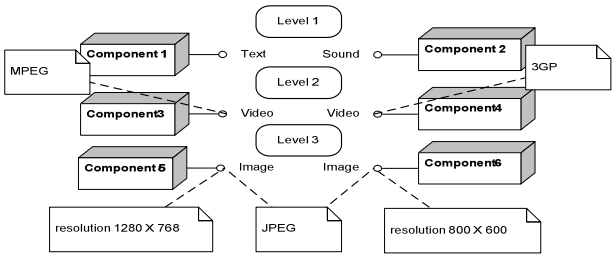

Figure 4. Heterogeneity between components

The constraints of data flows such as the type, the format and the media parameters must be specified at the architectural level. For that, we consider a new type of component intended to ensure a non-functional concern that of the adaptation, which one calls the adaptation connector related to the component which provides and/or requires the data multimedia. We propose a graphical notation of the ports of multimedia interfaces allowing to visually identify the heterogeneity points per media type and to highlight the need for the search of adaptation connectors.

TABLE II. PORT OF MULTIMEDIA INTERFACE

Type	input Output	Format		
Text		DOC	DOCX	ODT
Image		JPEG	BMP	PNG
Sound		WAVE	RM	MP3
Video		MPEG	AVI	MP4

The detection of heterogeneity is done automatically by the checking of the constraints of forms and colors.

- Adaptation of type

The heterogeneity of components that manipulate the media of different types is detected by the use of different forms to represent the components ports (level 1, figure 4). Therefore, two components which have different ports (example: text port and sound port) can be connected only by the use of one or several adaptation connectors of media type. This problem will be solved by the integration of the transmoding connectors at the architectural level.

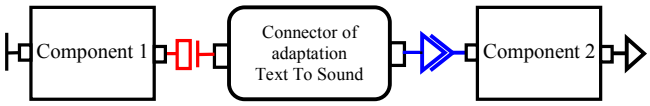

Figure 5. Transmoding connector of text toward sound

- Adaptation of format

The heterogeneity of the components that manipulate the same type of media but with two different encoding format (level 2, figure 4) can be detected by the presence of color differences between the formats of the same type. Therefore, two components which have different colors for the same port (example: red port for MPEG video and blue port for 3GP video) can be connected only with the use of one or several connectors of format adaptation. This problem will be solved by the integration of the connectors of transcoding at the architectural level.

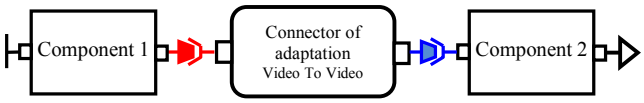

Figure 6. Transcoding connector of MPEG to 3GP

- Adaptation of media properties

The heterogeneity of components that manipulate the same media type with the same format (level 3, Figure 4) but with different properties (example: resolution and color for image, sampling and speed for video, etc.) cannot be expressed visually in our architecture, due to the parameters that depend on the media and on the adaptation service (parameters of the service). Therefore, two components which have the same color for the same port (example: image port) can be connected with a simple communication connector, and during the execution, the adaptation manager and the QoS manager both manage together the adaptation if necessary. At this level the problem of heterogeneity is resolved at runtime, by the manipulation of the parameters of the adaptation service; if this service is configurable; regarding the parameters of flow.

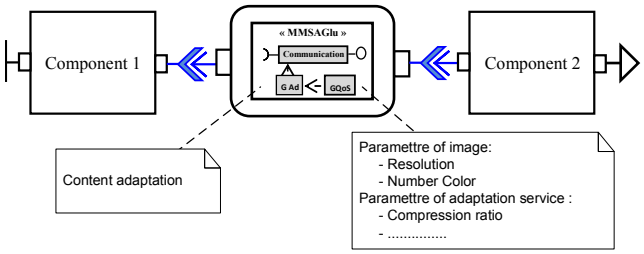

Figure 7. Adaptation Connector of image content

The adaptation service is configured, in order to allow an adaptation in different situations; it is applied in several contexts, for example, adaptation of the resolution of an image.

D. A meta-model of multimedia software architecture

MMSA meta-model describes the software architecture of the system as a collection of components interacting with connectors. Components and connectors have the same abstract level and are defined explicitly by the separation of their interfaces and their internal configurations.

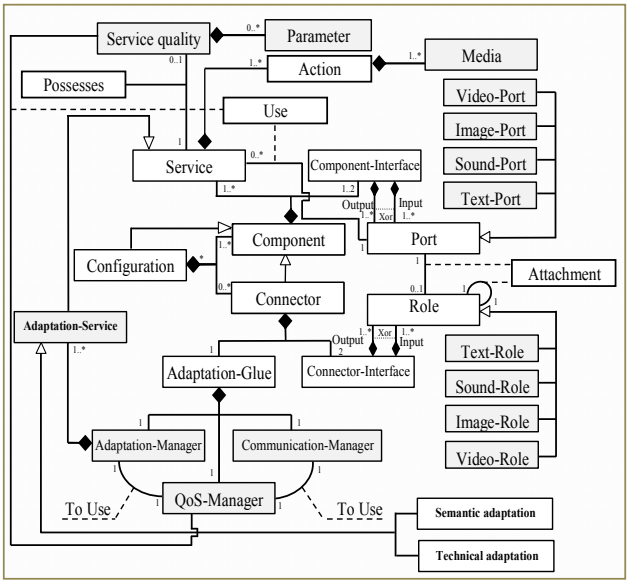

Figure 8. Class diagram of software architecture MMSA

The basic concepts of MMSA software architecture are the same as in most software architectures: configuration, component and connector. The software architecture model of MMSA is a hybrid model based on the concepts of component-based software engineering (CBSE) and service-oriented architecture (SOA).

A component is defined by a set of services that interact to fill a role of component and communicate with environment through its required/provided interface. Generally, the connectors define abstractions which encapsulate the mechanisms of: communication, coordination and conversion (type, number, frequency and order of interactions) between the components. A connector is represented by an interface and glue [18] [36]. This description considers the connector as a mediator between components, which limits its role in communication. The specification of glue describes the functionality expected from a connector. It represents the hidden part of a connector. The glue can be a simple protocol of communication linking the ports, or a complex protocol that uses various operations especially that of: links, conversion of data format, transfer, adaptation, etc. Generally, the connector glue is the connection type of this connector.

In MMSA, a connector is a set of services (communication, adaptation, QoS, etc.) ensuring connection between the components. It can ensure the nonfunctional concerns of components (such as security, data transformation, communication, etc). This allows a possible change of the adaptation services during the execution of the application (dynamic and real time adaptation), and preserves the abstract specification of the component.

A component is a computation unit having a state and a unit of implementation (business part). It can be simple or composite. The Components in MMSA are abstractions which encapsulate services and handle media in several formats through the interfaces. There are two types of interfaces, an "Output" interface exporting the data of the components, and the "Input" one importing the data to the components. The interface describes the interactions of the component, including the connection points (ports). We distinguish one type of port by type of media identified (*cf* the previous Tab.II). Each one provides/requires media of the corresponding type. This distinction of the ports by data type (sound, image, video and text) can simulate the behavior of a component at runtime at the design phase, in order to detect the heterogeneity points between components and to treat them at this level. This gives a better verification of the consistency and validity of the configurations of software architectures.

A MMSA connector is defined by two interfaces "Input" and "Output" and a glue unit represented by three managers: communication, adaptation and QoS. They manage the data transfer between components and allow adaptations to be made. A required/provided interface of connector is composed of a set of roles. Each role serves as a point through which the connector is connected to the component. This distinction regarding the components is expressed by the fact that two components can be only linked by connectors, while two connectors can be directly connected.

- *Example 1&2:*

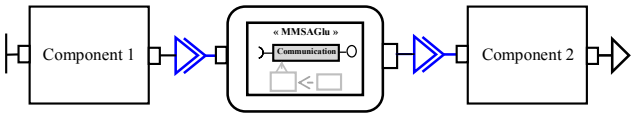

Figure 9. Communication connector

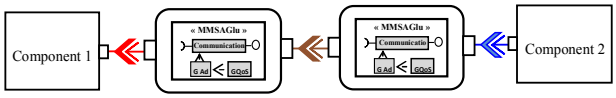

Figure 10. Tow connectors of image transcoding JPEG to BMP to PNG

In the first example, the adaptation manager and the QoS manager are deactivated (gray); this describes a minimum communication connector for connecting two components. While the second example shows the possibility to connect two heterogeneous components by two (or more) connectors depending on the complexity of adaptation.

In the communication roles, we added types to clarify the connections between various components regarding the data flow. On the intern level, we have enriched the glue by an adaptation manager which cooperates with a quality service manager to ensure the task of adaptation. An adaptation manager is a set of adaptation services that cooperate to realize adaptation. Two types of adaptation can be realized in software architectures, the semantics adaptation (conversion of type) related to the constraints of the data handled by components, the technical adaptation (conversion of format and adjustment of media characteristics) related to the capacity of components (memory, display, etc.). The QoS manager controls the adaptation manager in its work in order to change the parameters of adaptation services to provide adequate quality to component needs at runtime. The QoS manager participates in selecting parameters of technical adaptation services of data flows (e.g. reduction of resolution, reducing the number of images per second) and even the adaptation services of type or format at runtime (e.g. choice of compression ratio in the transformation from BMP to JPEG).

A configuration is an interconnection of components and connectors through interfaces. The constraints are necessary to describe the dependencies between components and connectors within a configuration.

The objective of the configuration is to abstract the details of various components and connectors (encapsulation of the components ensured by restricting access through the interfaces). A configuration has a name and can have an interface (represented by component interfaces which require/provide the flows from/to the external environment) and a set of services (encapsulated into components). The configuration is defined by components, attachments and connectors allowing the interactions between components. The attachment is a communication link between a port of a component and a role of a connector (an output port must only be linked with an "Input" type role and reciproquely).

A connector is a component, which explains the possibility of connecting connectors between them. In our configuration two components can be connected by one or more connectors, i.e. a component needs at least one connector to communicate with another one (*cf* fig 9). It can use several connectors depending on the complexity of the adaption task (*cf* fig 10).

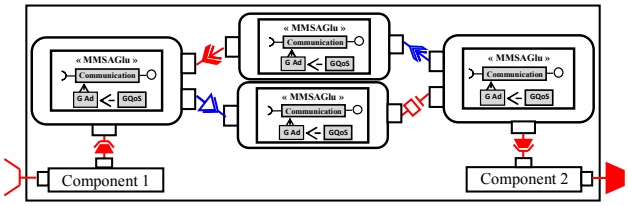

Figure 11. A configuration with multiple connections working in parallel and in sequence

Figure 11 describes an adaptation that involves several connectors; it is an adaptation of a video (sound and image) to a video (image and text).

V. ELEMENTS OF MMSA ARCHITECTURE

For graphical representation of components, we selected the "Osagaia" model (Bouix et al, 2005). Indeed, the Osagaia container integrates specific units in order to manage the data flows and the associated buffers. This container offers a certain number of supervision commands allowing to connect/disconnect/move/relace it. Moreover, it offers a set of information on its execution which allows possible reconfigurations to be decided. In its current version, this container has several implementations enabling it to be used on more or less constrained peripherals (PC, PDA, SunSpot SmartSensors – soon available on Sun Microsystems' forge). This set of characteristics makes it an interesting candidate for our work. Nevertheless, the Osagaia model does not propose typing of the inputs/outputs ports to represent the various media, a point that it is imperatively necessary to solve within the framework of our work.

A. Component

It is well known that a component "can be accessible only via well defined interfaces" [38]. The Interfaces represent the link of components with the environment. Component-based languages propose different concepts to describe the interfaces of elements such as the services, the ports, the interfaces, the protocols, etc. with sometimes different means. For example, in Fractal [14] or Enterprise JavaBeans [28], the port and the interface concept are mixed; this is the reason why we only speak about the interfaces. In UML components [15], the two concepts of ports and interface exist, as in ArchJava [2] where interfaces are called port of interfaces. Therefore we have chosen to clearly explain the choices we have made in MMSA. In MMSA, a component provides or requires services by the

ports described in the interface provided/required. Thus, MMSA proposes a typing of ports to differentiate them according to the media type handled (Text, Sound, Video, and Image).

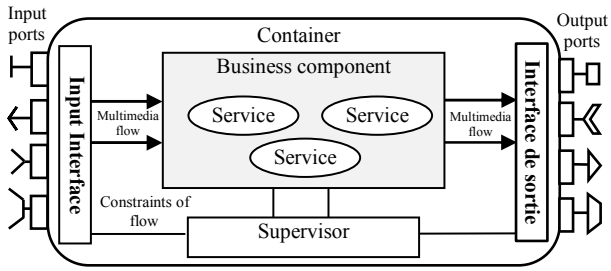

Figure 12. Model of multimedia component

A component provides functionalities are named services. Basically, a service is a subroutine defined in an element, like a method in the object model. A component has two categories of services: provided and required services.

"A component is a static abstraction with plug-in" [29]. The ports represent this plug-in which is the point of component interaction. This means that everything passes through these ports, like the services invocation for example. The port is present in almost all the component models, but with different semantics. In the models of components where the ports exist, they are unidirectional or bidirectional. By the unidirectional ports, as in ComponentJ [35] or Fractal [14], a component provides or requires all services via its ports. In ArchJava [2] or UML 2.0 [15], the ports are bidirectional and a component requires and provides services through each of its ports. In MMSA the ports are unidirectional, because a port can provide/require data via/from connectors. The latter can apply an adaptation to the data and generally adaptation services are not bidirectional (for example: the adaptation service of the text towards the sound is not the same service as that of the sound adaptation towards the text).

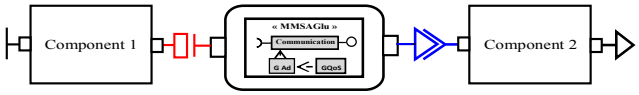

Figure 13. Transformation connector text into audio

B. Connector

Compared with those of the languages of description of architectures [3] [27], the connectors that we propose can be simple or composite and can ensure services. These connectors do not only ensure the communications links but also the adaptation of the data exchanged (functional part of connectors) between components.

The connector constitutes the entity of communication and adaptation in our approach, i.e. it is able to transfer the multimedia data between the various components while ensuring the adaptation of the latter.

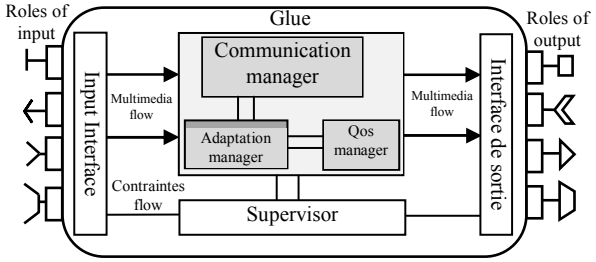

Figure 14. Model of multimedia connector

Allowing heterogeneous components to interact with each other is a significant task. The adaptation is considered as a nonfunctional concern of component, this task must be ensured by another element. The connector provides the nonfunctional concerns (communication, adaptation, security, etc.) which the component needs. The role of an adaptation connector is to receive the data, to adapt them according to the QoS manager directives and to forward them the following component or to connector.

The supervisors' role is to preserve the constraints of the flow received by the input interface and emitted by the output interface. He is also in charge of supervision of the connector to be able to ask for its reconfiguration with replacement of the adaptation services in the case where the QoS manager is not able to provide the required quality.

VI. CONCLUSION

In this paper a generic meta-model for the description of software architectures is presented. This meta-model integrates the multimedia concepts and QoS. This enabled us to present in a separate way the flow parameters and media which present a very important aspect of component configurations and assemblies. The contribution of this work is situated in a context of description per level of abstraction, integrating in a separate way the functional and nonfunctional concerns of the components. This ensures a quality of the components assembly by inserting the adaptation connectors, as well as management of adaptation service quality. The main advantages of MMSA are the consideration of the multimedia aspect and the separation between the functional and nonfunctional concerns of the components.

Our proposition can be used as a support to develop the management applications of the numerical resources (DAM: Digital asset management). This type of application handles a large variety of media, and communicates with the users through various platforms (Cellphones, PDA, PC, portables, etc). MMSA can bring an effective solution to the development of DAM, especially in the following: the acquisition, the treatment, the distribution and the use of multimedia contents. It offers the possibility of taking into consideration the factors generating the incompatibilities between components in the DAM architecture. It gives a solution at the architectural level by injecting the adaptation connectors and at the execution level by the management of QoS and the reconfiguration of these connectors.

As a perspective we propose to develop a modeling tool for our approach and to investigate other nonfunctional concerns. The development of the service quality aspect must be also taken into account.

VII. REFERENCES

[1] Amirat A and Oussalah M. First-Class Connectors to Support Systematic Construction of Hierarchical Software Architecture. In Journal of Object Technology, vol.8, no.7, 2009, p. 107-130.

[2] Aldrich J, Chambers C, Notkin D. ArchJava: Connecting software architecture to implementation. In: ICSE. ACM; 2002. p. 187–97.

[3] Allen R, GARLAN D. A Formal Basis for Architectural Connection. In ACM Transactions on Software Engineering and Methodology, vol. 6, no 3, 1997, p. 213–249.

[4] Allen R.J., A Formal Approach to Software Architecture, Phd Thesis, School of CompScien, Carnegie Mellon University, 1997.

[5] Allen R., Vestal S, Lewis B, Cornhill D. Using an architecture description language for quantitative analysis of real-time systems. In Proceedings of the Third International Workshop on Software and Performance, ACM Press, Rome, Italy, 2002, p. 203–210.

[6] El Asri B, Kenzi A, Nassar M, Kriouile A. Towards an MVSOA architecture for the implementation of multiview components. 3sd francophone Conference in software architectures, (2009) pp 1-17.

[7] Attiogbé C, André P, and Messabihi M. Correction d'assemblages de composants impliquant des interfaces paramétrées. 3sd francophone Conference in software architectures. Hermès, 3e francophone Conference in software architectures. 2009, Lavoisier.

[8] Avizienis A, Laprie J-C, Randell B, Landwehr C. « Basic Concepts and Taxonomy of Dependable and Secure Computing ». IEEE Transactions on Dependable and Secure Computing. 2004. pp. 11-33.

[9] Avgeriou P. Uwe Zdun. Modeling Architecture Patterns using Architecture Primitives. OOPSLA' 05, ACM (October 2005).

[10] Balsamo S, Bernado M, Simeoni M. Performance Evaluation at the Architecture Level Formal Methods for Software Architectures. LNCS 2804. Springer. Berlin, Germany. 2003, p. 207-258.

[11] Barais O. Build and control the evolution of components-based software architecture - PhD thesis, LIFL, Lille University , 2005.

[12] Bergner K, Rausch A, Sihling M, Vilbig A, Broy M. A Formal Model for Component ware. Foundations of Component-Based Systems, Cambridge University Press, New York, 2000, p. 189–210,.

[13] Bouix E, Dalmau M, Roose P, Luthon F. A Multimedia Oriented Component Model - AINA 2005 - Tamkang University, Taiwan.

[14] Bruneton E, Coupaye T, Leclercq M, Quéma V, Stefani J-B. An open component model and its support in Java. LNCS, vol. 3054. Berlin: Springer; 2004. p. 7–22.

[15] Cheesman J, Daniels J. UML components: a simple process for specifying component-based software. USA: Addison-Wesley, 2000.

[16] Clements P. C (1996). A Survey of Architecture Description Languages. IWSSD '96: USA, IEEE CS, page 16.

[17] Dashofy, E., Hoek, A.v.d., Taylor, R.N., "A comprehensive approach for the development of XML-based software architecture description languages", TOSEM, 2005,volume 14, issue 2, p. 199–245,

[18] Goulao. M, F.B. Abreu, Bridging the gap between ACME and UML 2.0 for CBS. In: Proceedings Workshop of Specification and Verification of Component-Based Systems, Helsinki, Finland, 2003.

[19] Garlan D, Monroe R-T, and Wile D. Acme: Architectural Description Component-Based Systems, Foundations of Component-Based Systems. Cambridge University Press , 2000, p. 47-68.

[20] Graf S. , Ober I. « How useful is the UML realtime profile SPT without semantics? » In SIVOES and RTAS 2004, Toronto Canada.

[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, 2008, p. 130-149.

[22] Marcel C, Michel R, Christian M, Calin L, Costin M. Dynamic adaptation of services. DECOR'04, Grenoble, France. 2004.

[23] Marcel C, Michel R, Christian M. Autonomic Adaptation based on Service-Context Adequacy Determination. In ENTCS, vol. 189, jul 2007, p. 35-50, Elsevier.

[24] Maximilien. E and Singh. M. Self-adjusting trust and selection for web services. June 2005, p. 385–386.

[25] Medvidovic N, Rosenblum D-S, and Taylor R-N. A Language and Environment for Architecture-Based Software Development and Evolution, ICSE'99, Los Angeles, May 1999.

[26] Medvidovic N, Taylor R. N. A Classification and Comparison Framework for Software Architecture Description Languages - IEEE Transactions on Software Engineering, 2000, vol. 26, no 1, p. 70–93.

[27] Mehta N R, Medvidovic N, Phadke S, Towards a taxonomy of software connectors », ICSE '00, ACM Press, 2000, p. 178–187

[28] Monson-Haefel R. Enterprise JavaBeans. Sebastopol, CA, USA: O'Reilly & Associates Inc.; 1999.

[29] Nierstrasz O, Dami L. Component-oriented software technology. In: Object-oriented software composition. Prentice-Hall, 1995. p.3–28.

[30] Object Management Group. « UMLTM Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE) » voted at OMG. http://www.omg.org/cgi-bin/doc?realtime/2005-02-06.

[31] OMG: "Unified Modeling Superstructure" from http://www.omg.org/docs/ptc/06-04-02.pdf, 2006.

[32] OMG: "Unified Modeling Language: Infrastructure" from http://www.omg.org/docs/ formal/07-02-06.pdf, 2007.

[33] Papazoglou M-P. Service-Oriented Computing: Concepts, Characteristics and Directions. WISE, IEEE CS, 2003, p. 3–12.

[34] Society of Automotive Engineers « Architecture Analysis & Design Language (AADL) ». SAE Standards no AS5506, November 2008.

[35] Seco JC, Caires L. A basic model of typed components. Lecture Notes in Computer Science 2000;1850:108–29.

[36] Sylvain Maillard, Adel Smeda, Mourad Oussalah: COSA: An Architectural Description Meta-Model. ICSOFT, 2007, p. 445-448

[37] Szyperski C. Component Software: Beyond Object-Oriented Programming. AddisonWesley Publishing Company, 1997.

[38] Szyperski C. Component software: beyond object-oriented programming. MA: Addison-Wesley; 2002.