



International Journal of Cooperative Information Systems
© World Scientific Publishing Company

TAILORABLE GROUPWARE DESIGN BASED ON THE 3C MODEL

NADER CHEAIB*

*University of Evry Val d'Essonne, IBISC Laboratory
40 Pelvoux Street, 91020, Evry, France*

SAMIR OTMANE

*University of Evry Val d'Essonne, IBISC Laboratory
40 Pelvoux Street, 91020, Evry, France*

MALIK MALLEM

*University of Evry Val d'Essonne, IBISC Laboratory
40 Pelvoux Street, 91020, Evry, France*

Received (20 03 2011)

Accepted (23 11 2011)

In this paper, we propose a software architecture based on Web services and Software agents for groupware tailorability. Through our literature study, we realize that the property of tailorability has a significant impact on designing collaborative applications. Although online applications in the recent years have been growing exponentially, on-line collaborative work between users is often supported by software applications that provide static basic functionalities, mostly centered on communication tools (text, audio and video). Hence, adding more sophisticated tools for enriching the collaborative experience, as for example, an integrated environment for task coordination and production requires manually coding them into the application, which requires a significant effort in order to adapt the system to the real needs of users. In a collaborative context, the application designers are not able to predict all users' needs at design time. To remedy this problem, we propose a tailorable groupware architecture that enables the dynamic integration/composition of services into the collaborative application, gaining both in time and performance. Our work is based on the 3C functional model by Ellis that decomposes collaboration between users into communication, coordination and cooperation spaces. Through our research, we realized that Web services are powerful distributed components offering the desired tools to adapt a groupware to the real needs of users. In this paper, we propose a collaboration protocol based on Web services between machines over the network in order to exchange common services. Based on this protocol, we propose our groupware architecture, *U3D*, that introduces tailorability in collaboration applications.

Keywords: CSCW; Groupware Design; Tailorability

*nader.cheaib@ibisc.fr

2 *Nader Cheaib, Samir Otmane, Malik Mallem*

1. Introduction

The aim of Computer Supported Cooperative Work (CSCW) is to find ways in which applications should improve collaborative work between individuals¹⁹. Hence, there is a great need to address constraints related to the lack of flexibility and rigidity of current collaborative systems, through the adoption of adequate solutions to implement a better collaboration, depending on users' needs and the task that is being done^{32, 21, 22}. The emergence of collaborative work over the Internet was a solution to the high complexity of systems and the technical difficulties that could arise from their use, as users, geographically distributed want more and more to work together on a single task, but using rigid and often incompatible applications that may lead to interoperability problems⁴⁷. For the authors in¹⁷, groupware invention is a challenge, as the nature of collaborative work continually changes as a consequence of changing work needs, but also as a consequence of how the systems themselves tend to change work relationships and processes. As a consequence, the author argues that systems must themselves adapt to reflect the unpredictable differences between the requirements of support for collaborative work during analysis and the actual requirements. Hence research about tailorability in groupware design originated from the gap between design and use of collaborative systems²³. Making the system and the services offered within tailorable by users is an essential and ongoing research field that needs much attention to yet be concrete. For this reason, tailorability has shown to be an essential property that should be taken in consideration, as it offers the possibility to adapt the application based on users' needs and not the other way around⁹.

1.1. *Motivation and Objectives*

With the apparition and advancement of Internet technologies and the Web 2.0⁴⁶, universal interoperability between collaborative applications is becoming a reality, while geographically distributed people are highlighting the flexibility of cooperation by exchanging universally accessible services. On one hand, web services have become one of the most important architectures for the cooperation of heterogeneous systems and have ushered in a new era of software design⁴⁶. While computer networks have been able to pass data between different hosts, it was the emergence of web services that allowed these remote hosts to offer services in a more flexible and dynamic way. On the other hand, the autonomy and intelligence of multi-agent systems (MAS) have considerably increased software automation of some operational areas^{5, 39}. An important benefit of software agents is their ability to help, through collaboration, human beings and software logistics, while their concept is even older than web services and has been used successfully for the implementation of distributed applications^{53, 41}.

In this paper, we present an innovative approach for a tailorable groupware architecture integrating web services with software agents. The idea is to exploit

agents' proactive interaction capabilities in order to improve the behavior of web services, hence creating a cohesive entity that reinforces the individual advantages of these two technologies in the context of tailorable groupware design. Furthermore, we aim to propose a solution to some constraints in groupware design such as:

- Openess, where services/components are dynamically integrated into the collaborative system without stopping the collaboration process, neither manually coding, compiling and reexecuting the application.
- Adaptability, in order to be able to generate new behaviors relying on services/components that already exist in the system (composition).
- Interoperability, especially when users are using incompatible or heterogeneous applications.

The groupware architecture proposed in this paper enables the integration/composition of services in a collaborative context, hence creating new functionalities in order to enhance the collaboration between users. Furthermore, the groupware architecture proposed will insure interoperability between system's components.

1.2. *Collaboration and the 3C model*

In order to further understand the concept of collaboration, we base our work on the 3C functional model proposed by Ellis⁴, shown in Figure 1. According to this functional model, a groupware system covers three domain specific functions, production/cooperation, communication and coordination:

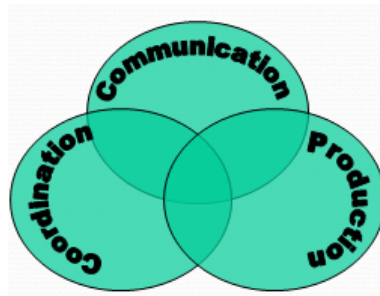


Fig. 1. The 3C Model by Ellis

The production space designates the objects resulting from the activity of the group (ex: word document, paint etc.). For Ellis, this production space is concerned with the result of common tasks to be achieved, and it is the space where the production takes place. The coordination space defines the actors and their social structure, as well as different tasks to be accomplished in order to produce objects in the production space. Ellis eventually completed the model with the commu-

4 *Nader Cheaib, Samir Otmane, Malik Mallem*

nication space that offers to actors in the coordination space means to exchange information in which the semantics concern exclusively the actor, and where the system only acts as a messenger. In this article, we will use this decomposition of groupware's functionalities in order to introduce a collaborative architecture supporting the functional decomposition of services that can be present in a groupware system.

In fact, there exist some work in the literature that make use of the 3C model in order to construe collaborative applications^{48,6,49,50}. The authors in⁴⁸ affirm that the way people connect and communicate with each other has changed through the change of the society over the years. According to the authors, the understanding of communication has transformed from being vertical, where orders are passed from above and reports are sent up the line, to a peer-to-peer paradigm where communication, coordination and cooperation predominate. This is due to the fact that command and control paradigm is losing effectiveness in the society. People are increasingly using tools and applications with no specific or centralized source that issues orders, but where people are collaboratively coordinating and dividing tasks between them, and eventually taking group decisions.

According to the authors in⁴⁹, the 3C model can also help evaluators focus their attention on the communication, coordination and cooperation aspects of the application in order to detect usability problems. Also, the relationship among the 3Cs of the model can be used as a guidance to analyze a groupware application domain. For example, a chat tool can be seen as a communication tool that requires coordination (access policies) and cooperation (registration and sharing). In our work, we use the 3C model in order to define the three main aspects of a collaborative application. We affirm that users need to pass through a communication phase, then a coordination phase and eventually a cooperation or production phase. Of course, users can communicate without coordinating when they share informal discussions, not specifically related to the task at hand. However, coordination cannot be realized without communication, and cooperation cannot be realized without coordination. Hence, an optimal collaboration pattern is achieved when the collaborative process is initiated by communicating, and ends by a concrete realization of the task at hand.

2. Groupware Tailorability- related work

As mentioned earlier, geographically distributed users are gradually leaning to collaborate with each other (through online games, shopping, project management, etc.), while the use of rigid and incompatible applications lead to interoperability problems^{34,45}. In fact, the need for flexible software is treated well in the areas of software engineering³². Motivated by the need to be more effective in software development, work has been conducted for better reuse of code and increased ex-

exploitation of software architecture (Object-oriented, Component-oriented, and more recently Service-oriented programming). In the area of CSCW, the motivation for flexibility is different: the software must be flexible enough to adapt to new work situations in the context of its use^{30, 21}. Thus, users should be able to adapt their applications according to their pace and work methods. However, some methods that have been developed for the evolution of software engineering, does not consider the role of the end user in software design. As a result, research on groupware tailorability is created as a result of the gap between the design and use of collaborative systems. Tailorability offers to users the possibility to adapt the application according to their needs, and not the contrary. Several authors have studied the concept of groupware tailorability^(19, 20, 21, 22, 23, 24, 25, 20, 17, 26). However, an exact definition of the term is still a subject of growing research. Given that several approaches can be adopted to achieve tailorability, the definition and concepts vary from one approach to another.

2.1. Definition

The study of the impact of the humanities' field on CSCW shows that tailorability is a fundamental property that should be taken into account when creating collaborative systems^{30, 7}. The authors in²⁶ define a tailorable application as a system that can be properly adapted to the changes and diversity of needs. The authors in²⁷ argue that tailorability is the capacity of an information system to enable a user to adjust the application to his/her personal needs, or the task that is being done. The authors in²⁹ emphasizes that a tailorable application is both usable and modifiable by its users, and the activity of its redefinition is itself an aspect of its use. Obviously, tailorability is a crucial property in groupware, but the lingering question is how to implement tailorability, particularly for end users who are not necessarily computer experts. In fact, the unpredictability of needs that will emerge during collaborative work makes it necessary for software systems to support a work that is expressed, for example, by the collaborative situation that arises, the specificity of the task at hand and the parameters within which the collaboration takes place (geographical distribution, software and hardware infrastructure, network connections). The authors in^{7, 30} break tailorability into three levels:

- **Customization:** achievable by a selection process through a set of predefined configuration options. For example, changing the appearance of an object, or editing the values of its attributes by choosing from a list of possible values. One of the problems here is that all tuning options should be provided at the time of the application's design, which limits its potential.
- **Integration:** involves linking predefined components within an application, or between disjoint applications. The integration goes beyond customization, allowing users to add functionalities to existing application code with-

out having access to the underlying implementation.

- **Extension:** corresponds to a change at the implementation level of the application, for example, by adding code to the existing one. In fact, this poses strong constraints on the APIs (Application Programming Interface), as the application should be flexible enough and easily manageable to be operated by end users, without compromising its effectiveness.

For the authors in ²⁸, tailorability does not necessarily mean the possibility of redefining the application in its own execution. However, it has often been identified as real time tailorability is better situated to support cooperative activities, where users can adapt the groupware without interrupting their activities.

2.2. Approaches for groupware tailorability

There exist several approaches that introduce tailorability in CSCW systems that received much attention in the literature. However, most of these approaches only deal with specific areas such as support for synchronous groupware, work processes ("workflow") and collaborative tasks. The author in ²⁸ justifies that tailorability has a theoretical basis for understanding the fundamental properties of the human activity. The author offers a set of properties in the construction of groupware, one of which is a fundamental property called reflexivity of the activity theory. Indeed, the activity theory shows that a CSCW system must be sufficiently flexible to be completely redefined during its own execution, where the only constants are the basic structure of the activity, as well as the basic mechanisms that make it survive. The lessons learned are put in practice in the CooLDA platform ⁵⁹ that aims at understanding the concept of co-evolution, which is a more comprehensive approach for tailorability. The authors in ²⁰ focused on defining the different behavior of systems in specific fields of validity. Hence, they developed an approach based on a concept called declarative tailorability, which shows how inconsistencies resulting from contradictory declarations can be processed either automatically or by involving the affected users. The authors in ⁶¹ introduce the concept of "tailorability to the extreme", which results from the extension of the sets of functions present in a system with new modules. An example is to allow a user to download modules from the Internet and plug them directly into their system (plugins, widgets etc.). Thus, interoperability standards between modules produced from different vendors are now essential and must describe the modules' type, their communication means as well as the services they provide. The authors in ⁶⁰ propose the concept of message enriching within the components of a system, which gives every user the means to attach information (metadata) to all messages circulating in the system. To the authors in ²⁵, a groupware must take in consideration a multitude of concerns, which are defined by the interests that relate to the system's development. They argue that the separation of concerns, which is the process of separating an application into distinct features that do not overlap in functionality, promotes tailorability in

a collaborative system.

2.3. Objectives and challenges of tailorable systems

Tailorable systems are different than systems that we call adaptive³¹. While tailorable systems allow users to have control over the modification process of the application, adaptive systems build what we call a user-model, where they automatically change the behavior of the system by initiating, proposing and executing possible changes. However, such system can not necessarily anticipate the needs of users in order to generate the appropriate changes. For this reason, research on tailorable systems must necessarily increase the control over the modification of the system by end users. Tailorability becomes the extension or modification of an application by creating persistent artifacts, or having to choose between several anticipated behaviors, or even creating a new behavior from the system's components. For the authors in³¹, tailorability is a human and technical art in order to change the functionalities of the technology while being executed. One of the reasons that software should be tailorable is the complexity of predicting users' needs before using the application or having a task at hand. The authors in²² provide three essential reasons for a software to be tailorable:

- multidimensional diversities that tailorability must take into consideration in order to implement a software able to support different uses.
- The dynamism of individual and organizational work that matches the changing nature of work, forces the software itself to change over time.
- The uncertainty and ambiguity due to work practices require the use of alternative methods to accomplish tasks.

Empirical research has identified two major challenges in building tailorable systems³². The first is to support the re-design of the application while it is being used, and the second challenge is to allow users to take a major role in restructuring the application's infrastructure. In particular, for the second challenge, significant improvements have been described as, for example, supporting tailorability to address the problem of users' divergent skills. In this case, users with little experience can delegate some complex activities to developers.

In fact, software design following the traditional "Waterfall"³³ model is concerned about the capture, production and testing of a set of requirements. This reflects a partial view of an application domain with specific requirements in terms of functionalities and its interface design. This type of model assumes that the system requirements are clear at the beginning of the design phase and are stable for a long time. For this reason, recent approaches in software engineering try to find solutions to the dynamic change of users' requirements. In these approaches, users

are actively involved in the design process. They have the opportunity to articulate their needs according to the application domain, as we can see in Figure 2 extracted from ¹⁹.

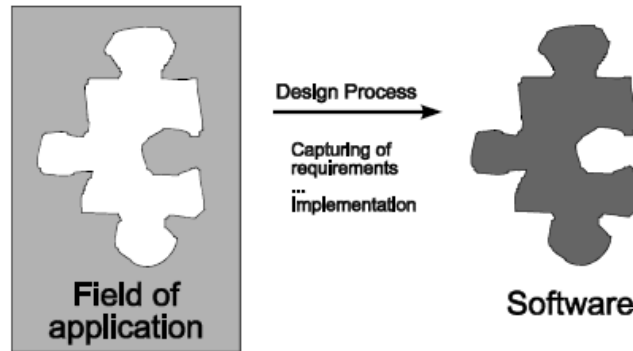


Fig. 2. An illustration of a tailorable application

Some authors ¹⁹ that are concerned with the design of tailorable systems suggest that for a good design of this type of application, some issues have to be taken in consideration:

- How can an application designer express the eventual needs and requirements of potential users, and how to introduce the necessary flexibility in these requirements?
- How flexibility can be technically implemented, resulting in the issue of software architecture?
- How can a flexibility offered in a software architecture can be accessible to end users?

In this work, we propose the use of web services and software agents in order to provide a remedy to the issue of the system's flexibility, which remains a concern for designers of tailorable groupware. In the following, we present the tools used in our work to build our groupware architecture.

2.4. Tools for tailorable groupware design

2.4.1. Web Services: Definition

W3C^a defines a web service as a software system that acts as an interoperable support in a machine-machine interaction. The system has an interface described in a form understood by the machine (Specifically WSDL), while other systems interact with the web service using SOAP messages transported through HTTP with

^a<http://www.w3.org/>

an XML serialization in conjunction with other web standards. In fact, service oriented architecture (SOA) has emerged due to its simplicity, clarity and normalized foundations. The concept of web services currently revolves around three acronyms³⁴, as we can see in Figure 3:

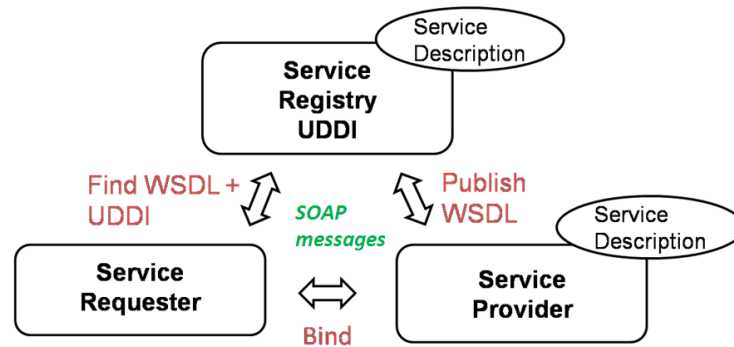


Fig. 3. Service Oriented Architecture (SOA)

- SOAP (Simple Object Access Protocol) is a protocol for inter-application exchange that is independent from any platform and based on XML. A SOAP service call is an ASCII flow embedded in XML tags and transported to the HTTP protocol.
- WSDL (Web Services Description Language) gives the XL description of web services by specifying the methods that can be invoked, their signatures and access point (URL, port, etc.). It is therefore equivalent to the IDL language for CORBA distributed programming.
- UDDI (Universal Description, Discovery and Integration) is a standard distributed directory, allowing both publishing and exploration of web services. UDDI acts as a web service itself, whose methods are called using the SOAP protocol.

Our choice of using web services is driven by the fact that they are: Language and platform independent (separation of specification from implementation), deployed over the internet (no centralized control, use of established protocols), loosely coupled (using synchronous and asynchronous interactions.) and interoperable (using standards already deployed and functional to support systems interoperability). On the other hand, a shared limitation of description standards based on XML is their inefficiency to express semantic information. In fact, to use a web service, a software agent has to have an interpretable description and means to access that

service. Hence, a matching mechanism is important for an effective discovery, which needs rich and flexible metadata that is not currently supported by the UDDI. An important objective is to establish a framework in which semantic descriptions are created and shared. For this reason, current web applications have to be built in way to support an ontology in order to declare and describe the services offered. This philosophy is known as the "Service-Oriented Computing (SOC)"¹¹.

Ontology: Ontologies provide a remedy to the constraints of traditional web. To the authors in ⁴¹, an ontology is defined as "the science or study of being", which is a set of terms that includes vocabulary, semantic interconnections and inference rules and logic for some popular domains. Ontologies are used by people, databases and applications that need to share domain information (medicine, fight against terrorism, imaging, automotive repair, etc.). An important achievement is the development of a new generation of web languages, allowing the creation of ontologies for any domain, as well as their instantiations in the description of specific websites. Ontologies are explicitly mentioned in a formal language, where several have been proposed in the literature, but the most common ones revolve around RDF^b and OWL-S^c.

2.4.2. *Software Agents*

There exist several definitions of software agents in the literature. The authors in ⁵ have identified the agent as a computing object (in the sense of object-oriented languages) whose behavior can be described by a script with its own means of calculation, and can move from a place to another in order to communicate with other agents. The authors explain that some researchers have given the definition of agent through a good description of its functioning. For example, an agent must necessarily have the necessary motivation to achieve a certain goal for its existence to be worthwhile in its environment. An agent can communicate with other agents in the environment and must have means which enable it to achieve its goals. According to ³⁹, an agent is a piece of software that acts on an autonomous basis to initiate charges on behalf of users. The authors say that the design of many software agents is based on the approach that users only need to indicate a high level goal instead of issuing explicit instructions, leaving the decisions to the agent. The agent shows a number of features that makes it different from other traditional components, including self-direction, collaboration, continuity, character, communication, adaptation, mobility and temporal continuity.

^b<http://www.w3.org/RDF/>

^c<http://www.w3.org/Submission/OWL-S/>

3. Machine-Machine collaboration

In this work, our aim is to integrate the notion of tailorability for the design of collaboration applications (groupware) by the integration and composition of services in order to generate new functionalities. The integration mechanism is put into practice by a collaboration protocol used by the machines over the internet. In the literature, this type of collaboration is implemented using simple client/server architecture. In our research, we realized that this type of architecture is not sufficient enough in a collaborative context, as many users may demand many services at the same time. Therefore, special mechanisms should be put in practice other than a standard request/response protocol to support the heavy interactions between systems, which should be able to handle any breakdowns or inconsistencies in the network (by decomposing it into three independent spaces). In our work, this new collaboration protocol between machines (or machine-machine collaboration) begins with a communication phase, where machines exchange information about the services demanded or proposed, then a coordination phase where these machines put in practice a workflow to exchange services, and finally the production phase where the exchanged services are registered in the groupware and offered to users.

As we can see in Figure 4, we define two types of collaboration: an "internal collaboration", which we call machine-machine, and an "external collaboration" that represents the actual collaboration between users in order to achieve a common task. The internal collaboration intervenes between components of the system (composition) or between two or more machines connected to a network (integration). In fact, for the users in collaboration, they don't actually 'see' this internal collaboration, that is made in an autonomous and invisible manner in order to achieve the desired tailorability. The users interacting with a groupware are essentially interested to functionalities that they can simply use while interacting with others. However, these two types of collaboration (internal and external) are heavily dependent on each other, as the internal collaboration will dynamically generate new services, and in consequence, will enhance the external collaboration. In this paper, we are interested in setting up a frame or a collaboration protocol for the internal collaboration. We argue that this type of collaboration protocol is an initial approach to put in practice the needed tailorability in software systems in general, and in CSCW systems in particular (human-human or external collaboration).

For the authors in ⁵, the collaboration is a work between multiple users in order to produce a common task (final product). In our system, this definition corresponds to the external (human-human) collaboration. In fact, the communication between different members of a team is primordial for the success of collaborative work. We suppose that the collaboration is based on communication, in other words we cannot have a coordination phase without communication, neither production without coordination. However, we can communicate without coordinating, and coordinate without producing. As mentioned previously, our objective is to insure a tailorable

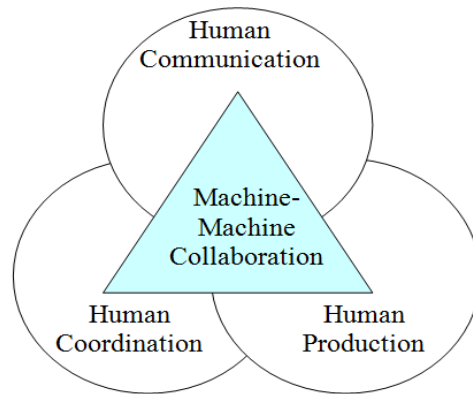


Fig. 4. Human-Human and Machine-Machine collaboration in the 3C model

external collaboration between users, by integrating/composing new services that can enhance the collaboration process in terms of communication, by (for example) chat and video conferences services, coordination by new workflow tools, and finally production that corresponds to using the generated services as a consequence of the integration/composition mechanism's. Hence, a tailorable (external) collaboration is triggered by a user demand for a new service, and terminates by an internal (machine-machine collaboration) (Figure 4) in order to satisfy the user's request. This process is put in practice by a temporary link between the local machine that demands the service and others machines or public repositories containing the needed services. We proceed by proposing a machine-machine collaboration formalism for this purpose.

3.1. *The basics of machine-machine collaboration*

As mentioned earlier, our aim is to implement tailorability in groupware systems. In fact, we focus on a particular type of tailorability that has two specific characteristics:

- It gives means to end-users to directly and dynamically change the behaviors of the collaborative system through the interface, without manually hard-coding nor stopping/restarting the system.
- It is based on service composition and integration in order to generate new behaviors on the fly.

Hence, we define composition and integration as follows:

- **Composition:** Process that allows composing two or more services, in order to create a new service with a new behavior.

- Integration: Process that allows adding one or more services to the application.
- Service: Software component that presents one or more functionalities. It can be: internal, which means implemented locally and integrated in the application during the design phase, or external, where it is seen as a web service that acts as an interoperable manner in the machine-machine collaboration.

Based on the 3C functional model for groupware, we introduce these two mechanisms (integration and composition) as a basis for generating new functionalities in a collaborative context. Therefore, to build a tailorable groupware based on the 3C functional model, the system should implement these two mechanisms in every space of the collaboration process (Communication, Coordination and Cooperation/production). We can see the new 3C model supporting groupware tailorability in Figure 5:

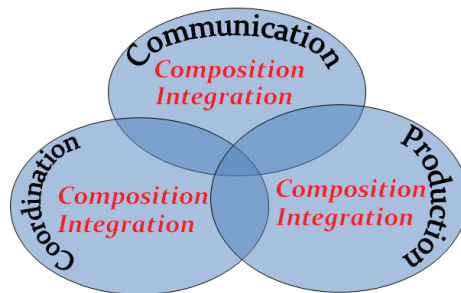


Fig. 5. The new 3C model supporting tailorability

3.1.1. Specifications and formalism

We present a collaboration formalism based on Web services for groupware tailorability. Our aim is to propose a generic formalism that can be applied on various applications using different technologies and interfaced with Web services. Indeed, one of the biggest advantages of Web services is to insure enough interoperability to connect various applications produced by different vendors and using different programming languages or frameworks. In fact, service-oriented architectures (SOA) are based on the engineering of traditional systems, but take in consideration specific characteristics and especially collaboration, where consumers and providers of services collaborate in order to invoke, search, and register services. Moreover, these systems can be composed during run-time by using existing services. Hence, it would be interesting to have design techniques that are independent from the

14 *Nader Cheaib, Samir Otmane, Malik Mallem*

used platform.

We extend the definition of a web service in ¹². In our work, a web service in our system is constituted of:

$$WS = \langle IOPE, QoS, Loc, Coll \rangle \quad (1)$$

Where

$$IOPE = \langle input, output, precondition, effect \rangle \quad (2)$$

IOPE are the semantic information of the Web service, and are defined using the OWL-S^d language to build web services ontologies. Hence, this information defines the Input, Output, Precondition and Effect of using a specific web service. IOPE is indeed an abstract characterization of what a service can do. These properties are based on the types of contents in the UDDI, by describing the necessary properties for a Web service to be dynamically discovered and/or composed.

$$QoS = \langle q1, q2, \dots, qk \rangle \quad (3)$$

qk is a quality of service (QoS) property, as the effectiveness, availability, response time etc.

In our work, we suppose that the Web services are localized in public repositories (UDDI), which can be accessed using SOAP messages and description files (WSDL).

$$Loc = \langle UDDI1, UDDI2..UDDIk \rangle \quad (4)$$

Finally, we suppose that the services are grouped in terms of communication, coordination and production services.

$$Coll = \langle Comm, Coor, Prod \rangle \quad (5)$$

We can see a collaboration between two entities i et j in Figure 6. In this example, these two entities are represented by a functional core (application kernel) and a service environment decomposed according to the 3C model. In fact, a collaboration requires at least two systems interacting together in order to execute a common task. Let's suppose that a system i demands to another system j to collaborate. This collaboration is designed by a communication, coordination and production of these two systems, as we can see in the equation 6.

$$Coll_j^i = \langle Comm_j^i, Coor_{ij}, Prod_i, Prod_j \rangle \quad (6)$$

In fact, a collaboration between two systems can be triggered by:

^d<http://www.w3.org/Submission/OWL-S/>

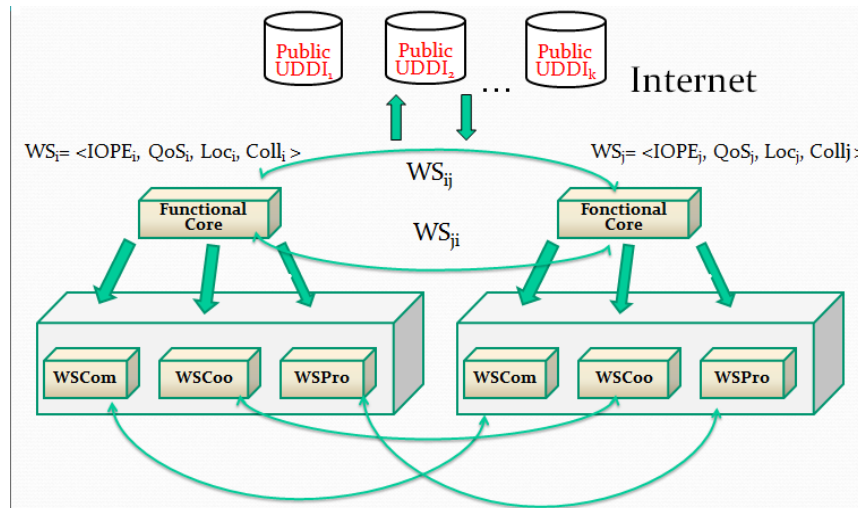


Fig. 6. Machine-machine collaboration based on Web services

- Demand of a user or group of users to use a specific service that is not present in the system.
- Demand of an internal service in order to be updated by new resources or information.

Hence, we define a set of protocols that decomposes the collaboration between entities into three main spaces ($\{Inf_{ij}\}$, $\{Actions_i\}$, $\{Resultats_i\}$):

$\{Inf_{ij}\}$ are exchanged information between two entities i et j , and are:

- Information (semantic or none semantic) concerning internal services, or services that are susceptible to be used/integrated in the system. This information can be the IOPE of the service, as well as non-functional Quality of Service (QoS) attributes such as performance, availability, security and localization in case the service is external.
- Information relative to the mission/task of the users. Hence, the system can adapt the services offered according to the task that users are participating in.

The $\{Actions_i\}$ is the subset containing all the triggered actions by the users of the collaborative system. These actions are:

- Search for a service using its syntactical (WSDL) or semantic (IOPE) de-

scription.

- Resigter/bind/delete a service in the system.
- Compose two or more internal services.
- Adapt the structure of the application based on QoS description, the performance of the machines and the underlying network used by each user (if geographically distributed). Hence, the application will offer a set of services that corresponds to the task, and hide the other services.

The $\{Resultats_i\}$ is a subset containing the results produced after executing the actions, and can be triggered following:

- A new integrated service, dynamically or statically (in case users intervene in integrating the service by manually coding or pointing to the WSDL description of the web service needed).
- A composed service of several atomic services.

The communication ($Comm_j^i$) phase starts when the entity i sends a demand Inf_{ij} to another entity j (equation (7)). The system j , in any case, sends a response Inf_{ji} back to i . Hence, the communication is based on an exchange of information (Inf_{ij}, Inf_{ji}).

$$Comm_j^i(Inf_{ij}) = \{(Inf_{ij}, Inf_{ji})\} \quad (7)$$

According to the agreements of the two entities i and j , the coordination phase begins (equation (8)), while continuing to exchange information about the services offered (Inf_{ij}, Inf_{ji}). Both systems will put action plans ($Actions_i, Actions_j$) where each one will execute. These action plans are considered as a type of workflow based on web services in order to coordinate tasks.

$$Coor_{ij}(\{Inf_{ij}, Inf_{ji}\}) = \{Actions_i, Actions_j\} \quad (8)$$

After the coordination phase, the production phase begins (equation (9)). Each of the two systems execute its proper actions ($Actions_i$ and $Actions_j$), and produce partial results ($Resultats_i$ and $Resultats_j$).

$$Prod_i(\{Actions_i\}) = \{Resultats_i\} \quad (9)$$

Once the partial results are obtained, they are combined (using a combination operator) in order to have the global result of collaboration between these two systems (a final product).

We consider the global collaboration $COLL$ (equation 10 between N services, as a triplet $\langle COMM, COOR, PROD \rangle$, where $COMM$ is the global communication, $COOR$ is the global coordination and $PROD$ is the global production of all the systems that are collaborating together for the purpose of exchanging services.

$$COLL = \langle COMM, COOR, PROD \rangle \quad (10)$$

The global communication *COMM* (equation 3.14) is represented by all the information tuples between a system i and another system j .

$$COMM = (Inf_{ij}, Inf_{ji}) \quad (11)$$

The global coordination *COOR* (equation 12) is represented by the set of actions $Actions_i$ of all the systems collaborating.

$$COOR = \{Actions_i / i = 1 \dots N\} \quad (12)$$

The global production *PROD* (equation 13) is the combination of all the partial results $Resultats_i$ of all the systems collaborating. This combination produces either a new composed service, or an integrated service. In case of a composition of services, two or more local services will collaborate together in order to produce the desired composed service in a system. Hence, we will be talking about two services i and j that are communicating, coordinating and producing.

$$PROD = \prod_{i=1 \dots N} (Resultats_i) \quad (13)$$

4. Human-Machine collaboration based on software agents - The C4 model

Software development requires a considerable progress in order to exploit new methods and techniques supporting the reuse of software components. In other words, software development need to move to compositional approaches³⁶. Software engineering methods give way to new development paradigms, including component-based and agent-based approaches. These approaches are gaining a lot of attention, where passive software components are remedied by the dynamics and social character of software agents. Indeed, agent-based technologies provide new mechanisms for components in order to engage in tasks as well as cooperate and process the requirements of dynamic and heterogeneous environments. From a multi-agent system (MAS) perspective, agents have access to an ontology with a semantic description of behaviors, which allows them to coordinate better in order to address specific tasks.

The C4 model is a formalism proposed by⁵, which later was developed into a groupware dedicated to the collaborative teleoperation via Internet. This model is based on the PAC*⁴² model that proposed three agents, each dedicated to three spaces of the 3C model to ensure the modularity of the system. In addition to these three spaces, the C4 model proposes a fourth agent: Collaboration agent. The combination of these four agents constitutes the "Collaborator Agent", as we can see in Figure 7.

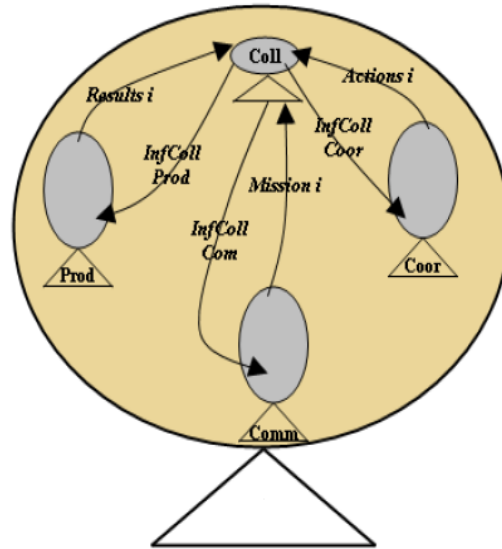


Fig. 7. Internal interaction in a collaborator agent

4.1. Collaboration Agent

The collaboration agent provides two main functions. On one hand, it enables communication between the three agents dedicated to the collaborator agent, and on the other hand, it establishes a direct interaction between each agent inside a Collaborator agent with the agent of another Collaborator agent. Thus, these two functions allow every agent of a Collaborator agent to communicate without the intervention of the collaboration agent, which increases the efficiency of the system. Moreover, it avoids the effect of bottleneck due to centralization. Indeed, when the communication agent finishes its work, it communicates its results to the collaboration agent. The latter is responsible for informing the coordination agent of the beginning of its work. The same process repeats with the coordination agent and the production agent.

A major drawback of the collaboration agent is the fact that it is a passive agent, pre-programmed to manage internal and external agents of each Collaborator agent in the system. The collaboration agent lacks the capacity to seek new resources, and hence to create new behaviors. In this paper, we remedy this constraint by proposing a groupware architecture that implements a proactive and a hybrid agent, capable of generating new behaviors in the system, by consuming web services as online resources. We discuss in details the groupware architecture proposed in the next section.

4.2. *Communication Agent*

The communication agent decides whether a mission can be accomplished or not and whether collaboration can take place. Accordingly, the communication agent is the cornerstone of the collaborator agent. The communication agent determines the state of the agent according to the perceptions it receives and its previous state. This agent manages all information provided to it, which represents the different perceptions that may be received the collaboration agent.

4.3. *Coordination Agent*

The agent coordination defines all the actions that the agent can perform according to the choice of the mission by the communication agent. The coordination agent receives a set of information sent to it by other coordination agents or the collaboration agent. This set includes the different perceptions received by the agent as well as the information sent to it by the collaboration agent.

4.4. *Production Agent*

The production agent is responsible for implementing actions due to the collaboration between different agents in the system, which are sent by the collaboration agent. Similarly to the other agents, the set of inputs and outputs of the production agent are the stimuli that the agent can produce, as well as the overall results due to the execution of procedures associated with the actions.

4.5. *Terminologies*

We propose a continuation of the work presented in ³⁵, where the C4 model (Collaborator agent) is proposed. A collaboration model is introduced, which is specific for the agents of the system and where the internal interactions are distinguished (Figure 7) from the external interactions (interactions between Collaborator agents). This model takes in consideration specific properties for multi-agent systems (MAS) and the characteristics of collaboration. The drawback of the model has led us to improve it by making it proactive and more dynamic. Thus, in the new groupware architecture proposed in this work, the collaboration agent is considered as a hybrid agent, which is able to seek new resources on the Internet as web services, and to exploit them internally as new behaviors. This process improves the collaboration process by making it "tailorable". We base our formalism for the software agents in the system on the work presented in ⁵:

$$\begin{aligned}
 \text{Agent} &= \langle i, w \rangle \\
 i &= \langle P_i, \text{Percept}_i, F_i, \text{Infl}_i \rangle \\
 w &= \langle E, \Gamma, \Sigma, R \rangle
 \end{aligned} \tag{14}$$

In this formalism, an agent i that interacts with the environment is considered as a tuple of a dynamic system $\langle i, w \rangle$, where the agent has only one global state in relation to the collaboration task. A world w of an agent is dynamic and changing at every action or reaction of the Collaborator agent. This world is modeled by:

- an environment E , which is the space in which it operates,
- a set of impact Γ induced by the agent in question following the transformation of its environment, and that represents the three subsets of information, actions and results (communication, coordination and production),
- a set of states Σ in which the agent passes through,
- and finally a law of evolution R of the world that the agent must comply in order to interact and cooperate with its counterparts.

An agent i is modeled by four parameters: Its perception function P_i , the set of perception that the agent receives $Percept_i$ which is the set of stimuli, a behavior function depending on its data F_i , and finally its production function of the influences depending on the agent's behavior $Infl_i$. We argue in our work that in order to arrive to a tailorable collaboration, the three spaces of collaboration (communication, coordination and production) of both worlds (web services and agents) have to interact.

5. Collaboration based on Web services (Machine-Machine) and Software agents (Human-Machine)

Services-oriented architecture (SOA) and multi-agent systems (MAS) are two increasingly important technologies for the construction of software systems. The objectives of these two architectures share some similarities such as, for example, creating flexible and distributed systems composed of loosely-coupled entities. However, there are major differences in their underlying technology: web services (which are the basic components of SOA) have standards for the description of interfaces and the protocols used. These standards are totally different from the communication protocol used by agents. Thus, these two technologies cannot interact directly with each other. Some research work in the literature studied this problem, and showed that the integration of agents and web services produced many interesting advantages. On one hand, web services have a well-defined and interoperable infrastructure, where on the other hand, software agents can provide social skills and intelligence for the applications (trust, reputation, commitment, etc.). Consequently, the integration of agents and web services can improve the adaptability, interoperability and openness of a software system in general, and a CSCW system in particular.

5.1. *Web services and Software Agents' Integration*

Many researchers argue that Web services are only regarded as passive, while agents can provide alerts and updates when new information becomes available. Unfortu-

nately, these technologies were originally developed separately using different standards and specifications. As a result, their integration becomes important in this context. Our idea is to exploit the proactive capabilities of agents in order to improve the behavior of web services in a collaborative and service-oriented architecture. With this paradigm, software components each representing a web service and a software agents will interact in order to provide unified services in a collaborative context. This corresponds well with the prediction of ⁵⁷ *"agents will become an essential part of most web applications, serving as a glue that allows a system as large as the web to be manageable and viable"*.

5.1.1. Approaches for integration

Various approaches exist in the literature that aim at integrating web services and software agents in many application fields. In this section, we present few of these approaches:

The authors in ⁵¹ offer a platform that provides a fast execution time of an agents' environment located inside a web container, which adds agents' functionalities to existing web servers. The components of the platform are deployed as web services, with SOAP (Simple Object Access Protocol) over HTTP acting as a communication channel through standard XML messages. Hence, a support for mobile agent can be added to the existing web infrastructure without the need to replace components or install any client software.

The authors in ⁵² offer a solution for the selection and composition of web services with software agents applied to the Marketplace. The use of agents introduces proactivity and autonomy in the composition process through autonomous operations and negotiations. The proposed software architecture enables sufficient flexibility and scalability, while providing an integrated set of tools as well as a support for communication and coordination.

The authors in ⁵³ propose an agent, "Do - I - Care", which is designed to help users discover the relevant changes on the web, by automating periodic visits to selected pages on the behalf of users. This agent uses machine learning in order to identify these changes and how often they occur. Once it detects a change, it will be attached to the webpage associated with the agent, which is also used for feedback as well as collaboration activities.

The authors in ⁵⁴ propose a 3-tier approach (intrinsic, functional and behavioral) in order to integrate web services and agents. The authors argue that to search for services, to integrate them into a composite service and to trigger and monitor their performances are among the operations that users are normally in charge of. However, most of these operations are complex and repetitive, with a big

22 *Nader Cheaib, Samir Otmane, Malik Mallem*

part adapted to the computer tool. Thus, software agents are suitable candidates to assist users in their operations.

The authors in ⁵⁵ present a balanced integration of web services and agents that is comply with the FIPA standards^e in order to create a software architecture in the context of the Agentcities project. The authors argue that web services have been developed without the concept of agents and can exist without agents. Thus, a "proxy" approach allows both platforms to evolve in parallel without imposing any restriction on one another. This approach accepts an equality between the roles of agents and web services, which is different from the traditional view that agents are to be considered on a higher level, and take only the roles of web services' suppliers and consumers.

The authors in ⁵⁶ propose an improved solution, WS2JADE for web services integration with multi-agent system. This approach has an interconnection layer that contains special agents called "Web Service Agent" or WSAG. These WSAGs glue agents and web services together, where the agents would offer web services as their own services. The second layer is the management layer, which is capable of active service discovery and to the automatic deployment of WSAGs in execution time. The combination of both static and dynamic layers is a distinct tool of WS2JADE, which is designed to actively discover web services and save them in multi-agent systems.

In fact, web services are increasingly used to provide active behavior on the internet, and give to end users features that were associated with multi-agent systems. It is therefore natural to consider what types of relationships exist or should exist between the agents and web services. We argue that they are separate components, although they may share common goals. In the next section, we present our tailorable groupware architecture *U3D*.

5.2. Groupware architecture: *U3D*

We recall that our objective is to integrate software agents and web services into a coherent entity that attempts to overcome the weaknesses of each technology, while enhancing their individual strength in the context of tailorable groupware design. In our work ^{37, 18}, we proposed the use of service-oriented architecture for the design of tailorable groupware architecture, which offers the necessary interoperability and re-configurability to system's components. We also discussed the importance of using software agents to improve the proactive discovery of services. In this section, we propose a tailorable groupware architecture that we call *U3D*, based on the integration of Web services (Machine-Machine) collaboration and Software agents

^e<http://www.fipa.org/>

(Human-Machine).

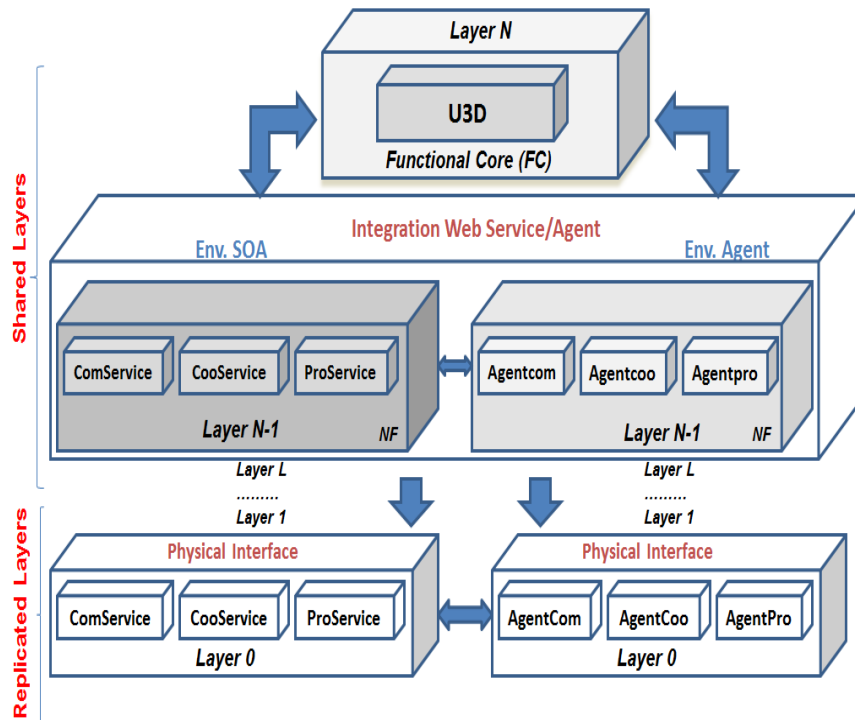


Fig. 8. U3D groupware architecture based on web services and software agents

We can see the overall architecture in Figure 8. This new architecture is based on the integration of the C4 model based on software agents⁵, and the *UDDIAC* architecture based on web services proposed in our previous work⁴⁴ that is based on the machine-machine collaboration formalism presented in the previous section. Thus, we get a groupware architecture composed of two parallel environments but working together in order to create a collaborative and tailorable software architecture, taking advantage from the properties of both technologies.

We rely on the Arch model² that aims to separate the physical interface (Layer 0 in Figure 8) from the functional core (FC) of a system (Layers N-1 and N). However, in contrast to the Arch model where the functional core (FC) is a dead-end component (implements static domain functionalities), our FC is connected to the internet in order to put in practice the collaboration protocol discussed in the previous section, with other machines over the network. In this paper, we will solely concentrate on the FC which is the main component of the system, while we make no assumption about the structure or specifications of the other components of

the groupware architecture. Furthermore, we rely on Dewan's model ³, that is a generalization of the Arch model, and that structures a groupware system into a variable number of replicated and shared layers. Thus, it defines a collaboration degree between the system's components and users, where the highest layer is the most semantic one corresponding to the FC (coincides with the one of the Arch model), and the lowest layer representing the material level (Arch's Physical Interaction component).

Our groupware architecture is constituted of a root representing shared layers, meaning that it is shared among all the users in the system, and several replicated layers for every user. The layers communicate vertically using interaction events, and use collaboration events (formalism presented) for interacting with machines over the network. However, in contrast to the clover model ⁶ where the functional core is also split into two layers (one private and shared, while the other is replicated and public), the functional core (FC) in our model is represented by two layers that are both shared and constitute the root of the system. This mechanism allows all users to manipulate domain objects and have access to various centralized services during the interaction with the system. The layers underneath the FC are replicated, and manage resources for each user. It should be noted that Figure 8 shows only the FC and the physical interaction layer. In the last section, we put in practice the physical layer as a web interface that will serve as a case study of our model.

5.2.1. *Functional Core (FC) Decomposition*

The shared layers of the architecture constituting the system's FC enable users to manipulate domain objects and have access to various services in the system, while the replicated layers handles the set of services and the state of the system that is private for every user in collaboration. We extend this layer abstraction as in ⁶ by decomposing each layer of the architecture into sub-components, each dedicated to one facet of Ellis' 3C model, while providing and managing specific services for communication, coordination and production. However, we assume that only the layers on the level N-1 and on the lowest level (Layer 0) satisfy these three main classifications, while we have made no assumption till now about the decomposition of the highest semantic layer in the architecture, which is for us mainly composed of one single component for enabling collaboration with other machines. The sub-components on the level N-1 are enclosed in a software interface exposing its functionalities to the clients as in ⁶, by dividing the services in the system into three main services: communication, coordination and production services. These services can be considered as orchestrations of atomic services in the system ⁸ based on the functionalities they offer.

Indeed, this environment focuses on three main aspects: (a) a framework for organizing the software components through the network, (b) a mechanism for the

publication and registration of Web services so they can be dynamically discovered, (c) a set of standards that allows components to exchange data in the system as well as with distributed components on the internet. In our system, the software agents of the system behave as a set of basic components that interacts with online resources that are Web services. The aim as discussed is to be able to implement tailorability in a collaborative context that is being achieved through the integration of external Web services or the composition of internal ones in order to generate new communication, coordination and production services. In order to create a link between the software agents and the web services of the system, we have integrated into the FC a special component of the JADE^f platform called Web Service Integration Gateway (WSIG). This component is deployed on the highest level of the FC, and acts in our system as a hybrid software agent that creates SOAP invocations in order to integrate external Web services, as well as compose internal Web services using their semantic description specified in an internal ontology.

5.2.2. SOA environment

As we can see in Figure 8, the first component on the level N-1 is based on an SOA environment. This component contains all the web services in the system grouped into 3 main services: communication services, coordination services and production services. By classifying services in the system into these three main categories, the main spaces of the software collaboration process defined by the 3C model⁴, as we have mentioned, are satisfied. Note that we use the term 'Production' to mean 'Cooperation' of activities (used in the 3C model: Communication, Coordination and Cooperation):

- ComService: contains all services offering means of communication between users in collaboration (videoconference service, voice recorder service etc.).
- CoorService: contains services implementing rules of coordination by codifying their interaction (i.e. workflow).
- ProService: contains services that are the collaborative product of using the architecture. (Ex: Paint application, Word document etc.).

These services can be considered as orchestrations of various other services in the system⁸, and include services based on the functionalities they offer. Compared to the architecture proposed in³⁷, the UDDI is viewed as a dynamic registry for web services description enhanced with software agent's capabilities, and containing definitions of services running in the system that are susceptible of undergoing tailorability activities. The definitions of these web services are provided using the standardized language for web services description (WSDL) and are connected to adaptable aspects that are the services themselves, residing in the SOA environment.

^f<http://jade.tilab.com/>

5.2.3. *Agents' Environment*

In parallel to the SOA environment, a JADE environment constitutes the other part of the FC on the level N-1. This layer is populated with software agents that are deployed on a JADE environment using its libraries for implementing agents' behaviors. The adopted paradigm of communication between layers is an asynchronous message passing with a format specified by the ACL (Agent Communication Language) defined by FIPA. This format includes a number of fields, more specifically the sender of the message, the list of recipients, the communication intention that indicates the purpose of sending the message, the message content and its language i.e. the used syntax to explain the content that could be understood by the sender and the recipient, and finally the ontology i.e. The vocabulary of symbols used in the contents and their meanings that also should be understood both by the sender and the recipient of the message.

As in the SOA environment, all agents are grouped into three main classes: communication, coordination and production agents. The use of agents, as we have mentioned before, is to make the discovery of new services in the system dynamic, meaning that new web services will be actively integrated into the FC without stopping the execution of the system. These services are normally used by users on the physical layer, which is the lowest layer in the architecture. The functional decomposition of the layer into three main sub-components corresponding to the 3C model will enhance the interaction between web services in the system with software agents. Hence, every agent in one particular sub-component would know exactly where to search for a particular web service in the SOA environment that best suits the functionalities it can offer. Each sub-component in this layer manipulates semantic objects dedicated to one of the 3C model functionalities, and performs specific processing functions on its services.

5.3. *Collaborative Hybrid Agent - Orchestration of services*

In this section, we present an essential component of our groupware architecture, which is the main agent that handles the integration/composition of services. In section 3, we have defined a machine-machine collaboration formalism for the online exchange of services. Indeed, an essential component of our groupware architecture is a hybrid agent that acts as a mediator between the physical agents that implement the core of our system, with the internal/external services. Recall that our system insures the integration of external web services, as well as the composition of internal ones for the creation of new functionalities on runtime. As we can see in the Figure 9, our hybrid agent acts as the interface between users in collaboration and the services present in the system. In fact, the hybrid agent provides a replicated interface for the users in order to tailor the services in the system. Hence, every user in collaboration will be represented by an instance of the hybrid agent in the system, which will customize the services offered based on the needs of each user in collaboration.

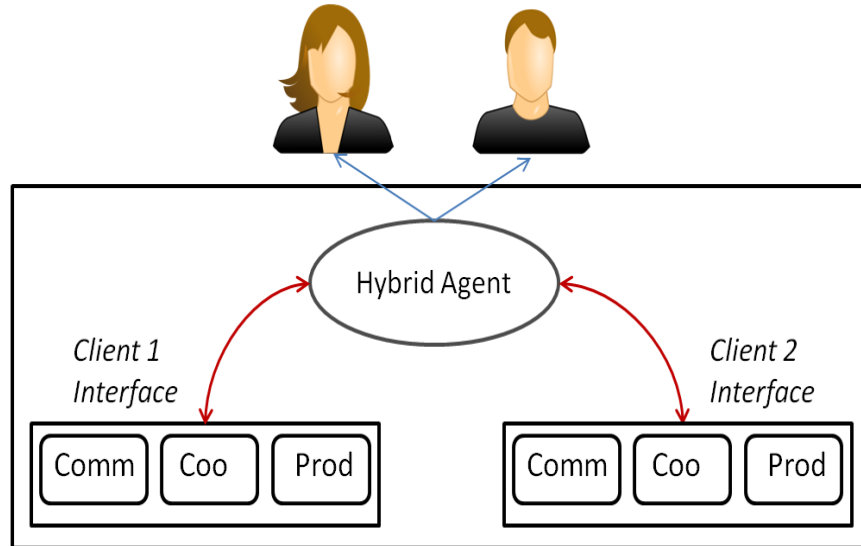


Fig. 9. An example of the use of hybrid agent involving two or more users in collaboration

In our work, we did not want to implement web services as intelligent agents themselves, which may break compatibility with existing web services. Hence, web services are considered external/internal resources to be exploited by software agents. Thus, we keep the autonomy of both worlds while taking advantage of their integration's synergy in order to create a tailorable collaborative architecture. As these two technologies use different protocols and standards, their integration is managed, in our system, by the hybrid agent that we discuss in this section. In Figure 10, services are divided into three main spaces: communication, coordination and production. Every space is considered as a set of basic services offering functionalities relative to the space they belong to. At the conceptual level, our hybrid agent that we call *WAG* ("Web Agent") manages the interaction between web services and agents in the system. In fact, the *WAG* agent is analogous to the collaboration agent of the C4 model introduced in section 4, however, the main difference is that our agent is a hybrid agent (where the collaboration agent is a passive one) that has special features such as proactivity, which enable it to interact autonomously with other components in the system as well as the outside world. Hence, the aim is to ensure the "orchestration" of services using workflow processes in order to integrate them based on their functional and non-functional descriptions, as well as compose them with existing services in the system, if applicable.

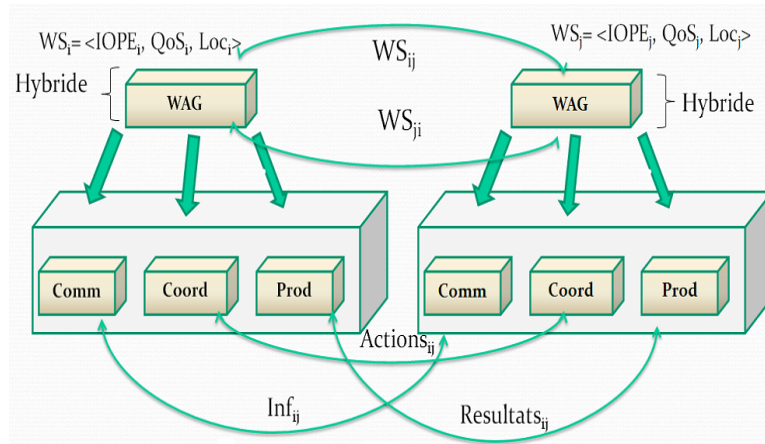


Fig. 10. WAG - Hybrid Agent

5.3.1. Agents and communication services

As mentioned previously, the purpose of applying the 3C model is to ensure the modularity of the system, which eventually reduces the complexity of implementation. For example, it will be easier to add a communication service (video, chat, etc.) without affecting other services. Communication services are managed by the communication agents. In the internal (machine-machine) collaboration, the information handled in this phase represents the different perceptions that the communication agents can receive ($Percept_j$), such as the information issued by WAG (Inf_i) or other agents in the system. The set of inputs and outputs of the communication agents are:

- $\{Inf_{ij}\}$: The set of information sent by a communication agent i to another communication agent j .
- $\{Percept_i\}$: The set of stimuli and sensations a communication agent can produce.

5.3.2. Agent and coordination services

Coordination is a fundamental step before actually producing. Thus, the coordination agent defines a set of actions with the WAG before producing, either by composing or integrating a new service. This agent accomplishes these actions after the mission has been selected by the communication agent. The coordinating agent receives a set of information that includes the different perceptions ($Percept_j$), as well as the information sent by WAG (Inf_i).

The set of inputs and outputs of the coordination agent coordination:

- $\{Actions_i\}$: The set of executed actions by the agent i . In our system, the most important coordination tasks are those defined by the communication agent: the integration of new services and the composition of two or more internal services.
- $\{Percept_i\}$: These stimuli are the functional (IOPE) and non-functional (QoS) parameters that the web services produce.

5.3.3. *Agent and production services*

The production phase is the result of the collaboration between machines. Thus, the production agent is responsible for implementing the actions issued in the coordination phase and sent by *WAG*. The set of inputs and outputs in this phase is:

- $\{Percept_i\}$: The set of stimuli and sensations that a production agent can produce.
- $\{Resultats_i\}$: The set of results as a consequence of executing the actions from the coordination phase. These results contain the newly produced service and all information concerning this particular service (physical address, semantic information and QoS).

In Figure 11 below, we can see a conceptual architecture of the communication, coordination and productions agents that represent the three spaces of the 3C model, respectively.

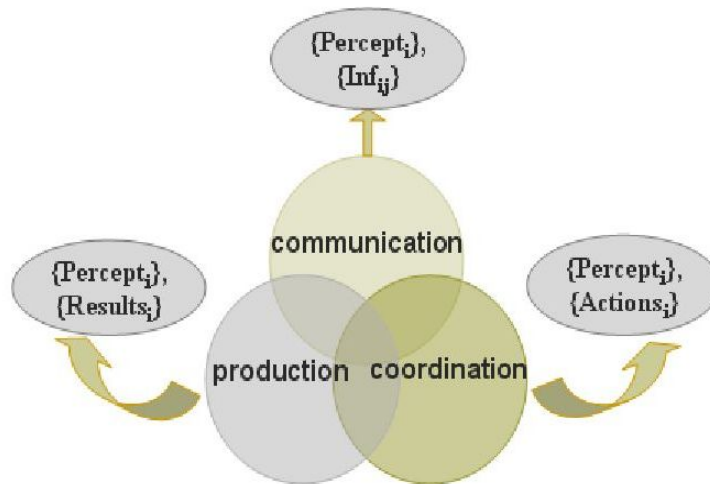


Fig. 11. A conceptual design of the communication, coordination and productions agents

5.4. *Tailorability algorithms*

In this section, we present tailorability algorithms that take into account the roles of the communication, coordination and production agents in the process of service integration and composition. Most importantly, this algorithms take in consideration the role of the WAG that, as mentioned, acts as a mediator between the users in collaboration with services provided in the system. We should mention that the algorithms presented deal implement the collaboration between our system and other machines connected to the network. Thus, they are an implementation of the machine-machine collaboration formalism that we presented in section 3.

The algorithms presented are shown in Figures 12, 13 and 14 respectively, and are applied on the groupware architecture discussed in the previous section. We also present the algorithm that deals with translating messages between services and agents in order to interact with an external web service (Figure 15). These algorithms put in practice the role of WAG in order to orchestrate tasks between the agents of the system for defining tailorability mechanisms. In these algorithms, the WAG introduced in the previous section is physically implemented as a special component in the JADE platform called Web Service Integration Gateway (WSIG)[§]. This component acts as a mediator between the web services and software agents, by creating SOAP requests from agents' descriptions, as well as allocating different

[§]<http://jade.tilab.com>

tasks to the agents' in order to integrate/compose a service.

In the communication phase (Figure 12), and in the case of a composition mechanism (Line 3), WSIG is notified of the mission and thus extracts the non-functional information of both services i and j to be composed (Line 4 and 5). The WSIG sends this information to the coordination agents (Line 7). In case of service integration, the WSIG is notified about the service to be integrated, as well as its QoS (Line 8, 9 and 10). The actions to be performed are sent to the WSIG (Line 12 and 13).

```

1. Communication ( $Comm_i$ ):
2.   While (MessageQueue NON empty && Service NULL) do
3.     If ( $Mission_i = COMPOSE$ ) Then
4.       WSIG  $\leftarrow$  missionCOMPOSE( $i,j$ );
5.       WSIG  $\leftarrow$  QoS( $i,j$ ) ;
6.       If (QoS = TRUE) Then
7.         | send  $CooAgent \leftarrow (ComService_i, AgentComm_i, WSIG_i, TRUE)$ ;
8.       If ( $Mission_i = INTEGRATE$ ) Then
9.         WSIG  $\leftarrow$  missionINTEGRATE( $i$ );
10.        WSIG  $\leftarrow$  QoS( $i$ );
11.        | send ( $Actions_i, WSIG_i$ );
12.        If (QoS & Loc = TRUE) Then
13.          | send ( $ComService_i, AgentComm_i, WSIG_i, TRUE$ );

```

Fig. 12. Communication algorithm of an agent $Comm_i$ with a service

In the coordination phase (Figure 13), and after the communication phase (Line 2), coordination agents receive the message containing the description of the service (Line 3, 4 and 5). In case of a composition (Line 6), coordination agents extract and test the IOPEs of the two services to compose in order to ensure that the product generates a service with a new behavior (Line 8 and 9). Then, they send the results to the WSIG (Line 10). In the case of an integration (line 11), coordination agents receive the non-functional information of the service to be integrated, and test its QoS and its location (Line 13 to 16). If the QoS is satisfactory, the service is extracted and sent to WSIG (Line 17 to 19).

After the communication and coordination phase, the production phase (14) begins, which contains the results of the collaboration. At Line 2 and 3, production agents receive the message. Whether the service is composed or integrated, the production agent saves the service in the system by creating a new entry in the UDDI. The results are sent to WSIG, and the production phase is completed (Line 4-8).

32 *Nader Cheaib, Samir Otmane, Malik Mallem*

```

1. Coordination ( $Coor_i$ ):
2.   If (Communication = TRUE) Then
3.     AgentCoo  $\leftarrow$  receiveMessage(WSIG);
4.      $AgentCoor_j \leftarrow$  message.getContent(1);
5.      $Mission_i \leftarrow$  message.getContent(2);
6.     If ( $Mission_i = COMPOSE$ ) Then
7.        $Get(Source_i, Source_j) \leftarrow (ServiceIOPE_i.ServiceIOPE_j)$ ;
8.       If( $(Inf(IOPE, Loc)Source_i) * (Inf(IOPE, Loc)Source_j) == TRUE$ );
9.        $Actions_i \leftarrow$  ChoiceActionsWith( $AgentCoor_j, Mission_i$ );
10.      send ( $Actions_i, WSIG_i$ );
11.     If ( $Mission_i = INTEGRATE$ ) Then
12.       AgentCoo  $\leftarrow$  receiveMessage(WSIG);
13.        $AgentCoor_j \leftarrow$  message.getContent(1);
14.        $Mission_i \leftarrow$  message.getContent(2);
15.       AgentCoo  $\leftarrow$  testQoS();
16.       AgentCoo  $\leftarrow$  testLoc();
17.       If (QoS & Loc = TRUE) alors
18.          $Service \leftarrow$  message.getContent(WSIG);
19.         send ( $Service_i, AgentCoo_i, WSIG_i, TRUE$ );

```

Fig. 13. Coordination algorithm of an agent $Coor_i$ with a service $Comm_j$

```

1. Production ( $Prod_i$ ):
2.   If (Coordination = TRUE) Then
3.     AgentPro  $\leftarrow$  receiveMessage();
4.     If (message.compose = TRUE OR message.integrate=TRUE) Then
5.       Register NewServicePro  $\leftarrow$  AgentPro();
6.        $Results_i \leftarrow$  ExecuteActions( $Actions_i$ );
7.       send ( $Results_i, WSIG_i$ );
8.       Terminate ( $Prod_i, WSIG_i$ );

```

Fig. 14. Production algorithm of an agent $Prod_i$ and a service $Comm_j$

In the algorithm below, we present the messages sent by the system in order to interact with an external service in case of an integration. Thus, communication agents receive information about these services and send them to WSIG (Line 4-7). The WSIG creates an invocation SOAP that contains the information relative to the service, particularly its WSDL description (Line 8). The WSIG receives the incoming SOAP message containing the information of the service to be integrated, and sends this information to the local UDDI (Line 10 and 11). Also, it notifies the agent that initiated the request about the completion of the task (Line 13, 14 and 15).

```

1. Translate ( $Comm_i, Coo_i, Prod_i$ ) :
2.   While (MessageQueue NON empty) do
3.     message  $\leftarrow$  receiveMessage();
4.     si (message.sender =  $Comm_j$ ) alors
5.        $AddressComm_i \leftarrow$  message.getContent(1);
6.        $ACLComm_i \leftarrow$  message.getContent(2);
7.       send  $WSIG_i(AddressComm_j, ACLComm_i)$ ;
8.        $SOAPOUT_j \leftarrow (AddressComm_j, ACLComm_i)$ ;
9.     si ( $SOAPOUT_j(Comm_i) = TRUE$ ) alors
8.        $SOAPIN_j \leftarrow (SOAPOUT_j, AddressCommS_j, QoS_i)$ ;
10.       $WSIG_j \leftarrow (SOAPIN_j)$ ;
11.      send  $UDDI_k(SOAPIN_j)$ ;
13.      Translate  $ACL_j \leftarrow (SOAPIN_j)$ ;
14.      send  $DF_k(ACL_j, Agent_j)$ ;
15.      send ( $Results_i, Coll_j$ )  $\leftarrow AgentInterface_j$ ;

```

Fig. 15. Algorithm for the construction of a web service invocation

5.5. Discussion

The originality of our model is the use of existing technologies' synergy in order to create a tailorable and interoperable software architecture for groupware. Moreover, our model is inspired by the Arch² and Dewan's³ models for separating the core functionality (logic of the application) from its interfaces, and thus carrying with it many essential properties such as flexibility, which is also crucial in the CSCW domain. Indeed, the two layers constituting the FC are both shared and handle exclusively the services and their dynamic integration, which is different from the Clover model⁶ that advocates a replicated functional core for every user by managing their private domain-dependent objects. We affirm that a functional core adaptor situated between the functional core and the physical layer (which was not discussed in the work presented in this paper) is more suitable to handle this type of data, while dedicating the core of the application solely to handle tailoring system's services. Hence, every newly added service will be shared by all users' participating in a particular session. Also, the 3C model is used in our proposed system in order to define and classify the services proposed in the system. In fact, the functional breakdown according to the 3C model contains several properties. In fact, from the implementation perspective, it will result in a greater modularity which reduces the complexity of groupware's implementation. For example, it would be easy to add a new communication service by adding, for example, a video stream mechanism without affecting existing services in the system. This could reduce the development cost and computational time, while enabling the addition of independent and heterogeneous services in order to improve the distribution of features and increase the modularity of code, and also by insuring interoperability in every aspect of the collaboration process. In fact, we have made no assumptions about the decompo-

sition of other layers in the system according to the 3C model. This fact enables the addition of independent and heterogeneous layers to improve the distribution of features and insuring interoperability on every layer of the architecture (by using FIPA and W3C^h specifications and standards). As for the branching point discussed in ³⁸, we have fix it after the FC layers, which induce a lower replication degree than the Clover model, but convenient in order to ensure state consistency of services, as well for collaborating users to share discovered services and reusing them when needed.

Moreover, our model identifies the implementation architecture that is deduced from the theoretical model in order to achieve tailorability in collaborative applications, where opposed to other models, it identifies explicitly a component as a web service and a software agent. In fact, we have put in practice our groupware architecture proposed in this paper on an academic project called ARITI-C¹⁵ for collaborative online teleoperation. In fact, the initial ARITI-C system that is based on a multi-agent system had a major constraint: The behaviors that allow the manipulation of the robot are static and pre-coded into the functional core of the application. Thus, users are limited to a few operations or behaviors (grabbing objects, rotating etc.), and the insertion of new behaviors during the phase of collaboration is not supported. In the tailorable ARITI-C system, users are able to collaborate together using a visual interface in order to create a composition of sub-missions that the robot should do. In fact, we have introduced an ontology in order to semantically encode the various behaviors of the robot. Using this ontology, the mission created by the users in collaboration is composed from the various sub-missions encoded in the ontology. This mission is then translated to code using the *OntologyBean Generator*ⁱ plugin of the *Protege*^j framework and dynamically injected in the system's core in order to be executed by the robot. In this manner, our new system enabled tailorability of services by the dynamic composition of new behaviors without the need to stop the system's execution to manually code new behaviors.

6. Conclusion

In this paper, we proposed a new groupware architecture based on web services and software agent that introduces tailorability to the design of collaborative applications. In this model, a hybrid agent that we call *WAG* (or *Web Agent*) handles the composition/integration of services based on users' preferences, bringing tailorability in real time without having to manually code new functionalities in the collaborative system. Also, we have introduced a machine-machine collaboration formalism in order to implement services' tailorability in a collaborative context.

^h<http://www.w3.org/>

ⁱ<http://protege.cim3.net/cgi-bin/wiki.pl?OntologyBeanGenerator>

^j<http://protege.stanford.edu/>

In fact, the originality of our model is in the use of existing technologies in order to create a tailorable and interoperable groupware architecture. However, some services to be integrated would need to store state information, which is still not possible for SOAP-based Web services. One solution is for the state and session information to be transmitted as XML parameters and stored in a database that can be implemented, in the groupware architecture proposed, on a software layer directly below the functional core of the system.

The work proposed in this paper is the basis of a groupware architecture that has been implemented in ARITI-C¹⁵ for online robot teleoperation. For our future work, we aim to shift the collaboration formalism discussed in this paper, and that primarily involves the services offered by the machines over the internet, to the software layers that are the closest to the users. In other words, we want the user to be the main entity in the tailorable collaboration formalism presented, and not the machine. One solution is to conduct research on tailorable interfaces (Layer 0 in our software architecture) in a collaborative setting. We believe that the work presented in this paper is one of the first steps towards shifting the web services technologies' into tailorable CSCW systems.

References

1. C. M. Wang, J. N. Reddy and K. H. Lee, *Shear Deformable Beams* (Elsevier, Oxford, 2000).
2. L. Bass. *A metamodel for the runtime architecture of an interactive system. User Interface Developers' Workshop, SIGCHI Bulletin.*, 24(1), 1992.
3. P. Dewan. *Architectures for collaborative applications. CSCW Journal, Trends in Software, John Wiley & Sons*, pages 169–194, 1999.
4. C. A. Ellis. *Conceptual model of groupware. Proc CSCW, ACM Press NY*, pages 79–88, 1994.
5. N. Khezami, S. Otmane, and M. Mallem. A new formal model of collaboration by multi-agent systems. *Proc IEEE KIMAS, Massachusetts, USA*, pages 32–37, 2005.
6. Y. Laurillau and L. Nigay. Clover architecture for groupware. *Proc CSCW, ACM*, pages 236–245, 2002.
7. A. Morch. Three levels of end-user tailoring: customization, integration, and extension. *Journal in Computers and design in context, MIT Press*, pages 51–76, 1997.
8. C. Pletz. *Web services orchestration. a review of emerging technologies, tools and standards. Hewlett Packard White Paper*, January 2006.
9. R. Slagter, M. Biemans and H. ter Hofte. *Evolution in use of groupware: Facilitating tailoring to the extreme. Seventh International Workshop on Groupware*, pages 68–73, 2001.
10. S.A McIlraith and T.C Son and H. Zeng. *Semantic web services. Intelligent Systems Journal, IEEE*, Volume 16, Number 2, Pages 46–53, 2005.
11. M.P Papazoglou and D. Georgakopoulos. *Service-oriented computing, Communications of the ACM*, Volume 46, Number 10, Pages 25–28, 2003.
12. E. Lee and B. Lee. *An Agent-Based Web Service Composition Using Semantic Information and QoS. Journal Agent and Multi-Agent Systems: Technologies and Applications*, pages 928–937, 2007, Springer.

36 *Nader Cheaib, Samir Otmane, Malik Mallem*

13. A. Dinis, N. Fies, N. Cheaib, S. Otmane, M. Mallem, A. Nisan and J.M. Boi. *DIGITAL OCEAN: A National Project for the creation and distribution of Multimedia Content for Underwater Sites. Proc. of the 14th International Conference on Virtual Systems and MultiMedia, Dedicated to Digital Heritage, VSMM'08*, pp. 389-396, (ISBN 978-963-8046-99-4), Limassol Cyprus, October 2008.
14. N. Cheaib, S. Otmane and M. Mallem. *Web services and Software Agents for Tailorable Groupware Design. in the book Emergent Web Intelligence: Advanced Semantic Technologies, Springer Verlag in the series "AI&KP"*, pages 185-208, 2010
15. S. Otmane, N. Cheaib and M. Mallem. *Internet-based Collaborative Teleoperation: Towards tailorable groupware for teleoperation. in the book End-to-End Quality of Service Engineering in Next Generation Heterogeneous Networks*, published by Wiley&ISTE/Hermes, pp 163-193, 2008
16. E.M Maximilien and M.P Singh. *A framework and ontology for dynamic web services selection. Journal of Internet Computing, EEE*, Volume 8, Number 5, pp 84-93, 2004
17. O. Stiemerling and A.B Cremers. *Tailorable component architectures for CSCW-systems. Proceedings of the Sixth Euromicro workshop on Parallel and Distributed Processing*, IEEE, pages 302-308, 2002
18. N. Cheaib, S. Otmane and M. Mallem. *Combining FIPA agents and web services for the design of tailorable groupware architecture. Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*, ACM, pages 702-705, 2008
19. O. Stiemerling, and A.Cremers, *Tailorable Component Architectures for CSCW-Systems. Proceedings of the 6th Euromicro Workshop on Parallel and Distributed Programming*, pages 21-24, 1998
20. V. Wulf, *Let's see your search tool ! collaborative use of tailored artifacts in groupware. In Proceedings of the international ACM SIGGROUP conference on Supporting group work*, pages 50-59. ACM New York, NY, USA
21. G. Teege, *Users as composers : Parts and features as a basis for tailorability in CSCW systems. Journal of Computer Supported Cooperative Work (CSCW)*, Springer, 9(1), 2000, pages 101-122
22. H. Kahler, *Supporting collaborative tailoring. PhD thesis, Department of Communication, Journalism and Computer Science. Roskilde University, Denmark.*, 2001
23. R. Slagter, *Dynamic Groupware Services : Modular design of tailorable groupware. PhD thesis, Telematica Instituut, the Netherlands*, 2004
24. A. Fernandez, *Groupware for Collaborative Tailoring. PhD thesis, University of Hagen - Germany*, 2005
25. D. Torres, A. Fernandez, G. Rossi, and S. Gordillo, *Fostering Groupware Tailorability Through Separation of Concerns. Lecture Notes in Computer Science 4715 :143*.
26. O. Stiemerling, R. Hinken, and AB. Cremers, *The EVOLVE tailoring platform: supporting the evolution of component-based groupware Third International Enterprise Distributed Object Computing Conference, 1999. EDOC'99. Proceedings.*, pp 106-115, 1999
27. M. Biemans, and GH. Ter Hofte, *Tailorability: state-of-the-art Gigaport project deliverable, Telematica Instituut, The Netherlands*, 1999
28. G. Bourguin, *Un support informatique à l'activité coopérative fondé sur la Théorie de l'Activité: le projet DARE University of Sciences and Technology of Lille*, 2000
29. G. Bourguin, *Proposition pour une gestion dynamique de l'inter-activités dans le TCAO Proceedings of the 16th conference on Association Francophone d'Interaction Homme-Machine table of contents*, pages 191-194, ACM New York USA, 2004
30. A. Morch, O. Stiemerling and V. Wulf, *Tailorable groupware: issues, methods, and*

- architectures *ACM SIGCHI Bulletin*, pages 40-42, ACM, 1998
31. V. Wulf and B. Golombek, *Direct activation: A concept to encourage tailoring activities Behaviour & Information Technology Journal*, Volume 20, Number 4
 32. V. Wulf, V. Pipek, and M. Won, *Component-based tailorability: Enabling highly flexible software applications International Journal of Human-Computer Studies*, Volume 66, Number 1, pages 1-22, 2008
 33. Y. Laurillau, *Conception et realisation logicielles pour les collecticiels centres sur l'activite de groupe: le modele et la plateforme Clover PhD Thesis, University of Joseph Fourier, Grenoble, France, 2002*
 34. E. Newcomer, *Understanding Web Services: XML, Wsdl, Soap, and UDDI Addison-Wesley Professional*, 2002
 35. N. Khezami, *Vers un collecticiel base sur un formalisme multi-agent destine a la teleoperation collaborative via Internet PhD Thesis, University of Evry Val d'Essonne, Evry, France, 2005*
 36. P. Buhler and J. Vidal, *Semantic web services as agent behaviors Agentcities: Challenges in Open Agent Environments*, pages 25-31, 2003
 37. N. Cheaib, S. Otmane and M. Mallem, *Integrating internet technologies in designing a tailorable groupware architecture 12th International Conference on Computer Supported Cooperative Work in Design, 2008. CSCWD 2008*, pages 141-147, 2008
 38. J.F. Patterson, *A taxonomy of architectures for synchronous groupware applications ACM SIGOIS Bulletin*, Volume 15, Number 3, pages 27-29, 1995, ACM
 39. Z. Maamar, QZ. Sheng and B. Benatallah, *Interleaving Web Services Composition and Execution Using Software Agents and Delegation Proceedings, Workshop on Web Services and Agent-Based Engineering*, 2003
 40. S. Otmane, M. Mallem, A. Kheddar and F. Chavand, *ARITI: an Augmented Reality Interface for Teleoperation on the Internet in Advanced Simulation Technologies Conference, High Performance Computing" HPC 2000*, pages 254-261, 2000
 41. M. Blois, M. Escobar and R. Choren, *Using agents and ontologies for application development on the semantic web Journal of the Brazilian Computer Society*, Volume 13, pages 35-44, 2007
 42. G. Calvary, J. Coutaz and L. Nigay, *From single-user architectural design to PAC*: a generic software architecture model for CSCW Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 242-249, 1997
 43. JADE Board *JADE Web Services Integration Gateway (WSIG) Guide JADE Documentation*, 2005
 44. N. Cheaib, S. Otmane and M. Mallem, *A Machine-Machine Collaboration Formalism based on Web services for Groupware Tailorability Proceedings of the 15th International Conference on Computer Supported Cooperative Work in Design (IEEE CSCWD 2011)*, Lausanne, Switzerland, pp 238-245, 2011
 45. M. Wright *A detailed investigation of interoperability for web services. Master Thesis, Rhodes University, South Africa.*
 46. G.C Gannod, J.E Burge and S.D Urban, *Issues in the Design of Flexible and Dynamic Service-Oriented Systems. Proc of SDSOA'07: ICSE, IEEE Computer Society Washington (2007).*
 47. S. Dustdar, H.Gall and R. Schmidt, *Web services for groupware in distributed and mobile collaboration. Proc of the 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing, 2004, pp 241-247, IEEE.*
 48. H. Fuks, A.B Raposo, M.A Gerosa and C.J.P de Lucena, *Applying the 3C model to groupware engineering International Journal of Cooperative Information Systems, v. 14, n. 2-3, pp 299-328, 2005*

- 38 *Nader Cheaib, Samir Otmane, Malik Mallem*
49. H. Fuks, A.B Raposo, M.A Gerosa, M. Pimentel and C.J.P de Lucena, *The 3C collaboration model Journal of the Encyclopedia of E-Collaboration, Ned Kock (org), pp 637-644, 2007*
 50. F.F Oliveira, J.C.P Antunes and R.S.S Guizzardi, *Towards a collaboration ontology Proc. of the Snd Brazilian Workshop on Ontologies and Metamodels for Software and Data Engineering,*
 51. I.E Foukarakis, A.I Kostaridis, C.G Biniaris, D.I Kaklamani and I.S Venieris, *Web-mages: An agent platform based on web services Journal of Computer Communications, Volume 30, Number 3, pp 538-545, 2007*
 52. M. Matskin, P. Kungas, J. Rao, J. Sampson and S.A Petersen, *Enabling Web services composition with software agents Proceedings of the Ninth IASTED International Conference on Internet and Multimedia Systems and Applications, IMSA 2005, Honolulu, Hawaii, USA, pp 93-98, 2005*
 53. B. Starr, M.S Ackerman and M. Pazzani, *Do-I-Care: A collaborative web agent Proc. Human Factors in Computing Systems (ACM CHI), pp 237-274, 1996*
 54. Z. Maamar, Q.Z. Sheng and B. Benatallah, *Interleaving Web Services Composition and Execution Using Software Agents and Delegation The 1st International Workshop on Web Services and Agent-based Engineering, Sydney, Australia, 2003*
 55. P.A Buhler and J.M Vidal, *Toward the synthesis of web services and agent behaviors Proceedings of the Agentcities: Challenges in Open Agent Environments Workshop, pp 25-31, 2002*
 56. T.X Nguyen and R. Kowalczyk, *WS2JADE: Integrating Web Service with Jade Agents Technical Report, SOCAB0, 2005*
 57. M.N. Huhns and M.P Singh, *Service-oriented computing: Key concepts and principles Journal of IEEE Internet Computing, V 9, Number 1, pp 75-81, 2005*
 58. I. Dickinson and M. Wooldridge, *Agents are not (just) web services: considering BDI agents and web services Proceedings of the 2005 Workshop on Service-Oriented Computing and Agent-Based Engineering (SOCABE'2005), Utrecht, The Netherlands, 2005*
 59. G. Bourguin, *Lessons learned from the implementation of a reflexive groupware system Proceedings of the 15th French-speaking conference on human-computer interaction, pp 40-47, 2003*
 60. D. Payet, *L'enrichissement de message comme support pour la composition logicielle PhD Thesis, Unviversity of Montpellier 2, 2003*
 61. R. Slagter, M. Biemans and H. Ter Hofte, *Evolution in use of groupware: facilitating tailoring to the extreme Proceedings. Seventh International Workshop on Groupware, pp 68-73, 2001*