Integrated Design and Process Technology, **IDPT-2002**
Printed in the United States of America, June 2002

A SOFTWARE ARCHITECTURE FOR COMPONENT BASED MULTIMEDIA APPLICATIONS

Marc Dalmau, Philippe Roose, 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. Here, 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 context in which it runs. Our approach consists in breaking down the application into two levels : the first one reflects the user's point of view in terms of functionalities. The second one, reflects the way of achieving these functionalities in terms of quality of service.

We propose to organize components into workgroups and subworkgroups corresponding to these 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 work was about a method for re-engineering of applications to provide cooperation. This method considers applications from an inter-module communication 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.

When examining how distributed multimedia applications are built, one can see several common points with our previous work. Actually, such 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 built up to realize a common task using communication. So, these workgroups of components may be considered as « super-components » participating to the realization of the objective of the application thanks to intercommunications. 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 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 constitute such applications with dynamic workgroups (groups of components evolving with time). Such groups are managed by rules included into the platform and allowing 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, not the internal functioning of components view but about their fitting and exchanges. 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 or creating some information in the communication scheme.

2 PREVIOUS WORKS : ELKAR

ELKAR (Roose 2001) 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). Then, the application uses the platform whose working is defined with ECA rules (Event-Condition-Action) formalized by the method. We are conscious that we could not always have all elements of cooperation we need for an efficient working, so we will use some specific rules (detective rules (Tawbi, 1995)) to compose algorithmically existing elements to build the missing ones.

2.1 THE METHOD

The method is entirely based on communication. It has to manage elements of cooperation which will be exchanged between components 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 components into workgroups. This cutting out should reflect an organizational view of the application but also cope with 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 constitution rules. It is necessary to be sure that incompatible components may not be included into the same workgroup 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 point, we have the totality of elements constituting the application : actors (components 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 an actor). So, we need to define operators allowing to create them from existing elements. We use detective rules which role is to capture some information available in order to create the one 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 rules 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 in order to send information to local but also to distant members.
- *An Element of Cooperation Manager (ECM)* : it receives all the information produced on this site and also from other sites transmitted 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 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.
- *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 required since modules have not been a priori designed to work cooperatively with others, but only to exchange information with their immediate environment.

All these managers are implemented as parallel processes and coexist with rule processes. The existence of such a platform with a knowledge of the application (workgroup structure, information exchanges) allows to completely separate the functional aspect from the organizational aspect. So, the analysis provides a detailed description of exchanges and how to implement the structure of the application.

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

2.3 COOPERATION AND MULTIMEDIA

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.

There are three kinds of communications elements 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. This method is implemented with Beans in Java.
- *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.

However, it is important to keep in mind that multimedia applications are characterized by their continuous data flow (video, sound) with a high rate. A multimedia flow can be characterized (Singhoff, 1998) by a succession of procedure invocations decomposed into events. These events correspond to the transmission/reception of the invocation (see Figure 1). So it is possible to see a multimedia flow as a succession of events.

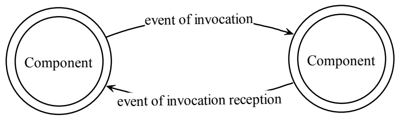

Figure 1 : Events issued from the invocation procedure

2.4 INTERACTIVITY AND QUALITY OF SERVICE

Multimedia applications are naturally very interactive. They need to adapt themselves, in real-time, to the needs of users. Each of them wants to get a personalized view of the application where he 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 in real-time to these evolving conditions in order to offer the best QoS as possible, at each moment.

With a web-service framework, the QoS may depend on (Chassagne, 1997) :
- The delay, i.e. the time between the sending and reception of a packet. The delay includes also the propagation delay and the transmission delay of datagrammes relative to intermediate systems.
- Delay variation from the start point to the end point.
- Maximum availability (error average level of a link) maintained between two terminal points.

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 as 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.
- *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 of components in the group.
- *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.

To illustrate this, we will see a proposition for a multimedia application like videoconference with automatic speaker detection and tracking using image processing technics (Liévin, 2000).

3 PRESENTATION OF AN EXAMPLE : UNSUPERVISED VIDEOCONFERENCE

To illustrate our purpose, we present a short example of videoconference (Figure 2).

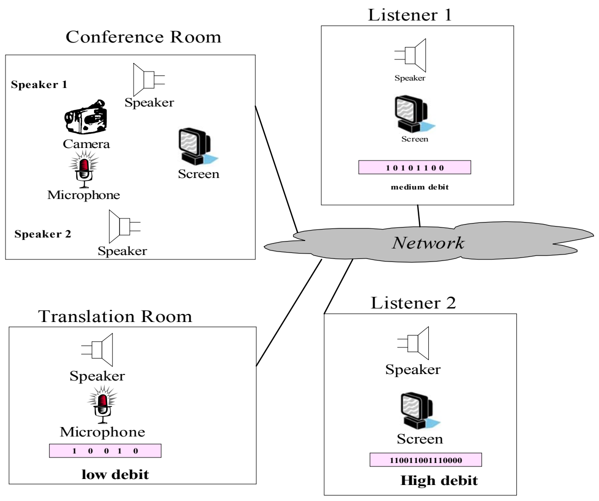

Figure 2 : Physical localization of the videoconference participants

3.1 WORKGROUPS AND SUBWORKGROUPS ORGANIZATION

A workgroup will be associated to each participant. Among persons participating to the videoconference, some of them may speak (*speakers*) whereas others just follow the discussion (*listeners*). Moreover, we also have *translators*.

Therefore, we will find three kinds of workgroups : speakers, listeners and translators. Each of them will be constituted of several subworkgroups according to the functionalities of the workgroup.

❑ Speakers Workgroups

Each speaker has functionalities to capture and restitute pictures and sound and to realize the detection and tracking. This allows to track him as he speak and to detect when there is a speaker's exchange. These workgroups are made of three subworkgroups.

- ***Picture Subworkgroup***
  - *Camera Component (CAM)*
  - *Picture Capture Component (PCC)*
  - *Picture Restitution Component (PRC)*
- ***Sound Subworkgroup***
  - *Sound Capture Component (SCC)*
  - *Sound Compression Component (SC)*
  - *English Translated Sound Restitution Component (ETSRC)*
- ***Tracking Subworkgroup***
  - *Speaker Tracking Component (STC)*
  - *Speaker Research Component (SRC)*
  - *Oral Intervention Detection Component (OIDC)*

❑ Translator Workgroup

When the speaker speaks in French, a translator provides a simultaneous English translation (the hypothesis is that all participants understand English). This workgroup is made of two subworkgroups :

- ***Sound Subworkgroup***
  - *Speaker Restitution Component (SR)*
  - *Sound Compression Component (SC)*
- ***Translation Subworkgroup***
  - *Sound Capture Component (SCC)*

❑ Listeners Workgroups

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.

- ***Picture subworkgroup***

Components and number of components constituting a subworkgroup will vary according to external constraints and particularly to those of the network. Thus, if during a laps of time, the available throughput does not allow the direct transmission of pictures of the speaker to other sites, it will be possible for these sites to replace components doing the transmission of pictures 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.

  - *High Quality Picture Transmission Component (HQPTC)*
  - *Medium Quality Picture Transmission Component (MQPTC)*
  - *Picture Compression Component (CI)*
  - *Picture Restitution Component (RI)*

To implement such a flexibility, we of course need information from the user side (what he may accept in terms of loss of quality) but we also need a constant evaluation of conditions of the running environment. We propose that the platform adapts the subworkgroup composition according to performance measures done on the application and according to the characteristics of users available into XML documents.

Identically to the picture subworkgroup, the sound subworkgroups will adapt themselves to the possibilities of the network but also to the needs expressed by users.

- ***Original Sound Subworkgroup***
  - *Speaker Restitution Component (SR)*

- *Sound Compression Component (SC)*

- ***English Translated Sound Subworkgroup***
  - *Speaker Restitution Component (SR)*
  - *Sound Compression Component (SC)*
  - *English Translated Sound Restitution Component (ETSRC)*

We can see that the addition or the substitution (Costa, 2000) of a subworkgroup by another into the workgroup associated to the user may adapt the service to the needs of the user. In particular, we pay attention to always transmit according to the listener language. 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 that each user can fill in. The platform will refer to these documents to constitute workgroups.

Here is briefly an organization when speaker 1 talks in French. We suppose that speaker 2 and listener 1 do not understand French whereas listener 2 does.

Speaker 1 (French) (speaking)	**Speaker 2 (Fnglish)**	**Listener 1 (English) Medium Debit**	**Listener 2 (French) High Debit**
Picture ❑ CAM ❑ PCC	**Picture** ❑ PRC	**Picture** ❑ *MQPTC* ❑ PRC ❑ CI	**Picture** ❑ HQPTC ❑ PRC
Sound ❑ SCC	**Sound** ❑ RSTA	**Sound Translated** ❑ *ETSRC* ❑ SC	**Original Sound** ❑ *SR*
Tracking ❑ *STC* ❑ Dloc	**Tracking** ❑ *SRC* ❑ *OIDC*		

Translator (Fr -> E) Low Debit
Sound ❑ *SR* ❑ SC
Translation ❑ SCC

3.2 COMMUNICATION BETWEEN COMPONENTS

Components of multimedia applications are naturally designed to communicate. One could think that the only problem is to transmit on the network these elements of cooperation. This would be true if the organization of the application were not dynamically modified by the platform (Xu, 2000). In fact, 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 ?

It seems to us not realistic to impose that components know how to dialog with all eventual collaborators. Such a rule is against our principle of flexibility, because it means that we need to manage all possible combinations of components into a subworkgroup before designing it !

Moreover, some essential components for a multimedia application are not easily adaptable to a workgroup based running. We think particularly of components near the hardware (camera drivers, picture or sound capture, …). Moreover, these components will be often modified to suit a new hardware functionality or new hardware availability. Can we imagine to have to re-implement a part of the application when we change the videocard ?

In the same way, subworkgroups of components we have constituted have to appear as 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 single component of the subworkgroup but has to be built 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. So, it seems opportune that these problems are solved by the structure of the application and not by the design of components. Such an organization allows to reuse components not initially designed to directly cooperate into workgroups.

We encounter here a problem similar to the one we got with the re-engineering of applications which were not designed to cooperate. The solution of such a problem requires to study precisely information produced and received by each component. 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 from available information, information needed by each component.

Of course, sometimes, a direct link will be enough and no rules will be used. In some other cases, just a formatting will be needed. For example, when we have to adapt the parameters of a procedure call to be accepted by the addressee. But sometimes, we need to create new events or messages by composition of other existing events and messages (Aniorté, 1999).

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) (Le Goff, 1994).

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.

4 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.

4.1 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.

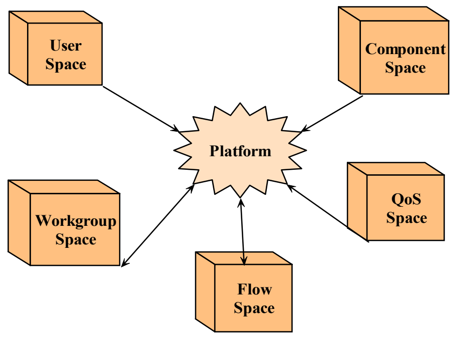

Figure 3 : General Architecture

Each space (see Figure 3) 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, eventually 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 (Li, 1999)) 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 (Costa, 2000).
- 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 derived 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.

The 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 (Marsic, 2000) (Rodriguez, 2001) 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 (W3C, 2001).

4.2 RUNNING OF THE PLATFORM

The roles of the various managers (CM, ECM, RM) are conform to those described in ELKAR (Roose, 2001) (communication, elements of cooperation management, rules management). The main modification with the ELKAR is the dynamic reconfiguration of managers using information described into workspaces previously mentioned instead of a static running.

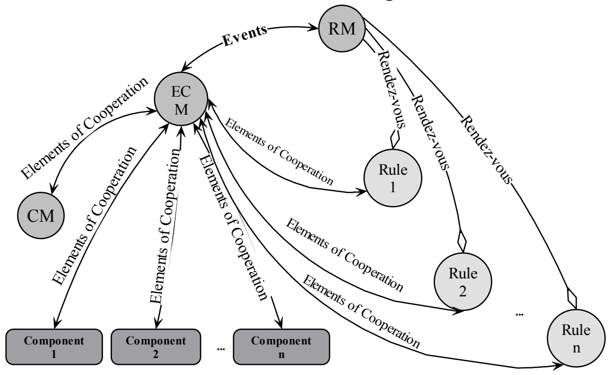

Figure 4 : The platform

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 from 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 nor 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 by a component to avoid its locking.

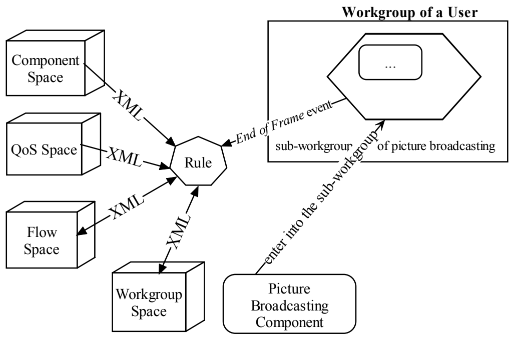

Figure 5 : Rules activity

5 CONCLUSION

The particularity of our approach is its fully distributed organization, open and web-services oriented on the contrary to POLKA (Demeure, 1998) for example. The POLKA approach is distributed but is proprietary due to the use of CORBA. Actually, high throughput networks become more and more available, so why do not use a « web oriented approach » allowing components to inter-operate without protocols such as IIOP or CORBA which are too restrictive.

Advantages are numerous : this approach provides facilities for components to inter-operate. It also has the advantage to be based on universal internet technologies but also, to allow a more nomad and flexible use.

So we 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. Considering the increasing complexity of applications, especially distributed multimedia ones, and the growing of requirements from the user side, we have to design applications able to manage both objectives and running constraints. It do not seems realistic to let components manage (in real-time) all of these constraints. This is definitely not the philosophy of the component paradigm. So, we have to provide a cooperation model in which components just have to ensure the work they have been designed for and not to deal with external constraints. Moreover, in this case, it would be very difficult for them to have a global knowledge of these constraints.

These new applications are essentially built on exchange of elements of cooperation (events, messages, data) between components. The only knowledge of these elements allows to know how the application works and so, to have a view of what it is doing. The control of these exchanges managed by a platform containing all information with the constraints to respect, allows a precise and dynamic supervision of the application.

Re-enginnering methods and those corresponding to cooperative applications put forward solutions particularly suitable to this kind of organization. The structuration based on workgroups managing the goal to reach and the QoS gives a global view of the application. This global view is described by a set of documents used by the platform to modify this structure.

To finish, we think that it would be utopian to imagine that it is possible to reuse existing components and to integrate them into workgroups without any intervention. That is why we propose to use detective rules which role is to ensure the inter-operability of components and to provide information not directly available or to translate it into the correct format.

6 REFERENCES

Tawbi Chawki, Jaber Ghaleb, Dalmau Marc, 1995, *Activity Specification Using Rendez-Vous,* 2nd International Workshop on Rules in Database Systems, RIDS'95, Springer, LNCS, Vol. 985, pp. 51- 68, Athens.

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

C. Chassagne, 1997, *Qualité de Service dans l'Internet*, CNRS-UREC, http://www.urec.cnrs.fr/metrologie/qos.html

M. Liévin, Franck Luthon, 2000, *A hierarchical Segmentation Algorithm for Face Analysis – Application to Lipreading*, IEEE International Conference on Multimedia & Expo, ICME 2000, Vol. 2, pp. 1085-1088, New-York.

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

Dongyan Xu, Duangdao Wichadakul, Klara Nahrstedt, 2000, *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.

Aniorté Philippe, Roose Philippe, Dalmau Marc, 1999, *Using active Database Mechanisms to Build Cooperative Work*, Journal of Integrated Design & Process Science (JIDPS), Vol. 3, N°1, pp1-14, ISSN : 1092-0617.

Le Goff, B., Guiard-Marigny, T., Cohen, M., & Benoît, C., 1994, *Real-time analysis-synthesis and intelligibility of talking faces*, Proceedings of the 2nd ESCA/IEEE Workshop on Speech Synthesis, New Paltz, New York.

Baochun Li, Klara Nahrstedt, 1999, *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, pp. 1632-1650.

I. Marsic, 2000, *Real-Time Collaboration in Heterogeneous Computing Environments*, Proc. of the International Conference on Information Technology : Coding and Computing (ITCC2000), pp. 222-227, Las Vegas – USA.

L.M. Rodriguez Peralta, T. Villemur, K. Drira, 2001, *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

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

Philippe Roose, 2001, *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.

Isabelle Demeure, Frank Singhoff, Francois Horn, 1998, *Automatic Scheduling of a Dynamic Multimedia Application with Polka*, A Case Study Fourth IEEE Real-Time Technology and Applications Symposium (RTAS'98), WIP, pages 15-19, Denver, Colorado, USA.