A Distributed Architecture for Cooperative and Adaptative Multimedia Applications

Philippe Roose, Marc Dalmau, Franck Luthon
LIUPPA, IUT Bayonne
Château-Neuf, Place Paul Bert
64100 Bayonne, France
{dalmau|roose|luthon@iutbayonne.univ-pau.fr}

Abstract

Previously, we developed a method and a distributed platform for the re-engineering of applications by adding cooperation. The goal was to supply a way of communication based on the exchange of events, messages and shared data. Now, we propose to adapt this approach (method and platform) to distributed multimedia applications. These applications present the characteristic to be organized around the communication. So, we can consider them as composed of distributed components which have to collaborate. Because it is heavily interactive, such an application needs to adapt itself in real-time to the user and to the environment in which it runs. Our approach consists in breaking down the application into two levels : the first one reflects the users' point of view in terms of functionalities. The second one, reflects the way to achieve these functionalities in terms of quality of service. We propose to organize components into workgroups and subworkgroups corresponding to this two levels. The platform manages these groups and makes them evolve in real time. It also ensures the inter-operability of components which cooperate in these groups.

1 Introduction

Our previous works were about a method for re-engineering of applications to provide cooperation. This method considers applications from a communication between modules point of view. It allows a workgroup organization of these modules and a specification of information exchanges between modules and workgroups. It is based on a platform which manages workgroups, circulation of information and creation of information with a higher semantic level. Actually distributed multimedia applications are generally realized with distributed components which are comparable to modules. With an organizational point of view, they use the notion of workgroup of components made up to realize a common task using communication. So, these workgroups of components may be considered as « super-components ».

The critical point in such applications is the need of a high flexibility to provide a certain Quality of Service (QoS). As they are very interactive, they have to adapt themselves quickly to the user's requests. Moreover, as they are distributed, they have imperatively to manage the QoS available at each moment on the network.

We recommend to structure such applications with dynamic subworkgroups of components and dynamic groups (composed of subworkgroups). A subworkgroup is a set of components constituted to achieve a common task respectively to a certain QoS (environment QoS). A group is a set of subworkgroups constituted to realize a service respectively to a user (user QoS). Such groups are managed by rules included into the platform and allow the dynamic reconfiguration of the application according to QoS criteria evaluated by these rules. The circulating of information between and inside dynamic workgroups is done to respect the set of constraints (users, services, throughput, time, etc.). To finish, the platform has a global view of the application (fitting and exchanges). It does not have an internal view of components (their functioning).

So, the platform can have a role of supervision, particularly detecting some critical situations and trying to solve them. Such interventions can be realized by modifying the constitution of workgroups and subworkgroups and creating some information in the communication scheme.

2 Previous works : ELKAR

ELKAR [8] proposed both a method for the re-engineering of applications and a software solution to implement it. We were interested in providing cooperation into existing applications to improve them. Generally, users of these applications had, with the time, included external solutions as file transfer, mail or even sometimes re-input of information.

Our proposition handles the re-definition of the application using existing modules and provides a practical implementation. So, this re-engineering method is entirely based on the structuring of the application with dynamic workgroups and on the emphasis of elements of cooperation (information that needs to circulate). It allows to define rules used by the platform to automatically manage the newly organized application.

2.1 The method

The method, entirely based on communication, has to manage elements of cooperation which will be exchanged between modules and workgroups. The first re-engineering task consists in doing an inventory of existing modules and information they will produce or use. In fact, this is a kind of audit. Next, we need to organize these modules into workgroups reflecting an organizational view of the application and managing functional constraints. Each workgroup can be now considered as an actor of the application and we need to bring to the fore information it needs and information it produces. The next step consists in establishing workgroup management rules. It is necessary to be sure that incompatible components may not be included into workgroups and that incompatible workgroups are not created. So, it is important to respect conditions before including/removing an element of a workgroup to avoid inconsistency. At this moment, we have the totality of elements constituting the application : actors (modules and workgroups) and elements of cooperation (events, messages and data). Nevertheless, some elements of cooperation identified as necessary might not be directly available (produced by a module). So, we need to define operators allowing to create them from existing elements. We use specific rules (detective rules [11]) which role is to capture some information available in order to create those missing in the application. And to finish with this study, we can write the set of rules which may be used by the platform. There are rules to do the circulating of information (with an eventual formatting aspect), rules of supervision to manage dynamic workgroups and those allowing to constitute elements of cooperation not immediately available.

2.2 The ELKAR platform

Each site (physical localization) participating to the application implements a platform allowing to provide cooperation. This platform is made of :

- *A Communication Manager (CM)* : it receives all the information sent to this site and ensures the emission on the network of elements of cooperation to other sites. Its knows the constitution of workgroups to send information to local but also distant members.
- *An Element of Cooperation Manager (ECM)* : it receives all the information locally produced and also those received by the CM. Its role is to identify information, to label it according to its origin and its destination (qualification) and to communicate it to other corresponding managers.
- *A Module Manager (MM) :* Each module of the application is associated with a MM. The MM implements the interface between the module and the platform. It captures information produced by the module, it labels, formats and communicates it to the system. Reciprocally, it receives, formats and transmits information to the module. The existence of the MM is justified because modules have not been designed to work cooperatively with others, but only to exchange information with their immediate environment.
- *A Rule Manager (RM)* : Rules are implemented as independent parallel processes. These rules use and produce elements of cooperation. The role of the RM is to activate rule processes and to take rendez-vous with them when it gets an information for these rules.

This approach of supervision of the cooperation seems to be adapted to other domains than re-engineering. That is what we will try to explain in the following parts.

3 Architecture for adaptive multimedia applications

Distributed Multimedia Applications (DMA) use the concept of component particularly when tools as Java/RMI or CORBA are used. We will be interested in these two technologies according to two aspects : communication between components and quality of service.

3.1 Cooperation and multimedia

There are four kinds of communications established between components :

- *Events* : they allow components to inform their environment of all significant internal changes. When received by other components, these events allow to synchronize their running with those of their partners.
- *Messages* : they can be seen under two main aspects. Firstly, we think about messages as electronic mail allowing to exchange asynchronous information. They are used as a tool for calling remote procedures because a remote procedure call (RPC) is just a message including the name of the procedure and its parameters. At last, with acknowledgement of messages, it is possible to provide a mechanism of

« rendez-vous » or to get back the result of a remote procedure.

- *Data* : Data constituting the information system of the application are distributed to avoid important transfer of information from site to site. Nevertheless, they are remotely available for components.
- *Flows* : It is important to keep in mind that multimedia applications are characterized by their continuous data flow (video, sound) with the highest rate possible. A multimedia flow can be characterized [9] by a succession of procedure invocations decomposed into events corresponding to the transmission/reception of the invocation. So, a multimedia flow is seen as a succession of events.

3.2 Interactivity and Quality of Service

Multimedia applications are naturally very interactive. They have to adapt themselves, in real-time, to the needs of users. Each of them wants to get a personalized view of the application where it can find the information he wants, and only that one. This information has also to be adapted to users (different languages, different tools and hardware, etc.). It is mandatory to provide this QoS without modifying profoundly the application, and moreover, without modifying the behavior of running components.
For the moment, it is not possible to have a constant QoS over Internet, especially as regards throughput and delays. Applications have to adapt themselves in real-time to these evolving conditions to offer the best possible QoS at each moment.

The proposed solution to address the problem of QoS (user and network aspects) is the use of workgroups and subworkgroups. Indeed, components of a multimedia application may be organized in workgroups and subworkgroups according to the following characteristics :

- *Component* : object or program that performs a specific function and is designed in such a way to easily operate with other components and applications.
- *Subworkgroup* : It is made of cooperating components to achieve a common task. It can be considered as a « super-component » characterized by its role and by the QoS it offers. To adapt to external constraints, we have to incorporate into these subworkgroups the components corresponding to the situation. This allows particularly to choose, for a same role, a weaker but quicker QoS when, for example, the performance of the network does not permit to have the best quality.
- *Workgroup* : they are characterized by the service they provide to the user. Consequently, a workgroup is associated to each user. Its goal is to provide the needed view of the application for the user. The flexibility of the application results from the inclusion/exclusion of subworkgroups in the group.

3.3 Supervision

Because components are not able to manage themselves problems of inter-operability (without a strong update) and global constraints of the application, we use rules. They are used by the platform to evaluate and adapt the running components to the environment. These rules adapted from the ECA model [11] – Event-Condition-Action – are fully managed by the platform. There are two types of rule :

- *Interoperability rules* which manage the flow between components that are not necessarily dedicated to work together.
- *Supervision rules* whose role is to (re-)organize the composition of workgroups and subworkgroups.

All the platform managers and rules are implemented with JAVA as parallel processes and coexist with rules processes. The existence of such a platform with a knowledge of the application (structure of workgroups, information exchanged) allows to completely separate the functional aspect from the organizational aspect.

4 Presentation of an example : an « unsupervised » videoconference

To illustrate this, we will see a proposition for a multimedia application like videoconference with automatic speaker detection and tracking using signal processing technology [5].

4.1 Application organization

The workgroup level will ensure the service layer. If a subworkgroup is included into a group, a new service is ensured. In our application, we have three kind of workgroups.

- *Speakers* : Each speaker has functionalities to capture and restitute pictures and sound and to realize the detection and the tracking. This allows to track him as he speaks and to detect when there is a speaker change.
- *Listeners* : The workgroup of components associated to each listener has to contain subworkgroups allowing to restitute pictures, sound and documents produced by the speaker. However, if we consider the problem of speech, it is evident that each user wants to hear the speaker with a language that he can understand. So, its associated workgroup may

contain a subworkgroup of components ensuring the direct transmission from the microphone of the speaker or a subworkgroup realizing a simultaneous translation or subtitles.

- *Translators* : When the speaker speaks in French, a translator provides a simultaneous English translation (the hypothesis is that all members understand English).

Each participant to the conference will be represented by an instance of one of these workgroups.

4.2 Workgroup organization

Workgroups are constituted of one or several subworkgroups. They represent services provided by the groups. For example, the subworkgroups that we can include into the group *Listeners* are :

- Picture subworkgroup,
- Original Sound subworkgroup,
- English translated sound subworkgroup
- Subtitle subworkgroup

We can see that the addition or the substitution [1] of a subworkgroup by another into the workgroup associated to a user may adapt the service to the needs of this user. In particular, we pay attention to always transmit according to the listener language. This is achieved by integrating either the "original sound" suworkgroup either the "translated sound" subworkgroup or by using the "subtitles" suworkgroup. Such a realization supposes that we have a way to know requirements of each user (for example here, languages he can understand). We propose a solution based on XML. Such information will be available as XML documents. Each user can fill them. The platform will refer to these documents to constitute workgroups.

Inside subworkgroups, the QoS will be managed by the dynamic (inclusion/exclusion/exchange) of components. Components and number of components constituting a subworkgroup will vary according to external constraints and particularly with those of the network. Thus, if during a laps of time, the available throughput does not allow the direct transmission of a high quality video of the speaker to other sites, it will be possible for these sites to replace components of the picture subworkgroup doing the transmission of video by components doing the same work but with a lower resolution, or to add components which role is to better compress pictures but with a loss of quality. To implement such a flexibility, we of course need information on components (what QoS it provides) but we also need a constant evaluation of conditions of the running environment. We propose that the platform adapts the subworkgroup composition according to measures done on the application performances and according to the characteristics of components available into XML documents.

4.3 Communication interactions

The organization of the application is dynamically modified by the platform [13]. So, even if multimedia oriented components are naturally designed to communicate, they cannot organize communications themselves. For example, a component C_1 conversing with another component C_2 may, instantly, and without being informed see C_3 substituted to C_2 according to a specific demand of the user or to suit with the possible QoS. So, how can the dialog continue without any problem ?

As the platform manage the supervision, components do not have to manage de dynamic of workgroups and subworkgroups. With such an organization, the design of components is fully independent of their use and therefore improve their reuse. Subworkgroups of components we have constituted have to appear like super-components. They have to exchange information between each other in order to collaborate. These elements of cooperation constitute information of a different level than those available on each component of the subworkgroup. They do not reflect the state of an internal component but the state of the whole subworkgroup. It is evident that such information cannot be available from one component of the subworkgroup but has to be made by composition of internal information of the subworkgroup. Because our application is executed on a platform managing the supervision, it is not desirable to delegate to components the management of the dynamics of workgroups and subworkgroups they are included into. Next, we need to study information directly available when this component is into a subworkgroup with a known composition. After that, it is possible to define rules which build the information needed by each component from available information. We can illustrate this in the previous application if we want to replace the sound by sub-titles (for disabled-listeners for example). When components doing the broadcasting of pictures collaborate with those doing the broadcasting of the sound, they have to exchange synchronization events corresponding to temporal events (end of frame transmission) [3].

On the other hand, when sound becomes text, we have to manage the conformity between speech and text. In the first case, components ensuring the broadcasting of sound are synchronized to those concerning the broadcasting of pictures by the use of temporal events. It is not possible to do the same in the second case, because it is not conceivable to suppose that the component producing sub-titles is designed to synchronize itself from pictures

broadcasted. So, we have to give it significant events to run correctly. We will design a rule which will receive events from components broadcasting the picture and events from components broadcasting the sound of the speaker. These events will permit to synthesize events corresponding to the wording of each sentence. These events will provoke procedure calls to the sub-title component of the subworkgroup to print the corresponding text. In fact, we can imagine that the rule itself do part of the signal processing to detect end of sentences with information provided by both corresponding components.
With this way to tackle the problem, we transfer onto rules part of the activity of the application. This part is not the role of components because it is not issued from the goal to reach but it is issued from how to reach this goal.

5 The platform

Each site needs to have its own instance of the platform. The platform manages the communication between sites, the capture of events and messages provided by each component, and the triggering of rules. Moreover, it has to dispose of XML descriptions of users and to do real-time measurement of the QoS.

5.1 Platform architecture

We propose an architecture close to the one we made in the ELKAR project. Thus, we find the Communication Manager (CM), the Rule Manager (RM), the Element of Cooperation Manager (ECM). Nevertheless, we have to foresee a service to provide XML documents. So we propose a distributed database containing the needed information to realize the supervision of the application. Furthermore, we will find updated information describing the constitution of workgroups, subworkgroups and the characteristics of each component. Each space contains an XML description of the elements it defines :

- The *user space* contains a description of the set of members allowed to be included into the application (login, role, location, language, …) and its requirements in relation to the application.
- The *component space* is a description in terms of roles, functionality and QoS of each component available for the application (procedures, call and return parameters, the site where it is executed, resources needed, …). We will find information on elements needed/created by this component and characteristics about its participation to subworkgroups (conditions to enter/exit, incompatibilities, etc.).
- The *workgroup space* allows to know both present workgroups and subworkgroups, their constitution and the geographical localization of each of their members.
- The *QoS space* (the Adaptor in [4]) contains information about constraints to respect in order to have a good execution of the application according to the QoS to obtain. This information will be expressed with temporal logic [1].
- The *flow space* contains a description of data flow into the application and a description concerning components which create or use these flows.

Some documents are directly issued from the analysis (Component and QoS spaces) whereas others are updated as the application is running (workgroup and flow spaces). To finish, the user space is a configuration of the application done for each new use.

This description of each space using XML, allows to have a generic model but also to validate the coherency using DTD (*Document Type Definition*). With DTD, it is possible to represent documents as graphs [6][7] and do checks and operations on them. Thereafter, the use of XML will facilitate the re-use of these documents because the platform implements procedure calls with SOAP (*Simple Object Access Protocol*) based on XML [12].

5.2 Running of the platform

The role of the various managers (CM, ECM, RM) are conform to those described in ELKAR [8] (communication, elements of cooperation management, rules management). The main modifications with the ELKAR are dynamic reconfiguration of managers using information described into work spaces previously described instead of a static running. The main part of work done by the platform is contained into the rules it uses. The running of these rules is synchronized by elements of cooperation circulating and configured with descriptions available in the different spaces. Some rules will measure performances to check the respect of constraints. These rules may, if needed, take the initiative to modify the composition of workgroups and subworkgroups in order to adapt them to current conditions (workgroup space). For example, a rule receiving messages corresponding to the broadcasting of pictures of the camera (of the speaker) can measure the time between two successive frames and check if it is under the time limit required by the QoS (QoS space). If the throughput becomes inadequate and according to the difference measured, the rule can choose to substitute some components of the subworkgroup broadcasting pictures of this user to replace them with lower quality components (component space). Of course, it not possible to remove a component from a subworkgroup regardless of the

coherency of the set. So it is imperative, before removing a component, to be sure that no other component is waiting for a message or an event from it. Rules managing the supervision of workgroups and subworkgroups have to wait the opportune moment to operate before removing or adding a component. The rule itself can also produce the information waited for a component to avoid its locking.

6 Conclusion

In this paper, we have presented a fully distributed and web-services based architecture for multimedia applications. Our works are at the crossroad of several research domains which are particularly actives : technical inter-operability to insure web-services (multimedia), approaches based on components (software and hardware), QoS and networks. The increasing bandwidth of networks let us suppose that Internet may become more and more accessible for such applications. So we decided to focus on "web-oriented approach". This has two main advantages : components may inter-operate easily using XML messages (becoming a universal technology) and the problem of protocols as IIOP is avoided. More and more, applications are based on distributed components. Nevertheless, and on the contrary to several approach, it does not seems realistic to let components managing both objectives and running constraints. This should be the opposite of the component paradigm. So, with the cooperation model we proposed, components do not have to include the management of constraints (moreover, it may be impossible because they cannot have a global knowledge of constraints). Distributed components based applications are essentially built on the exchange of elements of cooperation (events, messages, data). The control of these exchanges managed by a platform containing all information on the constraints to respect, allows a precise and dynamic supervision of the application. Because of these aspects, re-enginnering methods and those corresponding to cooperative applications put forward solutions particularly suitable to this kind of organization. The structuration on workgroups managing the goal to reach and the QoS gives a global view of the application which is described by a set of documents used by the platform to modify this structure.

7 Bibliography

[1] João Costa Seco, Luís Caires – *A basic Model of Typed Components* – ECOOP 2000, LNCS 1850 – pp 108-128 - Sophia Antipolis, Nice - France 2000.

[3] Le Goff, B., Guiard-Marigny, T., Cohen, M., & Benoît, C. - *Real-time analysis-synthesis and intelligibility of talking faces* - Proceedings of the Second ESCA/IEEE Workshop on Speech Synthesis - New Paltz - New York - U.S.A. - Sept. 1994

[4] Baochun Li, Klara Nahrstedt, *A Control-based Middleware Framework for Quality of Service Adaptations,* in IEEE Journal of Selected Areas in Communication, Special Issue on Service Enabling Platforms, 1999, Vol. 17, No. 9, September 1999, pp. 1632-1650.

[5] M. Liévin, Franck Luthon – *A hierarchical Segmentation Algorithm for Face Analysis – Application to Lipreading* – IEEE Int'l Conference on Multimedia & Expo – ICME 2000 – Vol. 2, pp. 1085-1088 – Now-York - August 2000.

[6] I. Marsic – *Real-Time Collaboration in Heterogeneous Computing Environments* – Proc. Of the int'l conf. on Information Technology (ITCC2000) – pp. 222-227 - Las Vegas - March 2000

[7] L.M. Rodriguez Peralta, T. Villemur, K. Drira - *An XML on-line session model based on graphs for synchronous cooperative groups* - 2001 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2001), Las Vegas (USA), 25-28 Juin 2001, pp.1257-1263

[8] Philippe Roose - *ELKAR : A Component Based Re-engineering Methodology to Provide Cooperation* - 25th Annual International Computer Software and Application Conference (COMPSAC 2001) - IEEE Computer Society Press - PR01372, pp. 65-70 - ISBN 0-7695-1372-7 - Chicago - USA - October 2001

[9] F. Singhoff, I. Demeure – *Environnement d'exécution pour les applications réparties sous contraintes temporelles : une solution CORBA-RTP* – RenPar 10 – Strasbourg – France – 1998.

[10] J.B. Stefani – *Computational Aspects of QoS in an object-oriented, distributed system architecture* – 3rd Workshop on Responsive Computer Systems – Lincold – USA – September 1993.

[11] Tawbi Chawki, Jaber Ghaleb, Dalmau Marc - *Activity Specification Using Rendez-Vous* - 2nd Int'l Workshop on Rules in Database Systems – RIDS'95 – Springer – Lecture Notes in Computer Science – Vol. 985 – pp. 51-68 – Athens – September 1995.

[12] SOAP Version 1.2 - *W3C Working Draft 9* -July 2001 - http://www.w3.org/TR/2001/WD-soap12-20010709/

[13] Dongyan Xu, Duangdao Wichadakul, Klara Nahrstedt, *Multimedia Service Configuration and Reservation in Heterogeneous Environments*, in Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS 2000), 1275-1278, April 10-13, 2000.