# FADYRCOS, a semantic interoperability framework for Collaborative Model-Based dynamic reconfiguration of Networked Services

Aymen Kamoun, Saïd Tazi, Khalil Drira

**Title:** FADYRCOS, a semantic interoperability framework for Collaborative Model-Based dynamic reconfiguration of Networked Services

Authors: Aymen Kamoun, Saïd Tazi, Khalil Drira

**Abstract**

The increasing interdependence of economies across the world has stimulated the collaborative development of complex products implying wide ranges of groups. In this context, the collaborative development of products provides new challenges in distributed systems. It requires continuous communication and exchanges between teams of collaborators having different roles and using different tools. A global model of collaboration is necessary to guarantee the quality of communication and to ensure adaptability and interoperability between tools whatever may happen. In this paper, we present a framework for collaborative model-based networked services development that supports a semantic adaptation model enabling the awareness of the presence, roles and tasks of collaborating actors. In this article, the implementation of the framework FADYRCOS and its conceptual model are presented. Algorithms that implement dynamic reconfiguration are also presented. A test case for collaborative software development has been developed to validate the framework.

**Key words:** collaboration; interoperability; semantic adaptation; dynamic reconfiguration

# 1 Introduction

Collaboration between enterprises requires more interoperability of services in dynamic ways that support the growing complexity of business processes. The collaborative development of complex products provides a wide range of new challenges in distributed systems. Teams of engineers have different roles and experiences; they must deal with numerous types of information both on the product being developed, on the technology used such as the development tool, but also on the organization adopted for work. Due to the size and the complexity of groups and products, the process may require several iterations of design, development and test and hence requires sophisticated modes of connections. Moreover it is important to guarantee transparent collaboration of actors respecting the different roles they are playing and despite the distance and the disparate tools they are using for collaboration and communication.

Teams need to be connected in a manner they feel themselves that the collaboration is continuous and fluid. It is important to ensure the continuity of communication services even if changes happen such as the absence, the presence or the role change of a collaborator. A global model of collaboration is necessary to guarantee the quality of communication and to ensure adaptability and interoperability between tools whatever may happen of such events. A global model of collaboration is necessary to guarantee the quality of communication and to ensure dynamic interoperability between tools.

The design of an appropriate system to deliver intelligently the communication flows between the different parts in all situations ensuring the continuum of collaboration is one of CSCW (Computer-Supported Collaborative Work) research challenge [1]. For example if a collaborator joins/leaves or changes his role within a project, how the system may enable/disable automatically the communication flow adapted to its context of use such as its role, optimized Quality Of Service parameters? How his collaborators may continue their work despite this change? And how the system may ensure the continuum of collaboration by providing new flow of communication regarding the role of each collaborator and their tasks? In this perspective, we propose to explore promising research methodology grounding on analysis of semantic collaboration to ensure interoperability and thus the communication. Semantic interoperability is the combination of the meaning of data and process actions shared in a collaborative work to ensure interchange between human being implied in the collaboration and between the systems that may be the subject or the means of collaboration.

In this paper, we present FADYRCOS (A Framework for DYnamic Reconfiguration of networked COllaborative Systems) that supports semantic adaptation enabling the awareness of the presence/absence, roles and tasks of collaborators. This framework is based on a generic multi-level modeling approach that ensures multi-level adaptation. A generic collaboration model, based on Semantic Web technologies is proposed in order to support real-time collaboration between groups of engineers working together in different enterprises. The framework defines common interfaces for collaborative systems to enable the management of cooperative actions. In the present work we focus on collaboration between individuals belonging to several teams working on different projects.

Usually, individuals participating to the collaborative activities are organized within sessions. A session is composed of a set of users that share common interests. These users may get connected to the session from different places. They manipulate data shared through the groupware tools that support collaborative activities. Sessions can be synchronous or asynchronous. In a synchronous session, user co-presence is required to realize the collaborative work. Users communicate interactively and handle the collaborative tools and data of the session in real-time. In an asynchronous session, user co-presence is not required. As a consequence, users can not perform interactive exchanges, and communication is made through asynchronous media. Swapping between asynchronous and synchronous mode is now

possible in recent collaboration support tools. For instance, collaborative production of an artifact can occur in an asynchronous mode, but synchronous sessions can be created to synchronize and coordinate artifact production. Sessions are called explicit when all session configurations are defined offline, before session enactment. Relations between members are clearly defined. Such sessions, with their evolution in time, are firstly defined by a privileged user called session designer. Then, systems handle the defined session instances. Other users can join the defined sessions or they can be invited by the session designer. The three main elements of sessions (i.e. users, hardware devices, and software tools) are explicitly modeled.

In this paper, we focus on architectural adaptation in collaborative environments in which participants have different roles and belong to one or many groups. Therefore, they are organized within sessions dynamically depending on their roles or tasks. In order to achieve this adaptation, a set of interconnected components must be deployed in the system. Since sessions are dynamic and participants may change during the collaborative activities, an adaptive components deployment must be considered in such systems to preserve this collaboration. This deployment is needed for establishing multimedia sessions between participants to support collaboration. Adaptation actions in these systems involve components (re)deployments and exchanged flows reconfigurations. For instance, if a new participant wants to join a group in a specific session and needs to collaborate in audio, the system should be able to deploy an audio component in his machine. An audio flow must be setup between this component and the other components deployed in the devices of participants with whom he can communicate. Therefore, a new deployment schema is generated by the system to ensure the communication.

The rest of this paper is organized as follows. Section 2 presents related works, we focus especially on collaborative activities and session management. Section 3 presents the proposed multi-level modeling approach and the framework FARDYCOS for adaptive group communication. Section 4 discusses the maintaining of the interoperability of different applications domain. Section 5 presents a case study and section 6 concludes the paper.

## 2 Related work

Significant research has been carried out on collaborative activities and session management. A majority of these solutions deal with different aspects of collaboration and communication. However, as far as we know, very few works treat specifically the problem of providing tools for building context-aware collaborative applications with dynamic reconfiguration of components in runtime.

In CSCW, a groupware is defined as “Computer-based systems that support groups of people engaged in a common task to reach a common goal”. They provide an interface to a shared environment [2]. There are many technical and social challenges that have to be faced in designing groupware products. Examples of technical challenges are conflict of operations, notifications, etc. Many issues has been resolved such as coherence and consistency of actions [2]**,** face to face interaction [3], unpredictable actions that can be carried out in several places [4]**.** Several taxonomies of CSCW have been proposed according to many criteria like functionality [5], purpose [6] and domain of application [7]**.** Most of published collaborative tools focus on communication (messaging) and coordination features (approval forms, video-conference), but few of them deal with collaboration amongst actors [8]. Some research deal with the content of communication, studies such as [9] show that the co-designers spend much time clarifying what it was written in shared documents

According to Dourish et al. [10], “Awareness of individual and group activities is critical to successful collaboration and is commonly supported in CSCW systems”. In CSCW, the notion of group awareness presents a significant feature. Dourish et al. [10] defined it as "an understanding of the activities of others, which provides a context of your own activity”. This concept depends on the group topology such as members who have strong or weak relationships or members who have different or same languages and experiences [11]. Moreover, as members are not co-present in virtual workspace, it is prominent to create a group awareness model based on different methods such as providing a clear synthesis of all information from different parts of system.

Communication presents a fundamental feature in collaborative workspace. Teeni [12] defines it as an act of building a mutual understanding between the sender and the receiver. Communication issues can increase when the sender and the receiver are distant. During team meetings, the members share documents and they can explain directly what they mean. But, communication problem becomes more complex when participants are distant collaborating in an asynchronous mode. As a result, communication matters can have a direct drawback on the shared knowledge. This can have thereafter an impact on project progress. Many studies have so addressed mediated communication issues between designers [13, 14].

Baudin et al. [15] has thoroughly studied graph-based explicit session management models and services in order to support collaboration inside groups of human users. The goal was to explicitly model relationships of information exchange between users in order to keep a tight coupling between communication and network layers. The proposed graph-based collaboration model is based on data producer/consumer relationships. Three elements have to be managed: users, tools and data flows. Vertices represent users and the labeled edges model the dataflow relationships between users. Therefore, data exchanges for synchronous and interactive work sessions are represented and modeled. Such sessions handle interactive data flows (e.g. video, real time audio). The dynamics of the session is expressed in terms of users entries/exists and user role changes. The session designer explicitly selects collaboration

graph structures where collaborative work can occur. Advantages of this collaboration model are twofold. Firstly, this model is simple enough to be easily handled by session designers for various collaborative configurations. Secondly, instances of this model can be automatically taken into account by services or platforms that can be configured by the model. The sessions explicitly designed are therefore managed by model-based platforms. Sessions management services are used in order to manage all session's elements over collaborative systems. These services provide also data flow management and tool management. Deployment of collaboration tools is important in order to ensure session's enactment and collaboration between individuals.

Implicit sessions emerge from the observation of users' actions and their context. When the system detects a potential situation of collaboration, such as human presences, it creates an implicit session and invites involved users to join it. As far as we know, few works were tackling implicit sessions. However, we can cite [16, 17] where models are based on set formalisms or [18] where models are based on first-order logic formalism, in order to describe unstructured sessions.

We have analyzed existing synchronous collaborative systems such as TANGO [19], HABANERO [20], DOE2000 [21] and DISCIPLE [22]. The main disadvantage of these systems is that the group member roles are pre-defined and cannot be changed dynamically during the collaborative activities. Furthermore, either the "master-Slaves" roles or the same role and identical rights for all participants are supported. Consequently, these systems cannot support complex structured sessions. Thus, model-based approaches are required in order to describe session's organization and to ensure the flexibility in the described systems. Moreover, several approaches propose models controlling only one element of the session such as media [23], communication [24, 25], floor control [26], quality of service control [27] and collaborative services [28]. Other sessions models describe only three components of a session: users, tools and data flows [16, 29]. These proposed models create collaborative sessions by monitoring members' activities. Some of them support dynamic change and provide a representation of the sessions, but this representation is too specific to the model, what restricts its use in other systems.

Some platforms provide tool-based implementation for the session management, whereas, they don't support the representation of complex structured sessions, and even those which allow complex sessions modeling are still theoretical and they don't provide real implementations.

Other awareness tools have been proposed: The TeamScope system [30] allows participants exchanging files and working on it without real time interaction between them. The Classroom BRIDGE [31], the Collablogger tool [32] and the CAliF system [33] support distributed group projects. However, they are not able to support dynamic changes of the session's structure.

Other ontologies-based works are proposed and applied to different problems of CSCW. Gu et al. [34] present an onotology-based system for collaborative editing and design documents. The system aims to avoid semantic conflicts. Yao et al. [35] introduce an ontology-based system for workflow management. This system can interact with other ontology-based applications. Andonoff et al. [36] proposes an ontology of high level protocols for agents conversations. Ontologies are used in order to provide semantic to these protocols and to ensure automation of coordination in distributed systems. Garrido et al. [37] propose an MDA-based approach for modeling enterprise organization and developing groupware applications. The domain model is formalized through a domain ontology in order to describe concepts and relations between actors sharing knowledge. Tomingas and Luts propose a semantic interoperability framework for data management like web services descriptions and ontologies [38].

The main disadvantage of classical CSCW systems is the lack of flexibility in sessions which are pre-defined and don't support dynamic changes, what cause a problem of interoperability. Indeed, a given CSCW system which supports collaborative activities in a specific domain and in particular in a specific group organization this system cannot coordinate collaborative activities in another domain; and even in a particular working group, sessions and roles organization cannot be changed. Few works consider implicit sessions which are spontaneous and depends on implicit collaboration activities. In systems represented above, sessions are initiated explicitly and the dynamic of changes is limited to roles changes. Moreover, if a new situation of collaboration is required at runtime, the system is not able to provide the adequate service which satisfies users' needs. Thus, session designer should define new session structure. Furthermore, even if a system defines all possible sessions, it should be implemented and hard-coded. This method limits the code reutilization and its maintenance. Therefore, a generic framework that defines common vocabulary for session modeling is needed. Session designers should easily define and implement new session structures for different groups working together and even for the same group at runtime.

Another disadvantage of these systems is the lack of rigid deployment services of components or application that they offer. Components and applications are often deployed manually on participants' machines and fixed at design time by a static way. This method cannot be applied to situations that need a high degree of adaptation, and in which even not known components should be deployed in advance.

In our work, we intend to support collaboration in distributed environments where sessions can be implicit, new mechanisms are needed for managing session evolution and role changes. In our view, Semantic Web techniques are well suited to achieve this task. As far as we know, there is no existing collaborative system that use semantic for session management and dynamic deployment service. As thus, in this paper, a semantic driven framework has been developed to enable session management and dynamic components deployment for collaborative systems. The next section will describe our approach towards the Framework we have developed.

# 3 FADYRCOS

This section presents our proposed approach for the modeling and the implementation of collaborative activities. We propose a generic framework enabling session management, semantic multi-level adaptation and automatic components deployment for collaborative activities called FADYRCOS (Framework for dynamic reconfiguration of networked collaborative systems).

## 3.1 Multi-Level modeling approach

In order to clearly separate different concerns in our approach, a multi-level modeling architecture is proposed (Figure 1). Each level encapsulates specific problems into a specific model. Models represent sets of linked entities like software components, called configurations. High level configurations represent high level requirements, such as participants' organization and their activities, while low levels models represent the real implementation ensuring such activities. Separation between levels may introduce the concept of multi-level adaptation. Indeed, distinguishing different abstraction levels allows mastering the specification and the implementation of adaptation rules.

A configuration is denoted $C_{n,i}$, where $n$ is the abstract level and $i$ is an index to distinguish the configuration. For a given configuration $C_{n,i}$ at level n, multiple configurations ($C_{n-1,1},\ldots,C_{n-1,p}$) may be implemented at level n-1.These configurations represent the refinement or a detailed view of $C_{n,i}$. When a new adaptation is required at level n, only one configuration is chosen among all configurations of this level implementing the new n-level configuration.

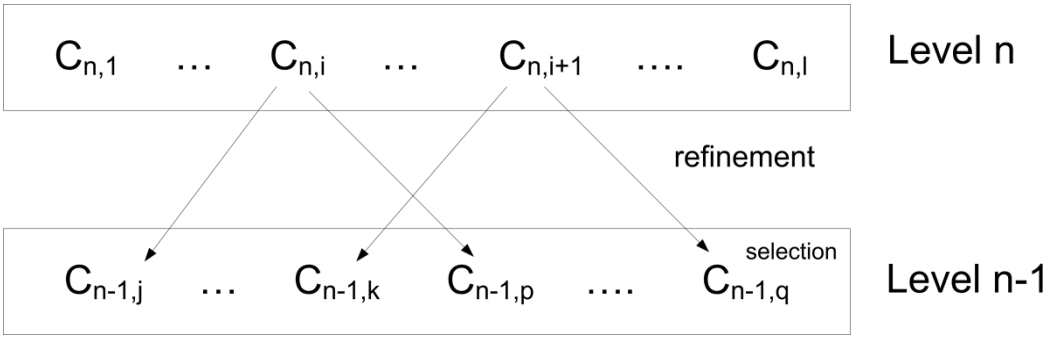


**Figure 1. Multi-level architecture modeling**

This adaptation requires two actions:

- Refinement which determines the set of associated (n-1)-level configurations that implement a given n-level configuration $C_{n,i}$. This refinement procedure is called *refine ()*. Figure 2 represents the generic version of this function. This function is generic because the action "implements" depends on the considered level.

$$\text{Refine } (C_{n,i}) \{$$
$$\text{return } \mathbb{C}^{i}_{n-1} = \{\ C_{n-1,j} \in \mathbb{C}_{n-1} \text{ where } C_{n-1,j} \text{ implements } C_{n,i},\ j \in \mathbb{N}\ \}$$
$$\}$$

**Figure 2. Refine function**

- Selection, which consists in retaining the optimal configuration among the all possible configurations at a given level. The selection procedure uses the resources context such as the variation of communication networks to eliminate the configuration that cannot be deployed depending on the current network architecture. Among the selected configurations, the choice of the best configuration will depend on a set of predefined policies.

## 3.2 Multi-Level modeling for collaborative systems

In this part, we present how we apply the presented multi-level approach to collaborative systems in order to clearly identify specific problems of each level for the considered system. Therefore, we provide an overview of the retained level and we detail the proposed models for each level. Adaptation at the highest levels should be guided by the evolution of activity requirements. Adaptation at the lowest levels should be driven by execution context constraint changes. The retained levels are represented in Figure 3. These levels are detailed as follows:

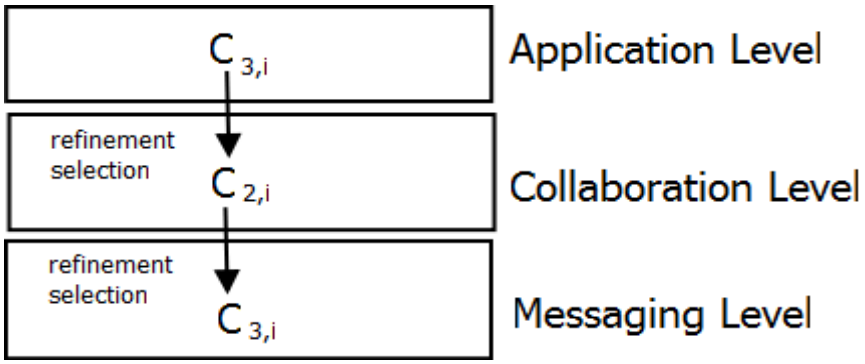


**Figure 3. Collaborative systems levels**

**Application level**

The application level models systems needs collaboration between participants that belong to several groups. The proposed model for this level represents a business view of the collaborating entities and the relations between them. Hence, the system takes into account the extern context of entities when instantiating this model. As this model is highly application-dependent, it must be built by the designers of each collaborative system and instantiated at runtime by the system itself. System designers also have to implement the refinement procedure allowing to obtain the set of collaboration level models that implement a given application model. Only collaboration-related elements of the application level model will be taken into account in the refinement process. Although this business view is often specific to one application, it is possible that several applications belonging to a given domain share the same application model.

**Collaboration level**

The collaboration level provides a session level abstraction. It describes how the members of a group are organized within sessions where they can send and receive data flows. The associated model provides a generic high-level collaboration schema that responds to the current collaboration needs of the system. Indeed, it manages collaboration sessions and all needed elements that implement such sessions like flows, components, etc. The abstract representation of these elements is independent of their implementation and they can be implemented with several technologies which will be detailed in the messaging level. Thus, this level ensures the interoperability between the collaboration' needs expressed in the application level and the real implementation in the messaging level. The architectural model produced by this level is a graph containing the following elements: *nodes*, *components* and *data flows* (which are organized within one or more *sessions*). This model is inspired by classic graph-based session description formalisms such as dynamic coordination diagrams [15].

**Messaging level**

The messaging level provides a communication model that masks low-level details (like TCP sockets, UDP datagrams, IP addresses, multicast, etc.) in order to simplify the representation of communication channels. In fact, this level abstracts distributed systems, so they are transparent for upper levels. For instance, this model may be based on abstractions like Peer to Peer [39], Remote Procedure Calls [40], CORBA [41], Event Based Communications (EBC) [42] etc. The produced model represents a deployment descriptor that contains all needed elements to implement sessions provided by the collaboration level. For instance, if this level is implemented with an Event Based Communication technology, the model contains the event producers, event consumers and channel managers to be deployed on each node, as well as the pull/push links between these elements. If a Peer-to-Peer implementation is considered, model contains peers, super-peers and pipes to link peers.

## 3.3 Multi-level collaborative framework

In order to apply the multi-level modeling approach in our work, we propose a model driven framework enabling implicit session management and adaptive component deployment for collaborative systems.

### 3.3.1 Framework architecture

The proposed architecture (Figure 4) is composed of four levels.

The lower level (infrastructure level) contains all needed software components and hardware infrastructure that enable to run the collaborative system using the framework. Hence, it ensures communication between participants' devices. In the following, we detail the three main levels of the architecture and the retained models.

**Collaboration level**

The goal of the central level (collaboration Level) is to provide collaboration schema that enables members belonging to several groups to communicate inside sessions where they can send and receive data flows. Thus, the main issue in this level is to determine which data flows have to be created in order to enable the needed communication. The proposed elements in this level are represented by a generic collaboration meta-model (GCM) which details the structure of one or more collaborative sessions and deployment configuration. Hence, new instances of this meta-model are generated at runtime after every context change of the application level configuration as well as arrivals, changing roles, changing groups of collaborating participants. GCM is a graph expressed in the GraphML language (an XML dialect for representing graphs [43]).

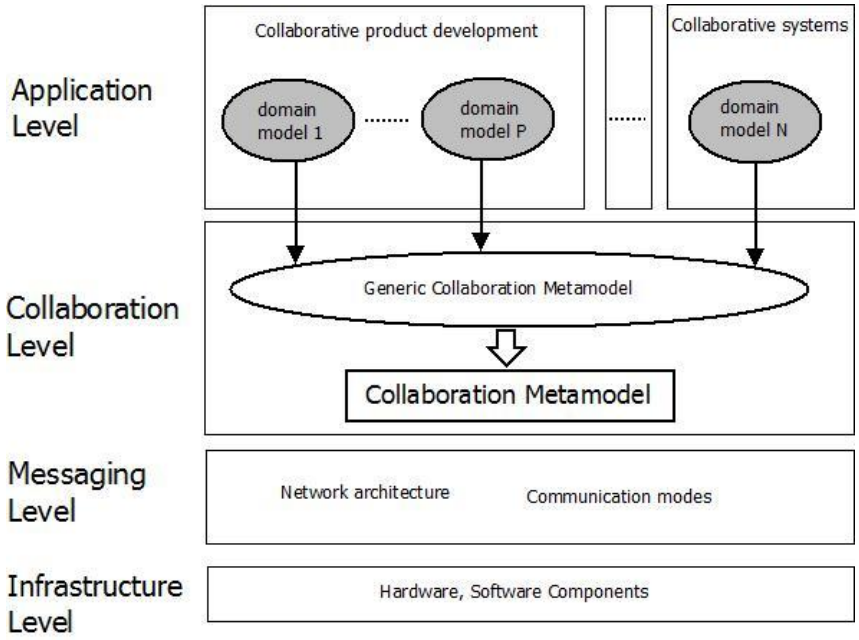


**Figure 4. Framework architecture**

In order to represent GCM, We have chosen ontologies because it constitutes a standard knowledge representation system, allowing reasoning and inference. Moreover, ontologies facilitate knowledge reuse and sharing through formal and real world semantics. Therefore, ontologies are high-level representations of business concepts and relations. We have chosen to describe levels' models in OWL [44], the Semantic Web standard for metadata and ontologies.

In general, ontologies are separated in two levels: a generic ontology and a specific ontology. The former is a domain-wide ontology, but independent of applications. The latter ontology extends the generic one with terms specific to an application category. We have followed the same pattern in our implementation: there is a generic ontology, GCM, (that describes sessions, participants, roles, data flows, nodes, etc.) and an application ontology that extends the collaboration ontology with business-specific concepts and relations.

The generic ontology is common to all applications, therefore it is provided within the framework. This ontology which represents our Generic Collaboration Meta-model is detailed in Figure 5. The proposed concepts are represented by round-cornered rectangles, while relations are represented as arrows going from one concept (the domain) to another (the range).

The main concept is *Session*. A session is a set of flows, represented by the concept *Flow*, which represents communication links between entities. A property, called *HasdataType*, relates the concept *Flow* to the *DataType* concept. Possible values of data type are captured by the *DataType* instances that can be text, audio, video or an exchanged artifact between participants. During the collaborative activities, flows are exchanged between communicating entities represented by the *Node* concept. Therefore, *Flow* is related to *Node* by the two relations: *source* and *destination* representing respectively the source node and destination node. Nodes may represent participants. A given node plays one or more roles determining the type of each activity for all involved participants. For example, a role may be a de developer in a collaborative software engineering. Depending on their roles, entities will have multiple tasks and they need to communicate with different members organized within different groups in order to achieve collaboration goal.

Each role belongs to one or several groups. Therefore, the concept *Role* is related to the concept *Group* by the relation *belongsToGroup*. In order to manage collaboration sessions, a set of session must be defined for each group. Therefore the relation *hasSession* relates the *Group* concept to the *Session* concept. The session definition for each group enables a valid deployment of appropriate sessions depending on participants' roles and groups.
Whether a node represents a collaborative entity that communicates with others entities, it has to be hosted by a physical machine represented by the *Device* concept (relation *HostingDevice*).

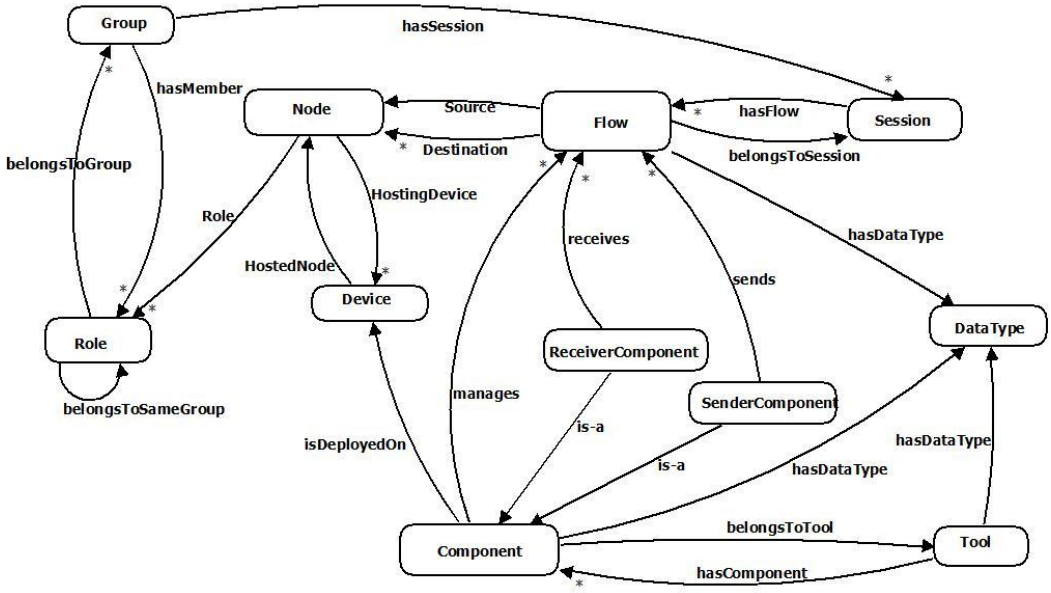


**Figure 5. Generic Collaboration Meta-model**

Nodes manage exchanged data flow using external components (*Tool* concept) deployed in the same devices in order to enable the separation between collaboration code (implemented in the external components) and the business code (implemented in entities' components and specific to the application). Tools are composed of one or several components (e.g. a sender component and a receiver component) represented by the concept *Component*. The *Tool* concept is related to the *Component* concept through the relation *hasComponent*. Components manage several flows. Therefore, the relation *manages* relates *Component* to *Flow*. Components have the same data type of the managed flows (relation *hasDataType*). Each component is deployed on one device (relation *isDeployedOn* which links *Component* and *Device*). The component' type depends on the handled data type (text, audio, video, files, artifacts …) and on the communication mode (real-time communication, asynchronous communication).

## Application Level

The retained model in this level is a domain specific ontology that represents concepts and relations modeling the context of the application. Such ontology depends on the application domain, hence it will be provided by the designer of the collaborative system using the framework. In general, this ontology is a specialization of the Generic Collaboration Meta-model presented above in order to enable the addition of specific semantics to the general collaboration concepts. Moreover, this ontology can be more complex depending on the application' needs.

The main generic collaboration elements that will be specialized are: *Node*, *Group*, *Role* and *hasSession*. The concept *Node* is specialized into sub-concepts specifying the type of the communicating entity (e.g. Actor). The specialization of the *Group* concept allows to define all communication group (e.g. Commercial-Team, development-Team, etc.).The system designer will also define all possible roles that can be played by the defined entities. The last element that will be specialized is the relation *hasSession* in order to define the needed collaboration sessions for each group.
Therefore the domain ontology contains elements that inherit from these elements and some rules indicating how roles can communicate, within the specified sessions.
At runtime, the domain ontology is instantiated depending in the application context, thus providing knowledge base containing explicit and implicit collaboration aspects about participants, their roles, their group and about the needed sessions for each group.

Thanks to the OWL importation mechanism, both instances of GCM concepts and those of the domain ontology are regrouped into the same instance of this model. This instance represents a business view of the collaborative activities. A reasoning engine is used in order to make explicit the implicit knowledge contained in this instance. Rules trigger the instantiation of the generic meta-model enabling a semantic-driven adaptation that enables managing spontaneous sessions and detecting implicit potential collaboration situations.

### Messaging level

The architectural model produced in this level is a detailed deployment schema that contains the needed elements in order to implement collaboration sessions determined by the collaboration level. For this level, we have retained two communication paradigms: the Event Based Communication and the Peer-To-Peer communication. The EBC is based on the Publish/Subscribe pattern and it represents a well-established paradigm for interconnecting loosely coupled components and it provides a one-to-many or a many-to-many communication pattern. The main advantage of the EBC is the opportunity for better scalability than traditional client-server pattern through message caching and network-based routing. Moreover, publishers are loosely coupled to subscribers and they operate independently of each other. The Peer-To-Peer model allows to make some resources available in the network participants without a central coordination. Its structure is privileged for file sharing that represents an important activity in collaborative systems.

The produced models in this level are EBC-based graphs and Peer-To-Peer-based graphs that contain the specific needed elements for each paradigm. The EBC model contains a set of *event producers*, *event consumers* and *channel managers* connected with push and pull links. For every determined session, a CM is created in order to manage, store and deliver exchanged data flow between multiple producers and consumers. On the other hand, the Peer-To-Peer model contains peers, super-peer and pipes that enable a the communication and the file sharing between participants
These produced models are represented by graphs expressed in the GraphML language.
Since the produced model represents a detailed deployment descriptor, we provide a deployment service that effectively deploys elements on participants' devices. This deployment service takes a deployment descriptor as input in order to install, start and update the required components.

## 3.3.2 Refinement and selection between levels

### Application–Collaboration Refinement

Reasoning based on SWRL rules [45] is used in order to implement the application-collaboration refinement. SWRL rules are applied to an instance of the domain ontology that extends the GCM. They have the form: $a_1,\ldots,a_n \rightarrow b_1,\ldots,b_m$ where $a_1,\ldots,a_n$ is the body and $b_1,\ldots,b_m$ is the head of the rule. The terms $a_1,\ldots,a_n,b_1,\ldots,b_m$ are SWRL atoms which can be a concept assertion, a relation assertion or built-ins. The semantic of the rule is the following: the interpretation of the head holds only if the interpretation of the body holds.
The proposed Generic Collaboration Meta-model includes 6 generic rules[1] that express some relations and especially those which allow to infer deployment schema from the sessions presents in the domain ontology instance. However, these rules are not sufficient for complete implementation of the refinement from the domain ontology to the collaboration ontology, because it depends on the collaborative system. Therefore the application designer has to specify additional rules in the domain application model which contains a part of the refinement process. An example of these rules is detailed in the section 5.
The processing of the SWRL, which is done by a rule engine such Jess, produces an instance of the Generic Collaboration ontology represented by a collaboration graph expressed in the OWL language. This graph is translated into GraphML by XSLT transformation in order to be provided to the messaging level. The application-collaboration refinement produces a single collaboration model from a given application level model.

### Collaboration-Messaging Refinement and selection

As the collaboration level and the messaging level models are represented by graphs, graph grammar theories represent an appropriate formalism to handle the refinement process. Moreover, graph grammar is an intuitive and powerful way of handling complex transformations.
In order to refine a given collaboration architecture, a generative graph grammar [46] is used. In this graph grammar, non-terminal nodes are collaboration entities while terminals nodes are deployment entities. This refinement infers two sets of deployment graph: a set containing all possible EBC-based graphs where terminal nodes are EBC entities and another set containing all possible Peer-To-Peer-based graphs where terminal nodes are Peer-To-Peer entities.

[1] An example of these rules is available at: http://homepages.laas.fr/akamoun/sessions.owl#same_group

In order to apply transformation's rules of the proposed grammar, Graph Matching and Transformation Engine (GMTE) [46] is used.
In order to select the optimal architecture among those built by the refinement process, the generic selection procedure presented in the Figure 6 is used depending on several parameters. This function allows: (1) to discard configuration that cannot be deployed, (2) to select the best adapted configuration to the current context.
At first, this function computes the value assigned by the *ContextAdaptation()* function for each configuration. These values reflect the degree of adaptation of each configuration to the current network context. Well adapted architectures are those that respond to the global communications' needs. Our criterion of selection for this function is as follows: EBC-based architecture is deployed for open collaboration such as synchronous or asynchronous, point-to-point or multicast communications that may require a central server which manages exchanged data flows. Otherwise, a Peer-To-Peer architecture can be deployed for private communication such as sharing and exchanging produced artifact during the collaborative activities.
If the configuration cannot support the current context, its assigned value will be -1. Otherwise, a positive value will be assigned. The candidates having the highest positive value will be added to the set *S1*. If *S1* does not contain any configuration, then there is no adapted configuration that supports the current context. In that case, the select() function return an empty set. If *S1* contains one candidate, it will be returned. Otherwise, the indicated policies are used by the select() function in order to retain the optimal configuration.

1 Select (Policy)
2 {
3 Let $C_{n,p} \in \mathbb{C}_n$, $p \in \mathbb{N}$
4 Let $A$ denote the context attributes
5 Select $S1 = \{ C_{n-1,k} \in \mathbb{C}^p_{n-1}, k \in \mathbb{N}$ such that :
$Context\_Adaptation(C_{n-1,k}, A) >=$
$Context\_Adaptation(X, A) \forall X \in \mathbb{C}^p_{n-1} \}$
6 if card $(S1) \neq 1$
7 if Policy = Dispersion
8 Select $S2 = \{ C_{n-1,k} \in S1, k \in \mathbb{N}$ such that :
$Dispersion(C_{n-1,k}) >= Dispersion(X), \forall X \in S1 \}$
9 if Policy = Distance
10 Let $C_{n,p}$ and $C_{n,q} \in \mathbb{C}_n$, $p,q \in \mathbb{N}$
11 Let $C_{n-1,p}$ the current mapping
at level $n-1$ of $C_{n,p}$
12 Select $S2 = \{ C_{n-1,k} \in \mathbb{C}^q_{n-1}, k \in \mathbb{N}$ such that :
13 $Distance(C_{n-1,p},C_{n-1,k}) =<$
$Distance(C_{n-1,p},X), \forall X \in S1 \}$
14 if card $(S2) \neq 1$
15 Select any configuration from $S2$ }

**Figure 6. Selection**

In order to ensure the framework robustness, we have retained two policies; however we can add new policies depending on the system requirements. The first policy, called dispersion, uses a function *Dispersion()* (Figure 7) which computes the dispersion degree of components deployed on participants' nodes. The goal of this policy is to balance resources consumption in order to ensure the collaborative systems robustness. The selection procedure aims to maximize the dispersion of components (line 8, Figure 6). The second policy, called distance, uses a function *Distances()* (Figure 8) which computes the number of needed redeployments between two n-1-level configurations implementing a given n-level configuration. The selection procedure minimizes (line 13, Figure 6) the distance in order to discard useless redeployments.

Dispersion $(C_{1,q})$ {
let $node_i^q$ be a deployment node of $C_{1,q}$
disp = 0
For each $node_i^q$
if ∃ component deployed on $node_i^q$, then disp = disp+1
return disp
}

**Figure 7. Dispersion function**

Distance $(C_q, C_k)$ {
Let $node_i^q(component_j)$ be the deployment node of $component_j \in C_q$
dist = 0
for each $component_j \in C_q \cup C_k$
if $node_i^q(component_j) \neq node_i^k(component_j)$ then dist = dist+1
return distance
}

**Figure 8. Distance function**

Once an adaptive communication model is selected, the two proposed policies *Dispersion* (Figure 7) and *Distance* (Figure 8), indicated by the parameter *Policy* in the selection function, are used in order to select the best adaptive deployment graph.

### 3.3.3 Deployment service

In order to manage dynamic change in collaborative systems, deployment needs to be adaptive at run-time, so a dynamic deployment service is needed. Therefore the result graph produced by the messaging level is a low-level detailed deployment descriptor that is used by the deployment service in order to carry out the effective deployment of components needed for supporting collaborative activities. The system keeps the same configuration until the arrival of new context events. The deployment descriptor implements the low-level elements of the required architecture. For instance, it contains producers, consumers, channel managers and link if the required architecture is EBC. Once receiving this descriptor, the deployment service downloads, installs and starts the required components on each participant' device.
The deployment service architecture is detailed in the Figure 9. At the same server where the framework is running, we propose a *deploymentService* entity and a repository that contains the available components. Once receiving the deployment graph, the deployment service finds the required components for each participant's device. Afterwards, it connects to the evolved devices through the deployment agents (broken lines in the figure) informing them about all needed components and the set of remote components to which the deployment agent must be connected. If the required components don't exist, they will be downloaded from the repository through the HTTP protocol (dashed lines).

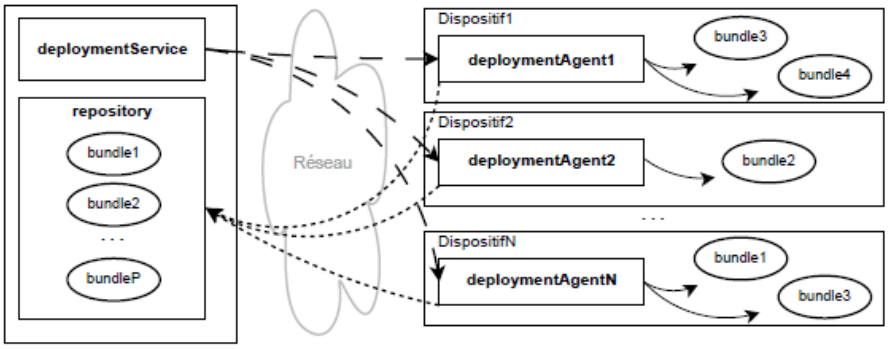


**Figure 9. Deployment service architecture**

The implementation of this deployment service is based on the *Open Services Gateway Initiative* (OSGi) technology [47]. Indeed, OSGi offers very promising functionalities such as dynamic code loading and execution. Besides, this specification offers a services management platform (with a dynamic register of services enabling the detection, the invocation, the addition, the cancellation of services at runtime). OSGI offers as well a complete and dynamic component model. In OSGI specification, components are called *bundles*.
Within this approach, deployable components are packaged as OSGi bundles that are handled by the proposed deployment service.

# 4 The Framework interoperability

In this section, we explain how to take advantage of the proposed framework in order to maintain the interoperability of enterprise systems in networked environment in the advent of their dynamic. We provide methods that enable the dynamic adaptation of such systems and the optimization of the product development process by managing the collaborative sessions during all development phases of a given product. We also address the discovery capabilities in order to enable the detection of new changes on the situations of collaboration within the systems' network. These changes trigger a notification to our communication framework which launches the adaptation process using the proposed knowledge representation technologies applied to a given domain model. The result of this adaptation is a messaging model that decides in what way should the network nodes react in order to enable the system to evolve for the new interoperable state.
In order to carry out the adaptation process during the collaborative activities, the collaborative process modeling requires three phases (Figure 10):

## Phase 1

In the first phase, the process designers define a process model which contains specific concepts such as groups, roles, activities, relationship, constraints, automatic collaborative sessions for each group, etc. Application-specific SWRL rules are also defined and will be used by the adaptation process in order to determine to what session should belong a participant having a given role or task. This process model is a specialization of the Generic Collaboration Meta-model detailed above (in the Figure 5). However, it can be provided as an UML diagram, so it must be transformed to a domain ontology (OWL file) via XSLT transformation.

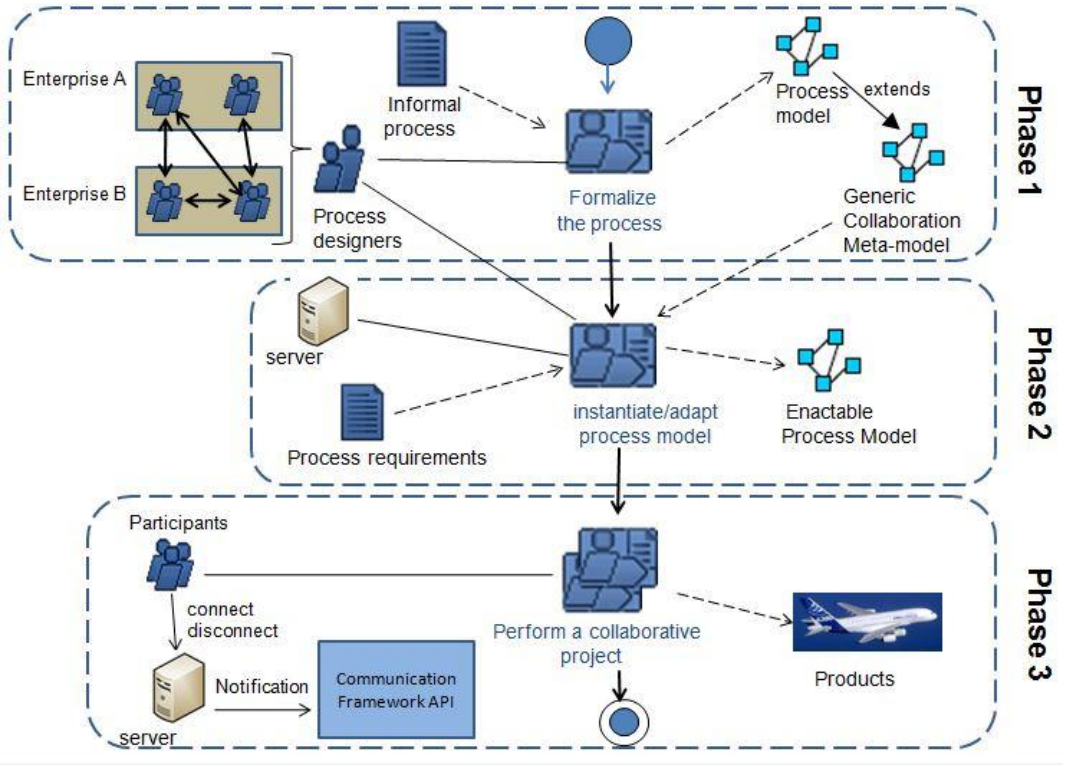


**Figure 10. Collaborative process modeling**

The process designers formalize the process depending on the needed structure that enable inter and intra-enterprise collaboration. For example, they formalize the collaboration between participants having a common experience, performing common tasks and belonging to one or more enterprises.
The main advantage of this architecture is that the process model can be updated at runtime without performing actions on the system's code implementation. This update may affect sessions' organization, groups' definition, possible roles; etc. For example, the process designers can add or remove one or more collaborative session of a given groups. Therefore, during the adaptation process, the system's server has a permanent HTTP access to the defined process model which can be updated.

## Phase 2

Once the process model is detected, the system's server instantiates the process model and allocates to the project all resources (project plan). This instance represents the initialization of the application descriptor that will be the entry of the adaptation process. This phase produces an enactable process model that contains the defined specific concepts and relations defined in the process model such as all potential roles that may be played by participants, all possible groups that may represents sub-projects and mainly the defined collaboration sessions for all groups.

## Phase 3

Once the process model is defined by the process designers and initialized by the system's server, participants will be assisted in performing the collaborative project according to the enactable process model, to their roles, to the teams' organization and to the defined collaboration scenarii.
Therefore, the system's server should enable the needed adaptation after each actions (such as change involving connection, disconnection , changing roles, joining or leaving a given working group) performed by each participants. Hence, the process model which has been instantiated and initialized in the phase 2 must be updated in order to represent all the performed actions. In our communication framework, this model represents the application descriptor which is the entry of each adaptation process.
In order to create dynamically the required application descriptors that represent the current collaboration state, services are proposed to the system's server enabling its communication with the proposed framework. The proposed services are as follows:

- **Connect (user Id, device IP, list Roles, list Groups)**: A user connects to system. The system core API should call this function to notify the communication framework (FADYRCOS). Each role will be added to all groups.

- **Quit (userId)**: A user quits the system. The system core API should call this function to notify the communication framework. The participant and all his roles will be removed from the application descriptor.
- **AddToGroup (userId, group)**: A connected user is added to a group on the system server. The core API should call this function to notify the communication framework.
- **RemoveFromGroup (userId, group)**: A connected user is removed from a workgroup on the system server. The core API should call this function to notify the communication framework.
- **AddRole (userId, role)**: A new role is added to a connected user on the system. The core API should call this function to notify the communication framework. Then the new role is added to all groups belonged by the user.
- **RemoveRole (userId, role)**: A role is removed from a user connected on the system. The core API should call this function to notify the communication framework.
- **ChangeRole (userId, previous Role, new Role)**: A connected user changes one of his roles.

In order to determine if the requested service has a potential conflict, a consistency check mechanism is used. Indeed, all these services depend on the process model which defines actors, groups, roles, etc. For instance, if a participant request to have two roles *r1* and *r2* and to join a working group *g1*, the communication framework refers to the defined process model in order to check if the two roles *r1* and *r2* and the group *g1* are already exist. If his request contains a conflict, the system will be informed by the communication framework.
Once the application descriptor is created using the proposed services, the communication framework carry into effect the adaptation process. Firstly, a collaboration schema that responds to the high level needs is generated by the collaboration level through reasoning using SWRL rules defined both by the Generic Collaboration Meta-model and the process model. After that, the Graph Matching and Transformation Engine (GMTE) infers a set messaging graph representing all possible communication models such as EBC and Per-To-Peer models. Then, the communication framework selects the well adapted messaging graph that ensures the needed communication between participants performing the collaborative activities. Finally, the deployment service takes charge of the effective deployment of components on participants' device.

Sessions on demand API are also defined in order to enable spontaneous sessions between participants. The proposed API provides the following services:

- **createSession(String userId, String name)**: A user creates a new communication session.
- **closeSession(String userId, String name)**: A user closes a communication session. Only the creator should be allowed to close this kind of sessions.
- **joinSession(String userId, String name)**: invites a user to join an open communication session.
- **quitSession(String userId, String name)**: removes a user from an open communication session.

# 5 Case study

The software prototype environment has been implemented in Java. In this section, we consider a software engineering project concerned with the development of complex software products. The ultimate goal is to ease the development of such products by assisting engineers, whose are organized within different teams, in their collaborative work. The present case study is implemented in the context of the ANR Galaxy project[2].
We detail the three required phases:
**Phase 1:** In this phase, the process designer of the Galaxy project defines the process model which contains the groups, the possible roles and collaboration sessions for each group. Figure 11 describes a proposed domain ontology (expressed by an OWL file) that represents the collaboration aspect of the defined process model.
In this figure, concepts and relations of the Generic Collaboration Meta-model are represented with broken lines, while specific concepts and relations are represented in white. The *Node* concept is specialized into the *Actor* concept which models a galaxy participant. For each actor, a set of possible roles, which specialize the *Role* concept, are defined:

- Designer: a designer can be:
  - Simple designer: designs and writes architecture description models.
  - Designers' leader: collaborates with the simple designers in order to produce the detailed architecture of the system.
- Integration manager: decides when features and fixes are ready for production, and merges them while being sure that the result is functional and reasonably bug-free.
- Developer: a developer can be:
  - Code developer: writes codes.
  - Test developer: writes integration tests for the system under development.

[2] http://www.irit.fr/GALAXY

- Deployment manager: deploys the project to production, monitors execution, and reports errors back to developers.

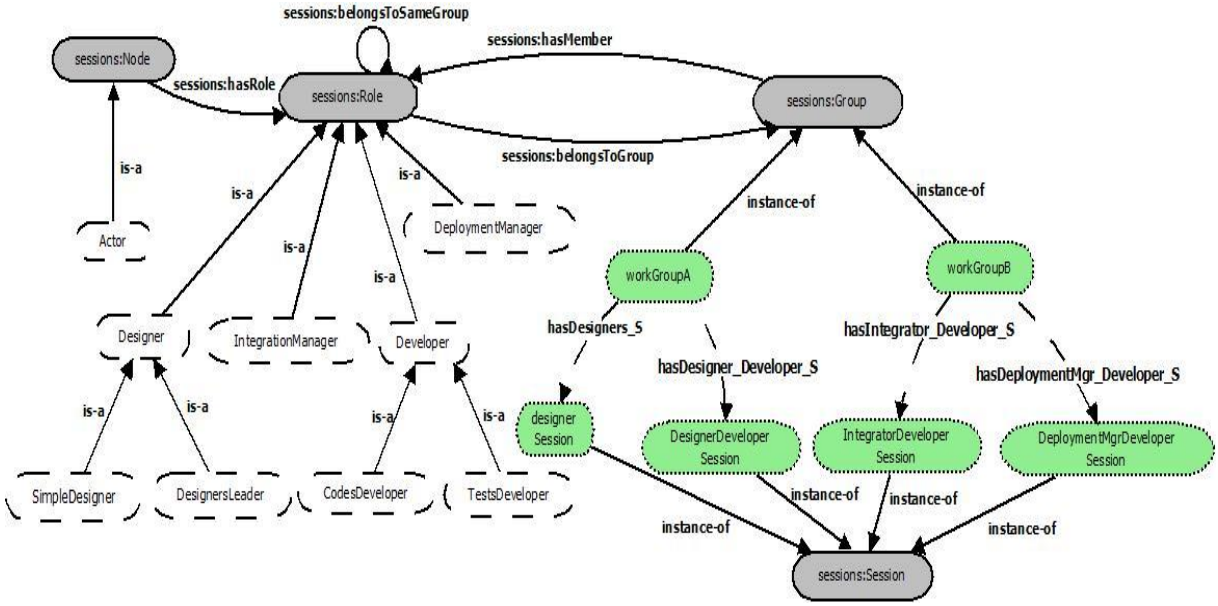


**Figure 11. Domain ontology**

Having one or more roles, engineers can belong to one or more working groups. The defined working groups for this project, *workGroupA* and *workGroupB*, are two instances of the *Group* concept of GCM (all instances are represented with dashed lines). For each working group are defined the associated collaboration sessions. Engineers who belong to the *workGroupA* are organized within two sessions: *DesignerSession* which regroups all project designers and *DesignerDeveloperSession* which regroups designers and developers. On the other hand, engineers who belong to the *workGroupB* are organized within the two sessions: *integratorDeveloperSession* which regroups the integration manager and developers when integrating the developed codes and *DeploymentMgrDeveloperSession* which regroups the deployment manager and the developers when a bug is found. These collaboration sessions are instances of the *Session* concept of the GCM. In order to assign sessions to the groups, the *hasSession* relation of GCM is specialized into four relations: *hasDesigner_S*, *hasDesigner_Developer_S*, *hasIntegrator_Developer_S* and *hasDeploymentMgr_Developer_S*.

In addition to the specific concept and relations, the domain ontology which describes the process model must also contain the application-specific SWRL rules that allow rule engines to create sessions and data flows between engineers according to their roles. As an example, let us consider the following rule (Figure 12).

This rule states that, whenever a simple designer and a designers leader are connected to the system and they belong to the same working group and they need to communicate in audio, an audio flow will be created having the node of the simple designer as source and the node of the designer leader as destination and another audio flow will be created having the node of the simple designer as destination and the node of the designer leader as source.

Similar rules are defined to handle other data flows enabling the communication between other members.

This domain ontology can be eventually updated at runtime by the process designer. Indeed, other sessions can be added for a given working group, other groups or roles can also be defined during a software product development.

$$SimpleDesiger(?sd) \wedge sessions:Node(?nsd) \wedge$$
$$sessions:HasRole(?nsd,?sd) \wedge DesignersLeader(?dl) \wedge$$
$$sessions:Node(?ndl) \wedge sessions:hasRole(?ndl,?dl) \wedge$$
$$sessions:belongsToSameGroup(?sd,?dl) \wedge$$
$$differentFrom(?sd,?dl) \wedge sessions:belongsToGroup(?dl,?t) \wedge$$
$$hasDesigners\_S(?t,?s) \wedge swrlx:createOWLThing(?af1,?sd)$$
$$\wedge swrlx:createOWLThing(?af2,?sd) \rightarrow$$
$$sessions:AudioFlow(?af1) \wedge sessions:Source(?af1,?ndl) \wedge$$
$$sessions:Destination(?af1,?nsd) \wedge$$
$$sessions:belongsToSession(?af1,?s)$$
$$sessions:AudioFlow(?af2) \wedge sessions:Source(?af2,?nsd) \wedge$$
$$sessions:Destination(?af2,?ndl) \wedge$$
$$sessions:belongsToSession(?af2,?s)$$

**Figure 12. An application-specific SWRL rule**

**Phase 2**: After the process model definition, an enactable process model is created instantiating the defined process model. For the present case study, this instance contains the two working groups and their associated collaboration sessions.

**Phase 3:** The participants-context awareness is taken into account in this phase during which the adaptation process is needed after each action. Let us consider the following scenario (Figure 13):

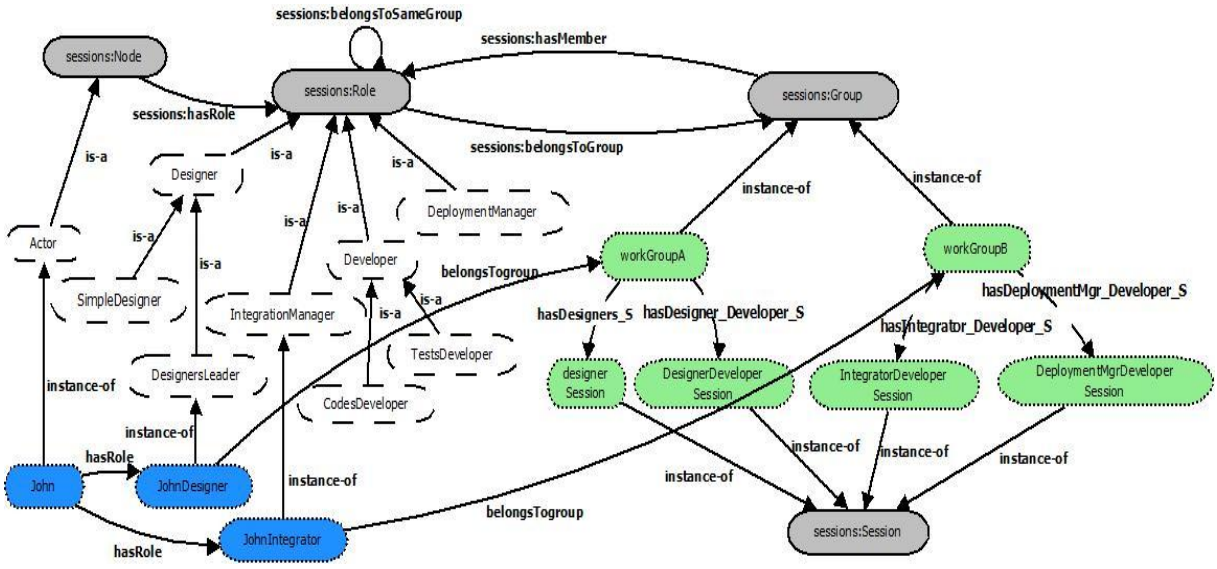


**Figure 13. Scenario**

John is connecting to the Galaxy system and he has both the designer role and the integrator role. The galaxy core API will update the instance of the domain ontology as above using the proposed API and especially using the service **Connect**. Hence, the process adaptation infers a collaboration schema through SWRL rule execution allowing John to collaborate within the *DesignerSession* and *DesignerDeveloperSession* of the *workGroupA* as a designer and the *IntegrationDeveloperSession* of the *workgroupB* as an integrator. Then it produces an adaptive deployment graph that models the needed components to ensure his collaboration. After the automatic deployment of components, John will be able to collaborate with all designers of *workGroupA*, on one hand, and with developers and integrators of the *workGroupB*, on the other hand.

Whenever the instance that represents the new changes is created, the proposed communication API checks the consistency of the required action. For instance, if the process designer removes the *workGroupA*, then the Galaxy server will be informed that this group does not exist.

# 6 Conclusion

In this paper, a semantic-driven, adaptive architecture has been developed to manage messaging for collaborative engineering activities within networked enterprises. This architecture is based on a Generic Collaboration Meta-model (GCM) which supports the implementation of session management services. GCM is a domain-independent ontology that can be extended for modeling collaboration knowledge in different application domains.
This work provides a framework for session management and adaptive component deployment. Knowledge representation technologies, such as ontologies and SWRL rules, are used to ensure awareness of collaboration situations and to decide when sessions have to be adapted.
The proposed architecture provides a generic level of collaboration. It enables the integration of different applications and deals with application-specific information as sub models that are plugged into the framework.
The developed prototype system establishes an open and scalable architecture to support inter and intra-enterprise collaboration. Indeed, a process designer defines a process model which represents the participants, their roles, their groups and the associated sessions. The process model is instantiated using the proposed services that are able to check the consistency of the requested actions. The main benefit from this approach is ensuring the continuity of the communication services when changes occur in the process model. The framework supports a dynamic component deployment using the deployment service which uses the produced deployment graph. The type of deployed components depends both on the selected communication model and on the needed communication mode such as synchronous and asynchronous communication. We can conclude that semantic representation and Web semantic technologies are useful to enable dynamic reconfiguration to ensure interoperability.
The work presented is being tested on real case in the context of Galaxy project.
Future work will consists in performing generic security functionalities and access control, at the communication framework level, which rely on the involved collaborative system.

**Acknowledgement**

This article was funded partially by the ANR Project Galaxy and the IST IP IMAGINE.

**Appendix. List for acronyms in this paper**

| | |
|---|---|
| CSCW | Computer-Supported Collaborative Work |
| FADYRCOS | Framework for dynamic reconfiguration of networked collaborative systems |
| GCM | Generic Collaboration Meta-model |
| MDA | Model Driven Architecture |
| OWL | Web Ontology Language |
| EBC | Event Based Communication |
| HTTP | HyperText Transfer Protocol |
| TCP | Transmission Control Protocol |
| GMTE | Graph Matching and Transformation Engine |
| OSGi | Open Services Gateway Initiative |
| XML | Extensible Markup Language |
| SWRL | Semantic Web Rule Language |
| UML | Unified Modeling Language |
| XSLT | Extensible Stylesheet Language Transformations |

**References**

[1] P.H. Carstensen, K. Schmidt, Computer supported cooperative work: New challenges to systems design, in: K. Itoh (Ed.), Handbook of Human Factors, 1999, pp. 619-636.
[2] C.A. Ellis, S.J. Gibbs, G.L. Rein, GROUPWARE - SOME ISSUES AND EXPERIENCES, Communications of the Acm, 34 (1) (1991) 38-58.
[3] M. Nagasundaram, Ambiguity in Human Communication and the Design of Computer Mediated Communication Systems, in, Proceedings of the Twenty-Fifth International Conference on Systems Sciences, Hawaii, 1992.

[4] J. Grudin, Computer-Supported Cooperative Work: History and Focus, Computer, 27 (5) (1994) 19-26.
[5] A. KARSENTY, Le collecticiel: de l'interaction homme-machine à la communication homme-machine-homme, in: Lavoisier (Ed.), Technique et Science Informatique (TSI), Paris, FRANCE, 1994, pp. 105-127.
[6] M. Lewkowicz, E. Soulier, N. Gauducheau, Collecticiels pour la construction collective du sens - définition, principes de conception, exemple, in: E. Soulier (Ed.), Le storytelling : concepts, outils et applications, Hermès, 2006, pp. 293-314.
[7] Y. Laurillau, L. Nigay, Clover architecture for groupware, in, Proceedings of the 2002 ACM conference on Computer supported cooperative work, ACM, New Orleans, Louisiana, USA, 2002, pp. 236-245.
[8] L.G. Yesilbas, M. Lombard, Towards a knowledge repository for collaborative design process: focus on conflict management, Comput. Ind., 55 (3) (2004) 335-350.
[9] P.N. Robillard, P. d'Astous, F. Détienne, W. Visser, Measuring cognitive activities in software engineering, in, Proceedings of the 20th international conference on Software engineering, IEEE Computer Society, Kyoto, Japan, 1998, pp. 292-299.
[10] P. Dourish, V. Bellotti, Awareness and coordination in shared workspaces, in, Proceedings of the 1992 ACM conference on Computer-supported cooperative work, ACM, Toronto, Ontario, Canada, 1992, pp. 107-114.
[11] J.D. Palmer, N.A. Fields, P.L. Brouse, Multigroup Decision-Support Systems in CSCW, Computer, 27 (5) (1994) 67-72.
[12] D. Te''eni, Review: a cognitive-affective model of organizational communication for designing IT, MIS Q., 25 (2) (2001) 251-312.
[13] M. Lewkowicz, F. Wijnhoven, A. Draghici, Misunderstandings in Global Virtual Engineering Teams: Definitions, Causes, and Guidelines for Knowledge Sharing and Interaction, in: A. Bernard, S. Tichkiewitch (Eds.), Methods and Tools for Effective Knowledge Life-Cycle-Management, Springer, 2008, pp. 145-157.
[14] W.K. Edwards, Putting computing in context: An infrastructure to support extensible context-enhanced collaborative applications, ACM Trans. Comput.-Hum. Interact., 12 (4) (2005) 446-474.
[15] V. Baudin, K. Drira, T. Villemur, S. Tazi, A model-driven approach for synchronous dynamic collaborative e-learning, in: C. Ghaoui (Ed.), E-Education applications: human factors and innovative approaches, 2004, pp. 44-65.
[16] W.K. Edwards, Session management for collaborative applications, in, Proceedings of the 1994 ACM conference on Computer supported cooperative work, ACM, Chapel Hill, North Carolina, United States, 1994, pp. 323-330.
[17] G. Texier, N. Plouzeau, Automatic Management of Sessions in Shared Spaces, J. Supercomput., 24 (2) (2003) 173-181.
[18] M. Rusinkiewicz, W. Klas, T. Tesch, J. Wäsch, P. Muth, Towards a Cooperative Transaction Model - The Cooperative Activity Model, in, Proceedings of the 21th International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc., 1995, pp. 194-205.
[19] L. Beca, G. Cheng, G.C. Fox, T. Jurga, K. Olszewski, M. Podgorny, P. Sokolowski, T. Stachowiak, K. Walczak, Tango – a Collaborative Environment for the World-Wide Web, in, Northeast Parallel Architectures Center, Syracuse University, New York, 1997.
[20] A. Chanbert, E. Grossman, L. Jackson, S. Pietrovicz, "NCSA Habanero- Synchronous collaborative framework and environment", in, Software Development Division at the National Center for Supercomputing Applications, white paper, 1998.
[21] EMSL DOE2000: Real-time Collaboration Management Project Summary, in, 1999.
[22] I. Marsic, Real-Time Collaboration in Heterogeneous Computing Environments, in, Proceedings of the The International Conference on Information Technology: Coding and Computing (ITCC'00), IEEE Computer Society, 2000, pp. 222.

[23] T. Holvoet, Y. Berbers, Reflective programmable coordination media, in: H.R. Arabnia (Ed.), Pdpta'2001: Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas, Nevada, USA, 2001, pp. 1236-1242.
[24] S.R. Fussell, R.E. Kraut, J. Siegel, Coordination of communication: effects of shared visual context on collaborative work, in, Proceedings of the 2000 ACM conference on Computer supported cooperative work, ACM, Philadelphia, Pennsylvania, United States, 2000, pp. 21-30.
[25] K. Gutzmann, Access Control and Session Management in the HTTP Environment, IEEE Internet Computing, 5 (1) (2001) 26-35.
[26] L.M.R. Peralta, T. Villemur, K. Drira, An XML on-line session model based on graphs for synchronous cooperative groups, in, Proceedings of the international Conference on Parallel and Distributed Processing Techniques and Applications PDPTA'2001, Las Vegas, Nevada, USA, 2001, pp. 1257-1263.
[27] M. Rodriguez Peralta Laura, SERVICE DE GESTION DE SESSION ORIENTE MODELE POUR DES GROUPES COLLABORATIFS SYNCHRONES, in, Institut National Polytechnique de Toulouse, 2003, pp. 166.
[28] C.A. Vissers, M.M. Lankhorst, R.J. Slagter, Reference Models for Advanced e-services, in, The 3rd IFIP Conference on e-Commerce, e-Business and e-Government, Guarujá – São Paulo – Brazil, pp. 21-24.
[29] H.-P. Dommel, J.j. Garcia-Luna-Aceves, Group Coordination Support for Synchronous Internet Collaboration, IEEE Internet Computing, 3 (2) (1999) 74-80.
[30] C.Y. Jang, C. Steinfield, B. Pfaff, Supporting awareness among virtual teams in a web-based collaborative system: the teamSCOPE system, SIGGROUP Bull., 21 (3) (2000) 28-34.
[31] C.H. Ganoe, J.P. Somervell, D.C. Neale, P.L. Isenhour, J.M. Carroll, M.B. Rosson, D.S. McCrickard, Classroom BRIDGE: using collaborative public and desktop timelines to support activity awareness, in, Proceedings of the 16th annual ACM symposium on User interface software and technology, ACM, Vancouver, Canada, 2003, pp. 21-30.
[32] E.L. Morse, M.P. Steves, CollabLogger: A Tool for Visualizing Groups at Work, in, Proceedings of the 9th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, IEEE Computer Society, 2000, pp. 104-109.
[33] H. Guyennet, J. Lapayre, E. Garcia, in, Proceedings of International Conference on information Systems Analysis and Synthesis, Orlando, USA, 1999, pp. 442-447.
[34] N. Gu, J. Xu, X. Wu, J. Yang, W. Ye, Ontology based semantic conflicts resolution in collaborative editing of design documents, Adv. Eng. Inform., 19 (2) (2005) 103-111.
[35] Z. Yao, S. Liu, L. Han, Y.V.R. Reddy, J. Yu, Y. Liu, C. Zhang, Z. Zheng, An ontology based workflow centric collaboration system, in, Proceedings of the 10th international conference on Computer supported cooperative work in design III, Springer-Verlag, Nanjing, China, 2007, pp. 689-698.
[36] E. Andonoff, W. Bouaziz, C. Hanachi, A protocol ontology for inter-organizational workflow coordination, in, Proceedings of the 11th East European conference on Advances in databases and information systems, Springer-Verlag, Varna, Bulgaria, 2007, pp. 28-40.
[37] J.L. Garrido, M. Noguera, M. Gonzàlez, M.V. Hurtado, M.L. Rodriguez, Definition and use of Computation Independent Models in an MDA-based groupware development process, Sci. Comput. Program., 66 (1) (2007) 25-43.
[38] K. Tomingas, M. Luts, Semantic Interoperability Framework for EstonianPublic Sector's E-Services Integration, in, Proceedings of the First Workshop on Ontology Repositories and Editors for the Semantic Web (ORES2010), 2010, pp. 96-100.
[39] R. Schollmeier, [16] A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications, in, Proceedings of the First International Conference on Peer-to-Peer Computing, IEEE Computer Society, 2001, pp. 101.
[40] A.D. Birrell, B.J. Nelson, Implementing remote procedure calls, ACM Trans. Comput. Syst., 2 (1) (1984) 39-59.
[41] S. Vinoski, Introduction to CORBA (tutorial session), in, Proceedings of the 22nd international conference on Software engineering, ACM, Limerick, Ireland, 2000, pp. 822.

[42] R. Meier, V. Cahill, Taxonomy of Distributed Event-Based Programming Systems, Comput. J., 48 (5) (2005) 602-626.
[43] U. Brandes, M. Eiglsperger, I. Herman, M. Himsolt, M.S. Marshall, GraphML Progress Report, in, Graph Drawing, 2001, pp. 501-512.
[44] M.K. Smith, C. Welty, D.L. McGuinness, OWL Web ontology Langage Guide, W3C Recommendation, in, 2004.
[45] I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C, Member Submission in, 2004.
[46] I.B. Rodriguez, K. Drira, C. Chassot, K. Guennoun, M. Jmaiel, A rule-driven approach for architectural self adaptation in collaborative activities using graph grammars, Int. J. Autonomic Comput., 1 (3) (2010) 226-245.
[47] OSGi Alliance. .OSGi Service Platform Release 4, in, 2007.