An automatic regulation mechanism based upon roles and rules: the VirtusCharte approach

Henri Eberspächer[1], Michelle Joab[2]

[1] PUCPR – Pontifical Catholic University of Parana, Curitiba – PR, Brazil

[2] LIRMM - Université Montpellier II/CNR , Montpellier, France

henri.eberspacher@pucpr.br, michelle.joab@lirmm.fr

Abstract. *In this paper we introduce Virtus, a virtual learning environment that provides enhanced functionalities to automate the management of users, groups and virtual communities using an expert rule-based system. Virtus supports collaborative learning using a group contract by automating the execution of the rules of the contract. The originality of our work consists, on the one hand, in proposing a declarative language to express contracts using declarative rules, commitments and role responsibilities assigned to members of a group or participants of a community and, on the other hand, in automatically executing the contract, implemented by an expert system module (VirtusCharte). Periodically, Virtus platform extracts information from the virtual environment and activates a knowledge-based system for each group.*

Resumo. *Este trabalho apresenta o sistema Virtus, um ambiente virtual com funcionalidades para automatizar o gerenciamento de usuários, grupos e comunidades virtuais usando regras de produção. Virtus oferece suporte à aprendizagem colaborativa através de um contrato de grupo, automatizando a execução das regras deste contrato. Um ponto chave deste trabalho é a proposição de uma linguagem declarativa para expressar os contratos através de regras que atribuem compromissos e responsabilidades aos membros de um grupo (ou participantes de uma comunidade) usando papéis. Outro ponto relevante é a aplicação automática do contrato, implementado por um mecanismo de regulação baseado em regras (VirtusCharte).*

1. Introduction

Virtual communities with learning or working (or both) purposes generally use collaborative learning and cooperative work in order to improve the benefits of learning/working together and to improve the effectiveness of the social interaction. A community whose organization and methods are based on these techniques normally obtains better results than the ones where individuals are working by themselves (Dillenbourg et al., 1996).

In this direction, we propose a contract model with a role-based coordination approach to support individual and group management in collaborative learning activities. The main idea is to create a contract that can be used to automate some substantial administration work and to both create and maintain mutual commitments between users and their working groups. Our aim is to increase individual and collective efficiency

adding an automatic management service which reminds the users their commitments in a personalized way and applies the terms of the contract (Eberspacher and Joab, 2008). By using explicit contracts in collaborative learning scenarios we intend to encourage the users to commit themselves in the learning process (Zhu, 2004). By defining roles within groups, responsibilities between members can be shared. This sharing is established through the contract, which describes the responsibilities and formalizes the rights and duties of each role. This contract, freely discussed in the group and then adopted, boosts the commitment of members for the tasks ahead (Zhu, 2006).

Our approach includes the management of groups gathering a restricted number of members around a common project which achievement is made possible through collaboration.

In this paper we present Virtus, our virtual learning environment that implements our proposal. Virtus is a web-based platform that support collaborative learning activities in groups using a rule-based contract (Eberspacher and Joab, 2006).

The Virtus platform consists of two modules: VirtusWeb which manages the services of the on-line environment (like usual Learning Management Systems or Groupware Systems) and VirtusCharte which implements an Intelligent Support System to support users following the rules of the contract. This means that the Virtus platform supplies a regulation and monitoring mechanism for group activities based on the concepts of contract (a set of rules) and roles. This mechanism is fully configurable through the rules expressed in the proposed contract language using freely defined roles. VirtusCharte is based primarily on an intelligent tool that will automatically support collaborative work using VirtusWeb.

According to Brusilovsky and Peylo (2003), Adaptive Systems differ from Intelligent Systems. Adaptive Systems attempt to be different for different users and groups of users by taking into account information accumulated in the individual or group models. Intelligent systems apply techniques from the field of Artificial Intelligence (AI) to provide broader and better support for the users of Web-based educational systems. Within these definitions, Virtus is more an intelligent system than an adaptive system because it does not build a model of the group while using the contract. According to Brusilovsky's classification,Virtus could be linked to Intelligent Collaborative System as it coaches and monitors individuals (tasks) and groups (activities); Virtus acts as an intelligent collaboration support to promote collaborative learning using contracts.

In the medium term, the goal is to achieve an adaptive environment in the sense that VirtusCharte will run a contract originally adopted by a group and monitor how the members of the group complies with the terms of the contract when performing an activity. If a discrepancy is found between the contract prescribed and the actual behavior of the individual (or his/her group), VirtusCharte could propose modifications to the contract in order to be more in line with the members' behavior (Paschke, 2005). The contract is then re-negotiated and revised by the group members.

This paper is structured as follows. Section 2 describes how to monitor a group according to a contract and section 3 discusses some related work. Then, section 4 introduces Virtus, our intelligent web-based platform. Finally, in section 5, conclusions and further work are presented.

2. Monitoring a group according to a contract

The contract is a set of basic rules of an organization; it expresses the rights and duties of the persons to whom they apply.

The key point is the dynamic process between VirtusWeb, where users are performing activities and VirtusCharte, which monitors the way the members honor their commitments.

In the VirtusWeb module, once a group is created, the members negotiate and then adopt a contract and then perform traceable activities (doing tasks), as shown in the first column of Figure 1.

In the VirtusCharte module, the trace is analyzed with regard to its compliance to the current contract. As a result, a revised contract can be proposed to the group, as shown in the third column of Figure 1.

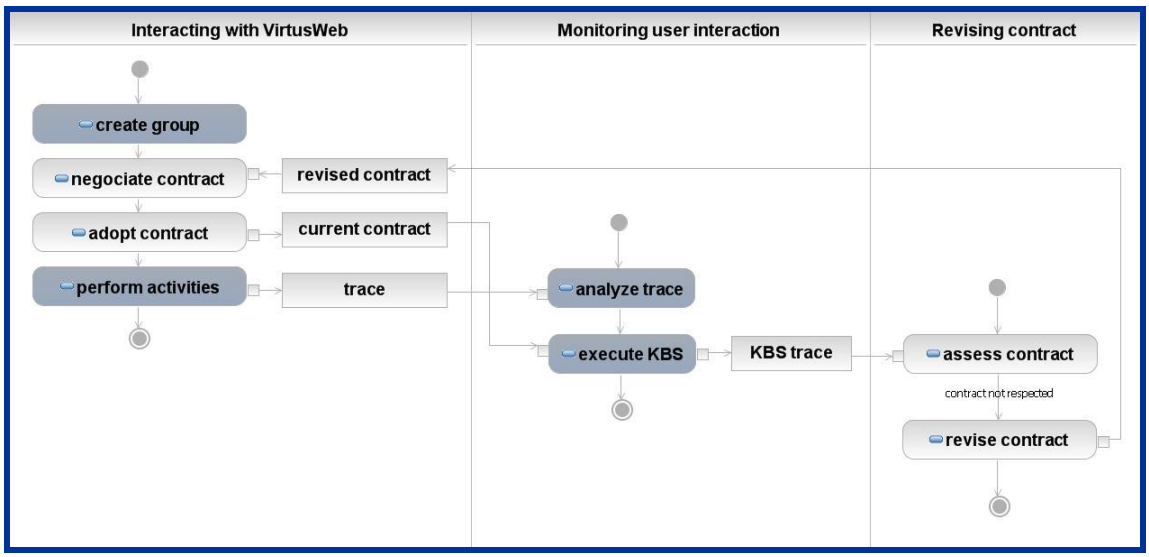

Figure 1. Activity diagram of the group life cycle and the contract

Our proposal is based on the relation between rules and roles:

- *rules* mean a guideline, a formula that indicates what needs to be done within a certain context,
- *roles* mean that each person is identified with a role, so that he/she can quickly identify his/her participation, prerogatives and commitments A role is a semantic construct forming the basis of access control policies. We use the term "role" in accordance with Zhu (2004, 2006): a role is a set of requirements that define how a member should behave in a group.

Using the charter, the users will benefit from an automatic group management support mechanism. As a human user, VirtusCharte may act on VirtusWeb working environment and could also recall the users' commitments by sending notifications. These actions and/or notifications are obviously a consequence of the group charter execution. To avoid black box behavior, VirtusCharte actions are emphasized as executed by a virtual user.

The rules are based on the concepts of the learning environment: resources, activities, events and roles which categorize the members of a group.

The declarative language of contract is able to describe how to monitor the status of an activity and the actions carried out on an activity (such as postponing an activity deadline). Using the rules, the author of the contract expresses, for example, how to react in cases where there is a procedure that is non-compliant with the behavior expected for the role. The assigned user's roles indicate the responsibilities and expected actions of each group member. The rules could question the work carried out by other roles or those that are performed outside the specified time constraints. The rule pattern presented in Table 1 specifies the contract on these points.

Table 1. A rule pattern example of the proposed contract language

Pattern 1	**Controlling actions on an activity**

If	// mandatory premise
	an activity is **<changed** | **postponed>**
	
	// optional premises
	[by (**<role>**| virtus **]**
	[<J> days before deadline**]**

then	*// possible actions* **[** undo change **]** **[** suspend activity **]** *// possible notifications* **[** notify the creator of this activity **]** **[** notify the author of the change **]** **[** notify the person responsible for each input resource **]** **[** notify the person responsible for each output resource **]** *// possible if it is an group activity* **[** notify the **<role>** of the group **]** **[** notify all group members **]**

Starting from the same pattern presented in Table 1, the author could write several rules as shown in Table 2.

The strategies chosen for the coordination mechanism are more suitable to support asynchronous collaborative working and long-term collaboration, i.e. collaboration that takes a long time and occurs at different times and at different places. Virtus is domain independent and could be used in a wide range of pedagogical experiments in computer-supported collaborative learning. In order to use a specific pedagogical strategy the tutor needs to include specific roles in accordance to the educational goals of the activities that will be carried out by the group.

Table 2. Examples of rules created using the rule pattern of Table 1

```
rule 1 ::
if
        an activity is postponed
        by secretary
        5 days before deadline
then
        notify the creator of this activity
        notify the leader of the group
endrule

rule 2 ::
if
        an activity is postponed
        by secretary
        1 day before deadline
then
        suspend activity
        notify the creator of this activity
        notify all group membres
endrule
```

3. Related work

Group management includes a wide range of possible control points, working methods and pedagogical strategies that enable several different architectural and functional approaches. Our work concerns group formation and regulations aspects using roles and rules expressed in terms of a group contract.

On the one hand, in the L3 project (Lifelong learning as a utility) with the IPoC - Intended Point of Cooperation, Wessner and Pfister (2001) emphasize the integration of collaborative learning into the learning environment so that knowledge about the collaboration context can be used to support the collaboration, principally in group formation.

On the other hand, the "participation model" takes into account the social aspects of collaborative work (Martel, 1998). It is a conceptual model to describe joint activities, their relationships of dependence and the structure of exchanges within the group. Moreover, this regulation approach has been used and developed by Ferraris et al. (2002) to construct collaborative pedagogical situations using scenarios and roles; as earlier presented in the collaborative drawing application for young children (Ferraris and Martel, 2000). This same approach is also used by Mezura-Godoy and Talbot (2001) to propose a framework of regulation components and a component management service for enabling users to develop regulated collaborative applications. The regulation components' main features are the rules (work rules, norms and constraints), the types of interactions (synchronous or asynchronous), the tools (regulative or not), the roles (thematic or causal) and the objects (means of communication or product of collaboration).

Considering the notification strategies, Shen and Shun (2002) proposed a flexible notification framework in which notification policy is separated from notification

mechanism. In the policy part, they used two parameters, frequency and granularity, to define a spectrum of notification polices. In the mechanism part, separate notification buffers and separate notification executors were used to support various out-going/incoming notification policies.

The works cited above perform certain strategies in group formation, role-based social regulation and notification strategies respectively that give us some perspectives. However, group managing is considered at some level of abstraction that does not allow the execution of automatic services. Our goal is to consider these points and to propose a tailored architecture for automatic group management using group contracts.

4. Virtus environment

We define a learning environment running on Virtus platform as a session in which a user uses the functionalities of a service in a given context.

The context is structured on three levels: (i) the individual (private user space), (ii) the group, and (iii) the community. Everyone has his own private space and can participate in public spaces structured in groups within communities. An object always exists in a context and it is always handled in functional spaces categorized in services. Currently, Virtus provides three types of services: (i) activity management, (ii) event management, and (iii) resource management.

As a result, we have, on one hand, VirtusWeb an environment where activities, events and resources are created, viewed and edited, and on the other hand, VirtusCharte, a system that interacts with these objects, applying the terms of the contract. VirtusCharte must be able to consult, edit and modify them using the contract in a specific context. By separating the two software modules of the platform we aim to run the contract mechanism independently of a particular platform. This means that VirtusCharte could, with some adaptations, be plugged in another LMS. This will be effectively applied in the system's design and in the prototype implementation.

4.1 Virtus platform architecture

Figure 2 shows the Virtus architecture with his two modules: VirtusWeb and VirtusCharte. The information stored in the relational database will be used by VirtusCharte to build its working memory (the facts), to which the rules of the contract would be applied. The execution of the contract can create and modify facts; these are consequently inserted in VirtusWeb relational database. The prototype was developed using client/server architecture and the platform is based on three-tier architecture.

4.2 VirtusWeb

It is important to mention that the different contexts are presented in the same user interface. In practice, a user views his/her events, activities and resources in a single interface that shows all the groups and communities in which he/she is engaged. All functionalities take into account the three contexts (a user could participate in several groups and communities playing different roles).

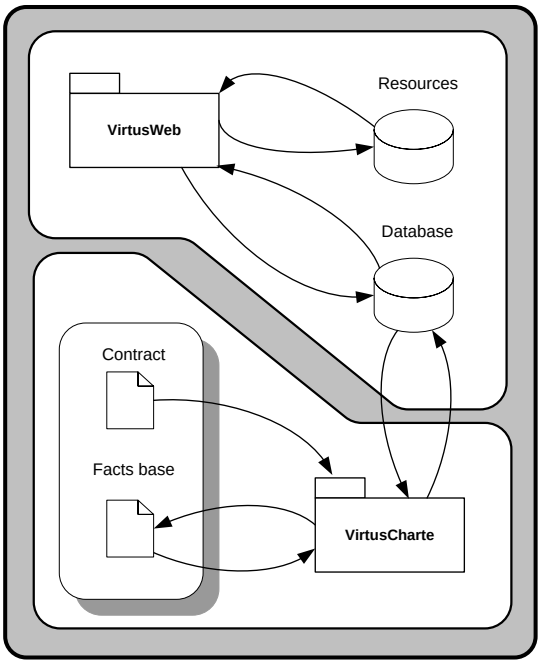

Figure 2. Virtus general architecture

The distinction between private entries, group entries or community ones is permanently highlighted in a single frame. Figure 3 shows a screenshot of VirtusWeb at the weekly view of the user calendar (including events and activities).

For the development of VirtusWeb, we used free software that is commonly used in open-source LMS platforms. We adopted a LAMP platform: Linux operating system; Apache to the web server; MySQL as Database Management System (DBMS) and PHP for the scripting language to generate dynamic Web pages and communicate with the MySQL server.

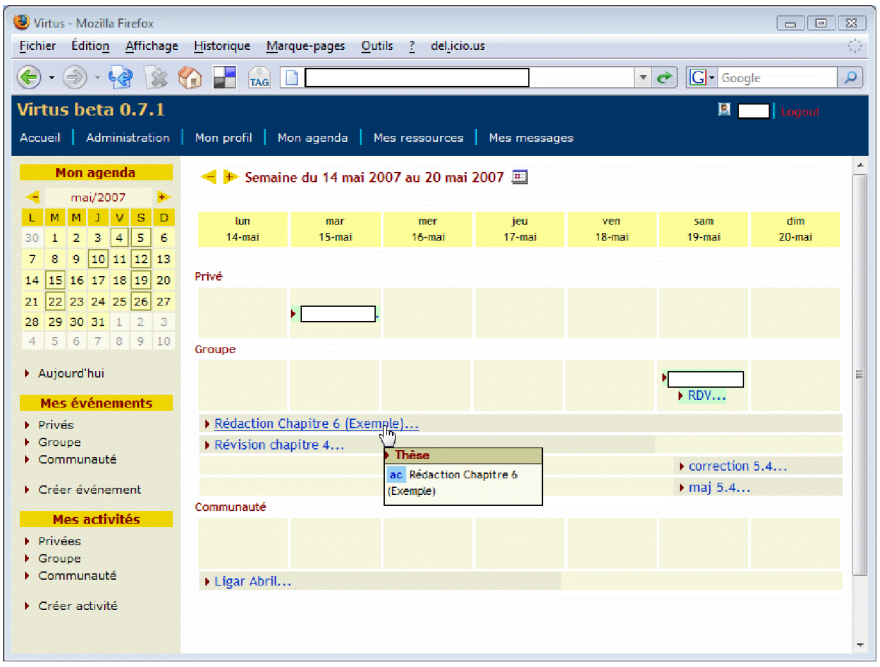

Figure 3. VirtusWeb calendar interface

Each time an activity is published, a new version is created. The user can monitor the progress with the option 'show changes', which proposes a history of changes and shows any previous version of the activity. Figure 4 shows the different versions of an activity and illustrates the sequence of statements from its insertion until the completion of the activity.

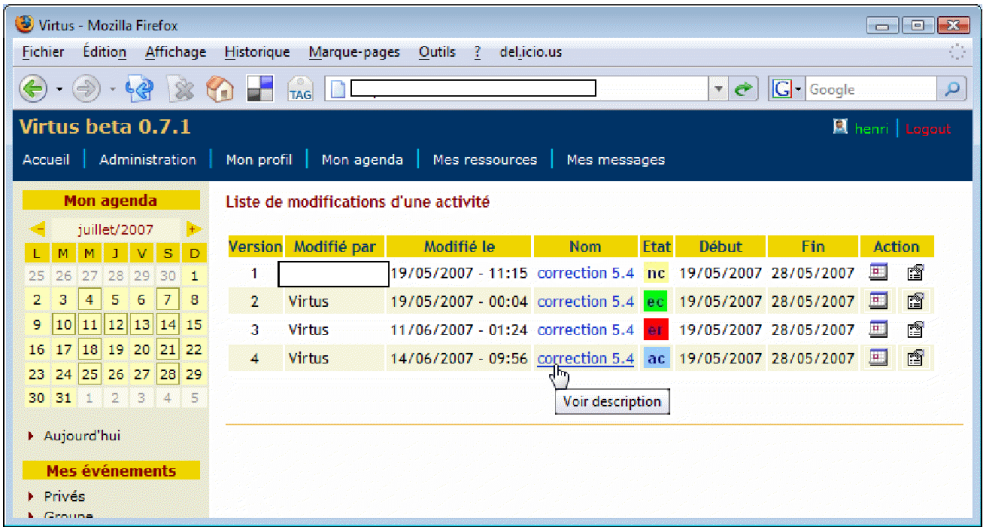

Figure 4. Different versions of an activity

4.3 VirtusCharte

VirtusCharte has been implemented in Java and uses the same classes as VirtusWeb with regard to the model of the system. We use Hash Map data structures in order to create lists with all instances of objects handled by the services and information that make up the analyzed context. We used JESS (Friedman-Hill, 2003), a Java Expert System Shell that has been developed in Java to build the knowledge-based system used in VirtusCharte.

For each execution of the knowledge-based system in a private, community or group context, VirtusCharte built the initial facts recovering them with a selective loading algorithm importing them from the VirtusWeb database. The actions undertaken by the methods of VirtusCharte are performed using web services in a session where VirtusCharte becomes a user of VirtusWeb.

An execution of VirtusCharte carries, in order:
i) querying the database with a selective load;
ii) creating templates defining the facts;
iii) populating the fact base;
iv) loading the contract in JESS (a file with the rules);
v) executing the knowledge-based system;
vi) executing actions and notifications according to the inferences.

At the end of the knowledge-based system inference cycle, VirtusCharte verify the working memory (which was changed by the rules) to find the facts which indicate actions to be undertaken on the Virtus platform and notifications to be sent (e-mail to the users).

Without caution, the working memory could generate a problem of exponential growth for the pattern matching inference engine. To address this problem, the working

memory must be loaded selectively to reduce its size. This selection is made according to the attributes of the analyzed context and services that are in use in a VirtusWeb session.

Once VirtusCharte has loaded the required information, it creates its own lists of objects and deals with this information to state the necessary facts to start the expert system.

The selective load is based on the following principles:

- the context addressed to be analyzed;
- the lifecycle of the communities and groups involved;
- the open states of events and activities;
- the life of terminated objects;
- the time window considered and configured in the contract.

5. Conclusion and further work

The Virtus Platform is composed of VirtusWeb which manages the contexts and services of the learning environment and of VirtusCharte which reasons automatically in terms of the rules of the contract. The Virtus platform has a regulation and follow-up mechanism based on the concepts of charter and role. This mechanism monitors the activities of groups and communities.

The user is free to act in the VirtusWeb environment but the virtual user Virtus can oppose the users or support them. Virtus acts as a group member to whom the charter has given special regulation rights. Virtus (as a virtual user) has the same limits as a human user.

The originality of our work consists in proposing a declarative language to express charters using declarative rules, commitments and role responsibilities and in automatically executing the charter, implemented by an expert system module. Once the charter is defined for a group, it is translated into an executable rule language. Periodically, the Virtus platform, resulting from our work, extracts information from the learning environment and activates a knowledge-based system for each active group in the virtual learning environment.

In order to facilitate contract specification and to improve final user and system administration interaction, we are developing a new version of VirtusCharte using Drools (Bali, 2009; Browne, 2009) as the built-in knowledge-based system. This improvement will allow us to use a DSL (Domain Specific Language) to express our declarative contract language (Bravenboer and Visser, 2004).

As further work, we propose to monitor and assess how the effective behavior of groups matches the terms of the contract in order to automatically propose a revised one to the group. It means we need to monitor the contract performance at run-time and not only apply the terms of the contract.

6. References

Bali, M. (2009). Drools JBoss Rules 5.0 Developer's Guide, Packt Publishing. pp. 250.

Bravenboer, M. and Visser, E. (2004). Concrete syntax for objects: domain-specific language embedding and assimilation without restrictions. In: Proceedings of the 19th

Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2004). ACM, New York, NY, p. 365-383.

Browne, P. (2009). JBoss Drools Business Rules. Packt Publishing. pp. 304.

Brusilovsky, P. and Peylo, C. (2003). Adaptive and intelligent Web-based educational systems. In: International Journal of Artificial Intelligence in Education, 13 (2-4). IOS Press. p. 156-172.

Dillenbourg, P., Baker, M.J., Blaye, A. and O'malley, C. (1996). "The evolution of research on collaborative learning", In: Reimann, P., Spada, H. (Eds.), Learning in Humans and Machines: Towards an Interdisciplinary Learning Science, Oxford; Pergamon, p. 189-211.

Eberspacher, H. and Joab, M (2006). Virtus: group support using role-based collaboration. In: 6th IEEE International Conference on Advanced Learning Technologies (ICALT 2006), Kerkrade. p. 1079-1081.

Eberspacher, H. and Joab, M (2008). An Intelligent Web-Based Learning System for Group Collaboration Using Contracts. In: 9th International Conference on Intelligent Tutoring Systems (ITS 2008), Montreal. Springer-Verlag, 2008. v. 5091. p. 734-736.

Ferraris, C. and Martel, C. (2000). Regulation in groupware: the example of a collaborative drawing tool for young children. In: CRIWG'00 - 6th International Workshop on Groupware. p. 119-127.

Ferraris, C., Brunier and P, Martel, C. (2002). Constructing collaborative pedagogical situations in classrooms: a scenario and role based approach. In: CSCL'02 - Computer Support for Collaborative Learning. Jan 7-11, Boulder.

Friedman-Hill, E. (2003). JESS in action. Greenwich, Manning. pp. 480.

Mezura-Godoy, C. and Talbot, S. (2001). Towards Social Regulation in Computer-Supported Collaborative Work. In: workshop CRIWG'01 - 7th International Workshop on Groupware, Darmstadt, p. 84-89.

Paschke, A. (2005). RBSLA - A declarative Rule-based Service Level Agreement Language based on RuleML. In International Conference on Intelligent Agents, Web Technology and Internet Commerce (IAWTIC 2005). Vienna, Austria. IEEE Press, p. 308-314.

Shen, H. and Sun, C. (2002). Flexible Notification for Collaborative Systems. In: ACM CSCW'02 - Conference on Computer Supported Cooperative Work, Nov 16-20, New Orleans, Louisiana. p. 77-86.

Wessner, M. and Pfister, H. (2001). Group Formation in Computer-Supported Collaborative Learning. In S. Ellis, T. Rodden & I. Zigurs (Eds.). the 2001 International ACM SIGGROUP Conference on Supporting Group Work, Sep 30- Oct 3, 2001, Boulder CO. New York. p. 24-31.

Zhu, H. (2004). From WYSIWIS to WYSINWIS: Role-Based Collaboration. In: IEEE International Conference on Systems, Man, and Cybernetics, p. 5441-5446, The Hague, Netherlands, IEEE Press.

Zhu, H. (2006). Role Mechanisms in Collaborative Systems. International Journal of Production Research, vol. 44, No. 1, Janvier, p. 181-193. Taylor & Francis, London.