Integrating Internet Technologies in Designing a Tailorable Groupware Architecture

Nader Cheaib, Samir Otmane, Malik Mallem

Integrating Internet Technologies in Designing a Tailorable Groupware Architecture

Nader Cheaib, Samir Otmane and Malik Mallem
IBISC CNRS FRE 2873, Université d'Evry, 91020 Evry Cedex, France
nader.cheaib@ibisc.univ-evry.fr, samir.otmane@ibisc.univ-evry.fr, malik.mallem@ibisc.univ-evry.fr

Abstract

In this article, we propose an approach to introduce tailorability in the design of groupware, as the approaches already existing are still ambiguous in putting it forward in CSCW systems. We will present a brief overview on some approaches that deals with tailorability in this field. Then, we will make use of concepts and notions from each in order to integrate them in an innovative, value-added and tailorable architecture. We will discuss the purpose of integrating internet technologies with software agents while putting it forward in the context of tailorable groupware design.

Keywords: Groupware Design, Tailorability, Service Oriented Architecture, Software Agents.

1. Introduction

As the use of the Internet and services offered with it is emerging more and more, people are in an increasing need of flexible and agile applications. 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 problems. The aim of CSCW (Computer Supported Cooperative Work) is to find ways for groupware to enhance collaboration between individuals. For [14], 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.

1.1. Need for Tailorability

The research about tailorability originated from the gap between design and use of collaborative systems. Making the system, its interfaces and the services that they could offer tailorable for 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 to users the possibility to adapt the application based on their needs and not the other way around.

Various authors have tried to study the notion of tailorability [1, 2, 14]. However, this definition is still ambiguous and lacking in most systems (multi/single user), where users' needs when collaborating could multiply rapidly (audio, video etc.), and the need for a generic and tailorable architecture to ensure interoperability and ease of integration remains significant.

In this paper, we will study some approaches found in the literature for the design of tailorable groupware architecture. Our aim is to understand how this notion is utilized, extracting advantages of some approaches in order to design a new architecture for collaborative applications that will be totally tailorable. The paper will proceed as follows: First, we will give some definitions of tailorability. The second part will be dealing with some approaches for tailorability in groupware, and in the last part, we will introduce our own approach, which combines some interesting concepts found in the later approaches. We will talk about the on-going and recent research field of web services and software agents' integration, and how we could put it in the context of groupware tailorability. Finally, we summarize the main ideas and give a short overview for further work in the field.

2. Tailorability Approaches

Various authors have tried to define tailorability for groupware. The authors in [6] underline that a tailorable application is at the same time reusable and modifiable by its own users, and the activity of its redefinition is one of the facets of its utilization. Other authors [1] define a tailorable application as a system that can be adapted properly according to changes and the diversity of users' needs, or [3] that defines tailorability as the capacity of an information system to allow a person to adjust the application based on personal preferences or different tasks. For [14] tailoring is the continued development of an application by making persistent modifications to it. It is in fact initiated in response to an application being inefficient or difficult to use. Clearly, tailorability is a crucial property for groupware applications, but the question remains of how this notion can be implemented, in particular for users that are not necessarily specialist in designing software applications.

Various approaches aiming to integrate tailorability in CSCW systems have received much attention in the literature [1, 2, 3]. However, most of these approaches apply only to certain specific domains, as support for synchronous groupware, workflow-based or collaborative writing, and it is not certain whether these approaches could be applied to

generic domains as well. In our research, we found that introducing tailorability in the design of groupware is still very limited and theoretical, as there exist various approaches without a sufficient support for comparison and classification. For this reason, we thought that providing a global view on some of these approaches is already a contribution to CSCW domain. In the rest of this paper, we will begin by building a global view on some approaches for tailorability in CSCW systems. We will mention respectively the activity theory approach [6]; component-based [1]; building blocks [3] and SOA [9] (Service Oriented Architecture) approaches. Finally, we will present our own approach based on the later approaches.

2.1. Activity Theory Approach

The author in [2] justifies that tailorability possesses a theoretical foundation enabling to apprehend it using fundamental properties of human activity. He proposes a set of properties for constructing a conceptual model for a generic environment of CSCW systems, based on a fundamental theory, reflectivity**.** This environment is called DARE (Distributed Activities in a Reflexive Environment) [6]. In the realization of DARE, a framework is proposed based on the concepts and mechanisms of the activity theory, which permits to distinguish two essential properties of the human activity:

- **Reflexivity,** that enables to access and modify the structure of the application during its execution.
- **Crystallization** or the reutilization of user's experiences. These experiences could be, for example, a specification of roles in a particular activity.

Based on the activity theory, all mediator elements influence the course of activity and thus it is impossible to predict its impact on a certain activity [2]. This is why, for the author, the tool should be considered a fully mediator element, meaning that if it could influence the collaborative activity, then it should be modified by it. The author was inspired by the Meta-Object Protocol (MOP) [6] for realizing DARE, as the reflexivity takes place with the introduction of a meta model whose main entity is the 'task', that is a specification of the activity that describes the objectives, resources and roles that should take place in collaboration between actors.

2.2. Component-Based Architecture

A lot of research has been made for the design of component-based architecture for groupware [4, 1, 5]. The concept of a component-based architecture is independent of any application domain, and thus it is highly probable to adopt this kind of architecture to integrate tailorability in the design of groupware [4]. In a component-based approach, a groupware is designed as a collection of components in which they could be added, modified, or deleted. This type of applications will be able to support the evolution that tailorability tries to introduce. The authors in [4] argue that an ideal collaborative system should be designed as a composable system where the integration of new components is build on top of a neutral basis. We will see here two component-based approaches, each using different ways and mechanisms to reach tailorability: A reflexive computational system [1] and building blocks architecture [3].

2.2.1. Reflective Computational System, The authors in [1] define a tailorable system as one that can be adapted for eventual modifications in its structure according to diversity of user's needs. The authors use the term adaptability to identify tailorability in its technical aspects. Here, the authors reused the notion of reflexivity in the activity theory seen in the first approach [2], by insisting that an adaptable application should include a representation of aspects of itself, and this self representation should be changeable by internal or external influences, and connected to certain aspects of the application. If the representation changes, the application changes as well, and only aspects included in the self representation of the application are susceptible to be affected by tailorability activities. As a simple example, consider an application with an initialization file that specifies the application's background color [1]. In this case, this initialization file is the self representation of the application, and the color is the adaptable aspect. This type of applications is seen as a "Reflective computational system". Note that a reflexive system is one that contains both representations of aspects of the real world, and representations of its own activities. In consequence, this type of application is capable of examining its own state and structure, and able to modify it according to user's and the context's needs. The causal relation implies that every modification of the (meta) representation is automatically shifted towards the behavior of the system.

2.2.2. "Building block" Architecture, The authors in [3] propose an approach based on building blocks for constructing tailorable CSCW systems. They argue that the evolution in the utilization of groupware is nowadays one of the main reasons for designing tailorable systems. In fact, the authors consider a tailorable system as one that permits for its users to perform modifications on the technical structure of the application after its implementation, according to their needs, personal preferences or different tasks. For the simple reason that all the modifications could not be predicted in the design phase by the application designers, it would be possible, according to the authors, to equip the users with means to accommodate these changes.

The authors introduce the concept of tailoring to the extreme [3]. This concept implies the extension of the set of functions in the system with new modules that could be dynamically integrated. An example of this concept is to permit the user to download modules from the internet and plug them directly into the system (plug-ins, widgets, etc.). However, this approach requires that functional modules (building blocks) should be analyzed before integrating them in order to determine the functions that they could offer and the way in which they will communicate and interconnect to other modules for minimizing interference in the system. The authors here insist that interoperability standards are therefore

essential between the building blocks that will be integrated into the system, probably resulting from different vendors, in order to standardize and facilitate the process of integration with other building blocks already existing, and therefore, insure the stability of the system as a whole. The authors implemented their concepts in CooPS (Cooperative People & Systems) [3].

2.3. Service-Oriented Architecture (SOA)

The demand for collaborative and flexible services is becoming more urgent as the competition in the marketplace is getting fiercer between service providers. For this reason, the authors in [9] propose the utilization of a Service-Oriented Architecture (SOA) for the construction of collaborative services. For the authors, SOA is becoming a new paradigm that aim at implementing loosely-connected applications which are extensible, flexible and integrate well with existing systems. Collaborative platforms have the potential of offering services on different layers of abstraction as their role is to offer a support tool for collaboration of activities [8].

SOA [8, 9] is a paradigm in full expansion that could be adapted to offer extensible services integrated in a platform for different users to collaborate between each other. Web services could facilitate the collaboration between groups or organizations, and can be defined by, for example, resource sharing, communication and interaction between collaborators (synchronous, asynchronous, communication channels etc.), virtual rooms, organization management (calendar, mail etc.). The support for web services offers interoperability between different collaborative or single-user systems [8], as they can be viewed as modular applications. The architecture considers a model of integrated services, where the interfaces of web services are described with a standardized definition language WSDL (Web Service Definition Language), and interact with each other using SOAP (Simple Object Access Protocol), while having their definitions saved in some norms of a web service catalogue using UDDI (Universal Description, Discovery and Integration).

3. Tailorable Design of Groupware

We base our approach mainly on the concept of a reflective system introduced in [6] and [1] (see paragraph 2.2.1). In addition, we use the concept of tailoring to the extreme [3] that relies on building blocks (that are Web services in our case), requiring them to be analyzed before integrating them into the system in order to discover the services they offer.

We also base our approach on SOA in designing collaborative services [9]. The use of SOA is mainly due to the interoperability that this approach offers. Using already standardized protocols, this approach will complement other approaches, and thus will combine the concepts of tailorable, reflective architecture with the concept of interoperability, for in consequence satisfying the maximum tailorability needed in the design of groupware. Finally, we introduce a hot research topic over the last years, which is web services' integration with software agents. We will see the advantages of using agents in conjunction with web services by attempting to integrate them into our approach. The integration of web services with software agents has the objective of giving the web services a proactive behavior in interacting with users. We will begin by describing our architecture, and then we will see how our approach integrates the later approaches and make use of each.

3.1. Tailorable Groupware Architecture

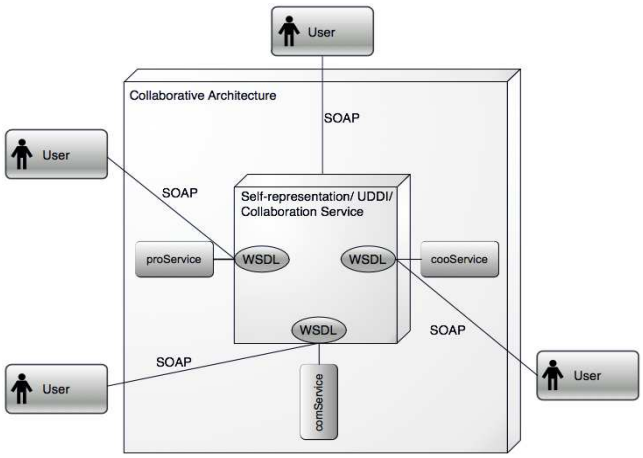

Figure 1: Design of our tailorable architecture

Figure 1 illustrates our architecture. The main square represents the boundaries of the system that contain the interfaces connecting users to the application. The square in the middle represents the self representation of the system. This self representation [1] is viewed in our approach as a norm of public directory that contains the list of all the services included in the system. This public directory is built using the protocol UDDI (Universal Description, Discovery and Integration), that is one of the core web services standards **[**12**]**. In other words, this self representation contains the definitions of services running in the system that are susceptible of undergoing tailorability activities by collaborating users. The definitions of these web services are provided using the standardized language for web services, WSDL (Web Service Definition Language), and interact with each other and the user using SOAP (Simple Object Access Protocol). For a more elaborate explanation on web services standards, please refer to [12]. The definitions included in the self representation of the system are connected to adaptable aspects, which are in our approach, the services themselves, as we can observe in Figure 1. These services can be considered as orchestrations of other services in the system [10], and include other services based on the functionalities they offer. In our approach, we distinguish three main categories of services: ComService, CooService and ProService:

• **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 between users, and codify their interaction (i.e. workflow).

• **ProService:** contains services that are the collaborative product of using the architecture. (Ex: paint application, word document etc.).

By classifying services in the system into three main categories (Communication, Coordination and Production), the three main spaces of the software collaboration process defined by the 3C model [11] are satisfied. Note that we use the term 'Production' to mean 'Cooperation' of activities (satisfying the terms used in the 3C model: Communication, Coordination and Cooperation).

3.2. Standards for Interoperability

By using SOA **[9]** as a basis to our approach, we insure interoperability between services in the application, and also between the user's needs and the system's capacity and performance. In fact, SOA offers three main standards that achieve interoperability: SOAP, UDDI and WSDL**.** We make use of the three standards in our architecture as follows:

• **SOAP** is a communication protocol written in XML, which permits to exchange data independently of the operating system used. To interact with the system, the user sends requests to the self representation of the application using SOAP messages.

• **UDDI** is a directory of web services' definitions called via the protocol SOAP. This type of protocol will implement the self representation of our architecture. In this way, the users interrogate the UDDI to know what are the services registered in the system, what type of functions they offer and the means to access them. UDDI implements 3 basic functions:

  • Publish: Lists the Web services' definitions in the self representation.

  • Find: Allows users to easily search for services using a search engine applied on the self representation of the system.

  • Bind: Insures the connection between a needed service in the system, and its clients.

• **WSDL** is used to list the definitions of the services in the self representation of the system (UDDI) that is be susceptible to be modified by tailoring activities. WSDL is also written in XML, listing the methods available, the messages formats of the services' interfaces and the way to access them.

3.3. Classic SOA Vs Tailorable SOA

In Figure 2, we can see the transformation of the classic SOA found in the literature to our vision of a tailorable SOA. In the classic SOA, there exist 2 actors: the service provider that registers the definitions of Web services (WSDL) in the public registry (UDDI). The user in this kind of architecture has only the possibility to send SOAP requests to interrogate the UDDI about a needed service, but does not have the possibility to modify the UDDI by adding new services that could better satisfy his needs. This limits the use and the flexibility of the approach, as users would only be limited to use the services already existing in the system, and thus wouldn't be able to adapt the application to their needs, but rather the other way around.

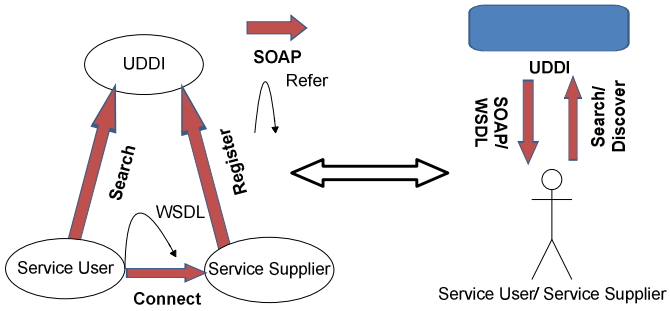

Figure 2: Classic Vs Tailorable SOA

In the tailorable SOA, we modify the structure of the classic SOA in a way that the service user is the service provider himself. In other words, the user will then have the privilege to interrogate the UDDI (self representation of the application in our approach) using standard SOAP requests, but also modify it using the same type of messages formats by directly plugging the new service definitions into the UDDI (i.e. drag/drop mechanism). The protocols provided (SOAP) in the SOA will be in charge of reconfiguring the links between the services added and the services already present in the system. In fact, the self representation part could be seen as an open implementation mechanism [7] where the users would be able to modify the structure of the application (inserting new service definitions through their WSDL files) without recompiling the system and stopping its execution. Also, this kind of system will satisfy the evolution of the use of the application due to temporal or behavioral changes. In this case, the classic Service-Oriented Architecture will be transformed into a tailorable Service-Oriented Architecture by giving the user tools to accommodate these changes.

3.4. Added-Value Tailorable Architecture

By integrating the three approaches described in the first section, we created an added-value architecture that introduces tailorability in collaborative applications to the fullest. In fact, we used the notion of reflexivity in [1, 6] by adopting the authors' view of a reflective system containing a representation of its own activities, and thus able to access and modify its structure according to user's needs. More specifically, we use the notion of self representation [1] that is viewed in our approach as a public registry containing definitions of services in the system connected to adaptable aspects of the application, that are the services themselves. This means that if the definitions of the services change, the services themselves change as well, and in consequence, the whole system would become tailorable by users. We than used the concept of tailorability to the extreme [3] that requires modules to be analyzed before integrating them into the system. This concept has also another essential requirement, which is the necessity of having interoperability mechanisms to allow reconfiguration of modules from different vendors with other functions constituting the system. We remedied this problem by using interoperability standards from the SOA approach [8]. Using these standards, the user will easily be

able to analyze the functionalities of the services [3] (by using the services' WSDL) before inserting them into the self representation of the application [1]. Thus, by combining the three concepts found in the literature (reflexivity, tailoring to the extreme and SOA), a tailorable architecture is emerged satisfying properties of interoperability [8], openness [7] and flexibility that are in our opinion, essential requirements for CSCW systems.

3.5. Software Agents and Web services Integration

For [18], current techniques for publishing and finding services (such as WSDL and UDDI) rely on static descriptions of service interfaces, forcing consumers to find and bind services at design time. On the other hand, web services are becoming one of the most important architectures used in heterogeneous cooperative information systems, as it was the appearance of Web services that permitted internet sites to offer services in a more flexible manner [14]. However, the concept of software agents is even older than web services, and it has been employed with success for executing distributed applications. Agents are defined briefly as a piece of software that acts autonomously to undertake tasks on behalf of users. For [16], it is based on the fact that users only need to specify a high-level goal instead of issuing explicit instructions, leaving the *how* and *when* decisions to the agent. The same authors say that software agents exhibit a number of features that make them different from other traditional components including autonomy, goal-orientation, collaboration, flexibility, self-starting, temporal continuity, character, communication, adaptation, and mobility.

The reason behind our motivation to integrate software agents with web services is driven by the fact that agents put in practice the concept of mobile code, and through coordination with their flexible architectures, can easily be adapted to highly dynamic and heterogeneous environment as the web. Web services however are the fast emergence of dominant means for connecting distributed applications through well established internet protocols.

Software agents can be one of the essential developments to web services for the fact that they are functional entities instead of being just simple interaction delegations or communication means [15]. The idea in our design of tailorable architecture is to explore the capacities of agents' proactive interactions to enhance the behavior of web services in a service-oriented architecture (SOA). With this paradigm, software components, where each one is representing a service and an agent in collaboration, can interact with each other for providing unified services in a specified environment, as for example the exchange of multimedia applications in a virtual environment (we are currently working on such system, we call it Oce@nyd). This is aligned with the authors in [15]: "*agents will become an essential part of most Web-based applications, serving as the* '*glue*' *that makes a system as large as the Web manageable and viable*."

3.6. Purpose of integration

For [15], the purpose of the combination is to integrate agents and web services technologies into a cohesive entity that attempts to surpass the weakness of each technology, while reinforcing their individual advantages. This integration can be proposed on the design and implementation level, where on the design level, web services are encapsulated as semi-autonomous agents that can be employed for describing the external behaviors of software agents, and where every agent works in relation to the environment as a regular web service. In consequence, agents can be used to establish high-level, flexible interaction models, and the web services will be more appropriate for resolving the problem of interoperability of diverse applications in concrete realizations. At the execution level, UDDI WSDL and SOAP will provide capacities as the discovery, deployment and communication. Eventually, by integrating web services and software agents in the context of groupware tailorability, we introduce a totally innovative view of a groupware architecture design, offering tailorability at the system's level, where the system can be tailored by dynamically integrating agents with web services, thus offering to users tailoring capabilities. Software agents will be responsible for dynamic reconfiguration and discovery of services, along with openness and flexibility already satisfied by our architecture conceived from various tailoring approaches found in the literature. Eventually, by identifying these technologies, implementing real tailorable architecture will shift from theory to real practice.

3.7. Use of Agents in our architecture

Taking this into consideration, dynamic service selection needs an agent-based solution. Agents can represent autonomous service consumers and providers as well as collaborating to dynamically configure and reconfigure services-based software applications. In our architecture, the agent can play the active role of a consumer. That is, whenever a consumer application using the system needs to use a service, it employs its agents to communicate with the service. For each service, the architecture will create a service agent that exposes the service's interface, augmented with functionality to capture the consumer's preferences or needs and to query other agents for a suitable match [17]. The agent can determine objective attribute values (such as reliability, availability, and request-to-response time) on its own and gets user feedback for subjective attributes (such as the user's overall experience). The architecture will have a self-performance reliable data in which it could use to calculate the degree of tailorability offered to the user along with its performance capabilities according to user's satisfaction in delivering the needed services.

3.7.1. JADE and Web services, JADE (Java Agent DEvelopment Framework) is a middle-ware implemented in Java which simplifies the implementation of agents complying with the FIPA specifications [18] through a set of graphical tools that supports the debugging and deployment phases. JADE agents use ACL (Agent communication Language) to

communicate between each other, which is analogous to the SOAP protocol used by Web services. We rely on the approach in [17] presenting a Web service agent framework along with the approach in [18] for integrating web services with JADE agents, providing common means to dynamically invoke instances of each other at run-time. Another approach used is to allow the two platforms to evolve in parallel without imposing restrictions on each other, hence accepting equity between Web services and agents' roles. To do this, a module between the two platforms should exist translating ACL messages to Web service invocations, and vice versa. This module is registered as a special agent service in FIPA DF (Directory Facilitator in JADE) and a special Web Service endpoint in UDDI directories, so when an agent wants to invoke a Web service, the request is passed to this particular module to perform the actual Web service invocation. This reflects the assumption that Web services need to be registered before they can be discovered, which is true for a model like UDDI but does no longer hold in recent P2P models [18].

4. Conclusion

In this article, we gave a brief overview on some approaches that try to implement tailorability in designing CSCW (Computer Supported Cooperative Work) systems. Moreover, we described a theoretical foundation for a collaborative platform using Internet technologies to be put forward in the domain of groupware tailorability, giving concrete tools for its implementation: A synergy of tailoring concepts put together to arrive to a component-based, service-oriented architecture. Software agents are to be used enhancing the functionalities of web services by giving them a proactive behavior. However, we should say that the utility of agents can be limited when only considering the standard web services protocol stack without semantic annotations. Hence, we expect to expand our work by integrating tools to manipulate semantic Web service descriptions.

We are working on the implementation of a multimedia application (**oce@nyd)**, enabling users to share digital information such as photos and audio/video recordings in order to enrich simultaneously and in collaboration maps of underwater sites. Our architecture for designing groupware will be applied to the later multimedia environment. Moreover, experiments are taking place for testing agents' integration with web services capabilities in a JADE environment. Our aim again is to provide users with powerful mechanism for dynamically tailoring the services offered in the platform, and hence, enhance collaboration.

ACKNOWLEDGMENT

The implementation of this work is a part of a national project **DIGITAL OCEAN**, which has the objective of creating an innovative mode for the distribution of multimedia applications. This project is supported by the National Agency of Research in France (ANR).

REFERENCES

[1] O. Stiemerling and A. Cremers, "Tailorable Component Architectures for CSCW- Systems"**.** *Proceedings of the 6th euromicro Workshop on Parallel and Distributed Programming*, January 1998, pp. 21-24

[2] G. Bourguin, "Les leçons d'une expérience dans la réalisation d'un collecticiel réflexif ". *Actes de la 1ère Conference francophone* IHM 2003, pp. 24-28.

[3] R. Slagter, M. Biemans, and H.T Hofte, "Evolution in Use of Groupware: Facilitating Tailoring to the Extreme". *Proceedings of the Seventh International Workshop on Groupware, CRIWG*2001, pp. 68-73.

[4] M. Roseman and S .Greenberg, "Simplifying Component Development in an Integrated Groupware Environment"**.** *Proceedings on the 10th annual ACM symposium on User Interface software and technology*, NY, USA 1997, pp.65-72.

[5] R. Slagter, H.T Hofte and O. Stiemerling, "Component-Based Groupware: An introduction**".** *Proceedings on the CSCW2000 Workshop on Component-based groupware*, Volume 2, 2000.

[6] G. Bourguin, "Un support informatique à l'activité coopérative fondé sur la Théorie de l'Activité le projet DARE"**.** Thesis in computer science, University of Lille, France 2000.

[7] G. Kiczales, J. Lamping, C. Lopes, C. Maeda and A. Mendhekar, "Open Implementation Design Guidelines". *Proceedings of the 19th international conference on Software engineering*, ACM Press NY, USA 1997, pp.481-490.

[8] I. Jorstad, S. Dustdar and D.V Thanh, "A service Oriented Architecture Framework for Collaborative Services", *Proceedings of the 14th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise*. Copyright IEEE 2005, pp.121-125.

[9] S. Dustdar, H. Gall and R. Schmitt, "Web services for Groupware in Distributed and Mobile Collaboration". *12th Euromicro Conference on Parallel, Distributed and Network-Based Processing*, 2004, pp.241-247.

[10] C. Peltz**,** "Web Services Orchestration. A review of emerging technologies, tools and standards"**.** *Hewlett Packard White Paper*, January 2003.

[11] C.A Ellis and J.A Wainer, "Conceptual Model of Groupware", *ACM Conference on Computer Supported Cooperative Work (CSCW94)*, ACM Press New York, USA, 1994, pp.79-88,

[12] E. Newcomer, "Understanding Web Services: XML, WSDL, SOAP and UDDI", *Addison-Wesley Professional (E)*, 2002.

[13] A. Fernandez, "Groupware for Collaborative Tailoring". *Thesis in computer science, des Fachbereichs Informatik der FernUniversitat in Hagen*. Hagen, April 2005.

[14] I.E**.** Foukarakis**,** A.I. Kostaridis**,** C.G**.** Biniaris**,** D.I. Kaklamani and I.S. Venieris**. "**Webmages: An agent platform based on web services"**.** *Computer Communications Journal, Volume 30, Issue 3 2007, pp. 538-545.*

[15] W. Shen, Q. Hao, S. Wang, Y. Li and H. Ghenniwa, "Agent-based service-oriented integration architecture for collaborative intelligent manufacturing**"***. Robotics and Computer-integrated Manufacturing Volume 23, Number 3*, June 2007, pp. 315-325.

[16] Z. Maamar, Q.Z Sheng and B. Benatallah,"Interleaving Web Services Composition and Execution Using Software Agents and Delegation". *AAMAS2003 Workshop on Web Services and Agent-Based Engineering. July 2003.*

[17] E.M Maximilien and M.P Singh. "A Framework and ontology for dynamic web services selection". *Internet Computing IEEE, Volume 8, Number 5*, 2004, pp.84-93

[18] T.X Nguyen and R. Kowalczyk. "WS2JADE: Integrating Web Service with Jade Agents" Technical Report, *SOCAB05*, Springer 2005.