Agent-Based Architecture for Interactive System Design: Current Approaches, Perspectives and Evaluation

Christophe Kolski[1], Peter Forbrig[2], Bertrand David[3], Patrick Girard[4], Chi Dung Tran[1], and Houcine Ezzedine[1]

[1] LAMIH – UMR8530, University of Valenciennes and Hainaut-Cambrésis, Le Mont-Houy, F-59313 Valenciennes Cedex 9, France
firstname.name@univ-valenciennes.fr
[2] University of Rostock, Computer Science Department, Albert-Einstein-Str 21, D-18051 Rostock, Germany
Peter.Forbrig@informatik.uni-rostock.de
[3] LIESP, Ecole Centrale de Lyon, 36 avenue Guy de Collongue, F-69134 Ecully Cedex, France
Bertrand.David@ec-lyon.fr
[4] ENSMA / LISI, Teleport 2, 1 Avenue Clément Ader, B.P. 40109, F-86961 Futuroscope Chasseneuil Cedex, France
girard@ensma.fr

Abstract. This paper proposes a survey concerning agent-based architectures of interactive systems. This survey is focused on certain models and perspectives. Indeed, general agent-based architectures are first presented. Then agent-based approaches dedicated to CSCW systems are reviewed. The appearance of web services requires new agent-based approaches; basic ideas are introduced. Agent-based interactive systems necessitate new tools for their evaluation; an example of representative evaluation tool is presented.

Keywords: Human-computer interaction, Architecture model, agent-based systems, CSCW, design, evaluation.

1 Introduction

Since 1983 and the Seeheim's workshop, architecture is an important research topic in the Human-Computer Interaction domain. It started by defining recommendations for developers, and today, it allows tool definition that will help designing, developing and validating interactive systems. Different types of interactive system architectures have been proposed in the literature. The paper proposes a survey about agent-based architectures. A first global overview of models available in the literature is showed in Fig. 1.

The paper is composed of four main parts. In the first one, basic principles of architecture models will be introduced; general agent-based approaches will be listed. The second part concerns agent-based approaches dedicated to Computer Supported Cooperative Work (CSCW). The third part will link agent-based architecture and web services domains. Finally, the fourth part concerns evaluation of interactive systems based on an agent-based architecture; the first version of a dedicated evaluation tool will be briefly exposed.

J.A. Jacko (Ed.): Human-Computer Interaction, Part I, HCII 2009, LNCS 5610, pp. 624–633, 2009.

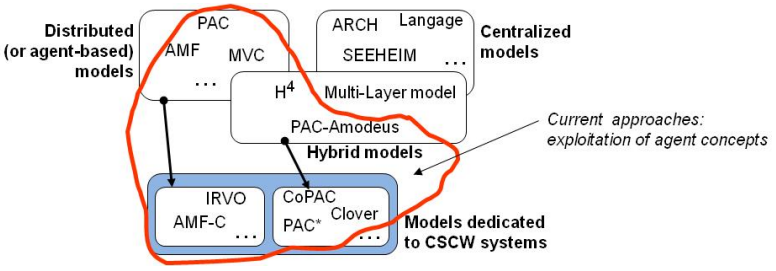

Fig. 1. Global overview of available architecture models

2 From Seeheim Model to Agent-Based Architectures

Two main approaches of architecture models were first elaborated: global models, and agent-based models. Global models define a precise structure based on a fixed number of components, whose role and nature are precisely defined. The well-known Seeheim model is the first of them [1]. It recommends developing user interfaces as a separate module, connected to a functional core on which it must lean. The interface itself is organized in three parts: the *Presentation* (devoted to the management of inputs and outputs), the *Controller* (defined as a component that manages the sequence of interaction elements) and the Application Interface (which allows the translation between the interactive "world" and the functional core). The main interest of the Seeheim model is to give original definitions that establish good foundations for all works on architecture and tools in HCI. For example, the Arch model [2] proposes some modifications of the Seeheim model (including the functional core into the model, defining an additional component, defining the notion of a "slinky model"), but keeps the main definitions, namely for dialogue control.

Nevertheless, global models bring forward some drawbacks, mainly when trying to apply object-oriented approach. While current object-oriented interactive application may involve hundreds of cases, the global structure gives no help on defining elementary interaction classes.

MVC (Model-View-Controller) [3], and then agent-based architecture models, such as PAC (Presentation-Abstraction-Control [4], AMF (multi-Agent-Multi-Facets [5]) and AoMVC (Agent-oriented MVC [6]), were designed to solve this problem. They define elementary software bricks composed of some parts (fixed number or not), and define the relations that must exist between bricks and parts. Some of them have been defined as design patterns. So doing, global functions such as Dialogue Control or Presentation are split in each elementary agent, what helps to support iterative design. Some tools to help define applications with these models have been designed, see for instance [7]. However, as global models, agent-based architecture models suffer from problems. Choosing the right level of decomposition is hard for non-experienced developers. More, ensuring strictly the rules of the model (for example, a PAC object only knows its father and its sons) may be difficult when implementation considerations are to be taken into account.

Hybrid models, which are supposed to benefit the most from the two approaches, emerged. Mainly, these models lean on a global definition of the architecture based on

the Arch model, and use an object-oriented approach to refine some of the main components, such as the *Presentation* or the *Controller*. For example, PAC-Amodeus [8] facilitates the design of multimodal applications. Another example is H^4, a model that was defined firstly for the Computer Aided Design area; tools were created for various applications, to help the design of applications [9, 10], to help their validation [11], or both [12]. Other related research proposes architecture models concerning distributed and plastic UI [13, 14].

3 Agent-Based Architectures: Approaches Dedicated to CSCW Systems

CSCW systems are not only interactive systems, but also and mainly multi-user distributed systems. For these reasons their architecture must answer new requirements. Three important characteristics are: (1) *taxonomy of collaborations*, which can be either related to the crossing in a matrix location (local or distant), and temporal view (synchronous or asynchronous), as suggested by [15], or related to the nature of cooperation (asynchronous cooperation, in session cooperation, in meeting cooperation and close cooperation [16]); (2) *awareness* is the information about activities done by other actors, needed in synchronous cooperation, which can be actor oriented (their effective participation) or production oriented expressed by WYSIWIS (What You See Is What I See) acronym with a strict or relaxed view of working data; (3) *nature of cooperation* activities can be examined, as initially proposed by [17] in relation to the support of three main kinds of activities, i.e. production, conversation/ communication and coordination between participants.

From an architectural point of view, CSCW systems are clearly inspired by interactive systems architectures, i.e. layered, agent and hybrid architecture are also used for CSCW systems. We can mention Zipper [18] and Dewan [19] models for layered collaborative systems, based mainly on ARCH model adaptation to multi-user distributed situations. ALV and AMF-C [20] are the representatives of agent-based systems. They generalize PAC agent model for collaborative distributed situations. CoPAC, PAC* and Clover (all described in [21]) are typical examples for hybrid systems. In this last case, they reuse ARCH model and adapt it to multi-user and distributed situations. All these architecture models take into account synchronous collaboration allowing real time interaction between cooperating actors. Distant and local interactions are treated in the same way, as only mediated interactions are taken into account, i.e. direct local non-mediated interaction is not supported. Asynchronous collaboration is not addressed mainly because in this case multi-user interaction, awareness and cooperative operations are not done by interaction. Awareness of shared artifacts (data) and participating actors is more or less supported as well as strict and relaxed WYSIWIS. Concerning cooperation activities (production, conversation and coordination), these are either fundamental elements (for PAC* and Clover) or naturally integrated (AMF-C). Hybrid architectures are either agent-based only in Control part of the model (CoPAC and PAC*) or agent orientation can be used also in other parts of the model.

Recent evolution of cooperative systems is related to the mobility of the actors, evolving in augmented real environment with pervasive behavior of the environment and related context-aware computing. The concept of nomadism (networking, handheld

devices, mobile communicating objects technology, localization and permanent or non permanent connectivity) extends the CSCW and allows us to introduce the concept of "capillary" CSCW [16]. We use this term by analogy with the network of blood vessels. As its name implies, the purpose of the capillary CSCW is *"to extend the capacities provided by co-operative working tools in increasingly finer ramifications, from their use on fixed proprietary workstations to small handheld devices"*. Main characteristics are: management of collaboration and coordination of the mobile actors, coherence and validity of the information exchanged between handheld devices which are connected only intermittently to the network and the "group" with the aim of having the most synchronized possible information, heterogeneity of the communication protocols of the handheld devices and constraints of interface and overall capacity of the handheld devices in terms of size of screen, speed transmission, memory, autonomy, as well as the interaction devices. In recent evolution of the AMF-C model, its transformation from a fully agent-based system to hybrid system, integration of IRVO perception of new paradigm of interaction (interaction with real and virtual objects) allows it to fully address problems with capillary cooperative systems.

In this new mobility context adaptation to different interaction devices, environmental situations, software and hardware platforms and user preferences becomes the core problem. Adaptation techniques can be classified in four different categories ranging from easiest to implement to most powerful: Translation techniques; Markup language-based approaches; Reverse and re-engineering techniques; Model-based approaches. Designing and implementing interactive collaborative applications that are adaptable (manually) or adaptive (automatically) to the context of use requires consideration of the characteristics of the user, the interactive platform as well as the constraints and capabilities of each environment. A state of the art survey shows us that among the large majority of existing approaches for adaptation, the model-based approach seems to be the most powerful. Such approach uses high level and abstract representations that can be instantiated latter on in the development lifecycle to meet specific usability requirements. However, these approaches need to combine apparently independent models such as concepts (e.g. UML), task (e.g. CTT), platform (e.g. CC/PP) or user profiles. The relationships between these models need to be defined at the design step and refined at run-time in order to be able to achieve the overall usability. Our belief is that, what we refer to as an interaction model is the right place to glue together all the models and usability attributes. This model must support both design stage linking other models and run-time. In addition, because Software Engineering and HCI have shown the importance of clearly separate functional core from presentation components, our interaction model is supported by a well structured architecture.

In this new version of the AMF-C architectural model [22], we maintain the basic characteristic of the model, i.e. the Multi-faceted approach allows the creation of new facets, to clarify the behavior and allow automation of implementation process; a graphical formalism that expresses the control structure of multi-user interactions and adaptation in real time of awareness characteristics; and a run-time model that allows dynamic control of interactions. We add IRVO interaction formalism allowing the expression of new augmented reality interactions and we structure the system with hybrid approach, allowing to mix XML specifications, engine based interpretation and connection to real components of functional core or managing new interaction devices (Fig. 2).

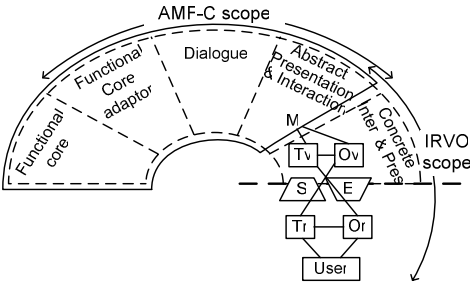

Fig. 2. Relations between Arch model (dashed lines), AMF-C and IRVO models

4 Web Services and Agent-Based Architectures

Web services lead to new possibilities and problems concerning distributed system design. Fig. 3 suggests a complex industrial organization exploiting web services.

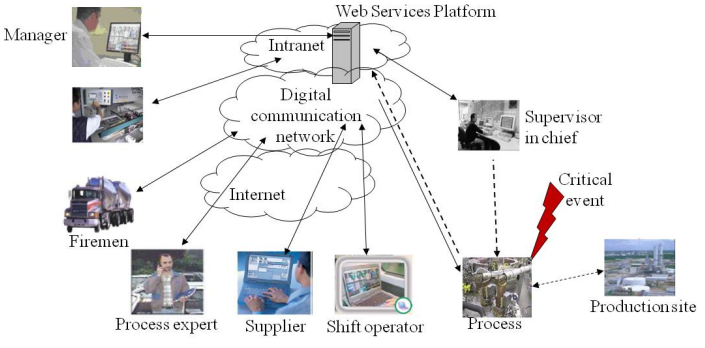

Fig. 3. Example of different actors communicating directly or not via web services [23]

The traditional web services provide functionalities based on classical client/server architecture, but agent-based architectures offer new perspectives in this field. They utilize autonomous and proactive behaviors of agents. Interesting new approaches appear in the literature. For instance, a technical framework for AWS (agent-based web services) is described in [24]; it supports the idea of capturing, modeling and implementing service functionalities with autonomous and dynamic interactions. Technically agent-oriented software construction, knowledge representation and interaction mechanisms are integrated. Fig. 4 gives an impression of the framework. DAML-S (DARPA agent markup language for services) is a semantic markup language for describing web services and related ontologies. It has been superseded by OWL-S [25].

A discussion of dynamic web-service invocation by agents can be found in [26]. Their infrastructure is a hybrid peer-to-peer model. Agents are used to specify service providers and service customers. For this purpose JADE [27] (Java Agent Development Environment) is used; it is a framework developed as open source project. A

web service can be published as a JADE agent service and agents services can be published as web service endpoints (see also [24]).

Such propositions have to be considered with attention regarding agent-based architecture perspectives concerning service-oriented interactive systems.

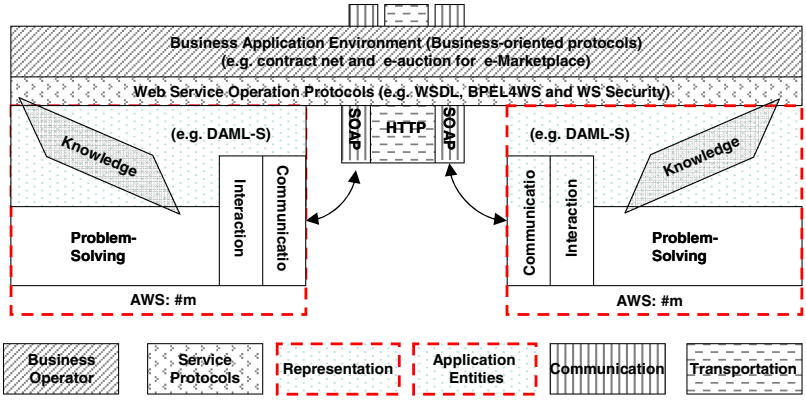

Fig. 4. Integrated technical framework for agent-based web services [24]

5 Agent-Based Architectures: The Evaluation Problem

The evaluation of interactive systems aims at ensuring that users are capable of realizing their tasks. The evaluation methods and tools are numerous and of different types; they are generally based on two global criteria: *utility* and/or *usability* [28]. When the interactive system uses an agent-based architecture, new methodological and conceptual questions appear. For instance: how to evaluate such systems? Is it necessary to combine several evaluation methods? Is it possible to be assisted by automated or semi-automated evaluation tools? How to connect such tools to the agent-based systems? How to link the agents' behaviors with the analyzed situations? There are several further questions.

We are particularly interested in automated or semi-automated tools.

An electronic informer (EI) is a software tool that captures automatically interactions between the user and the UI in real situations in a discreet and transparent way, so that the user does not feel hampered by the tool. The captured data are objective and can be scientifically analyzed by the evaluators. For a review about EI, we refer to [29]. Several tools are available, but very few of them take into account the specificities of agent-based interactive systems in their evaluation approaches [30, 31, 32, 33]. The architecture of a tool dedicated to such systems is showed in Fig. 5. This kind of EI aims at capturing not only interactions between user and interface agents in terms of occurred UI events like other EIs, but also interactions between agents themselves in terms of interactions between services. It aims also to go further than other EIs to assist evaluators in interpreting analysis results of captured data in order to evaluate three aspects of an agent-based interactive system: user interface (UI), some non-functional properties (such as response time, reliability, complexity, etc.), and

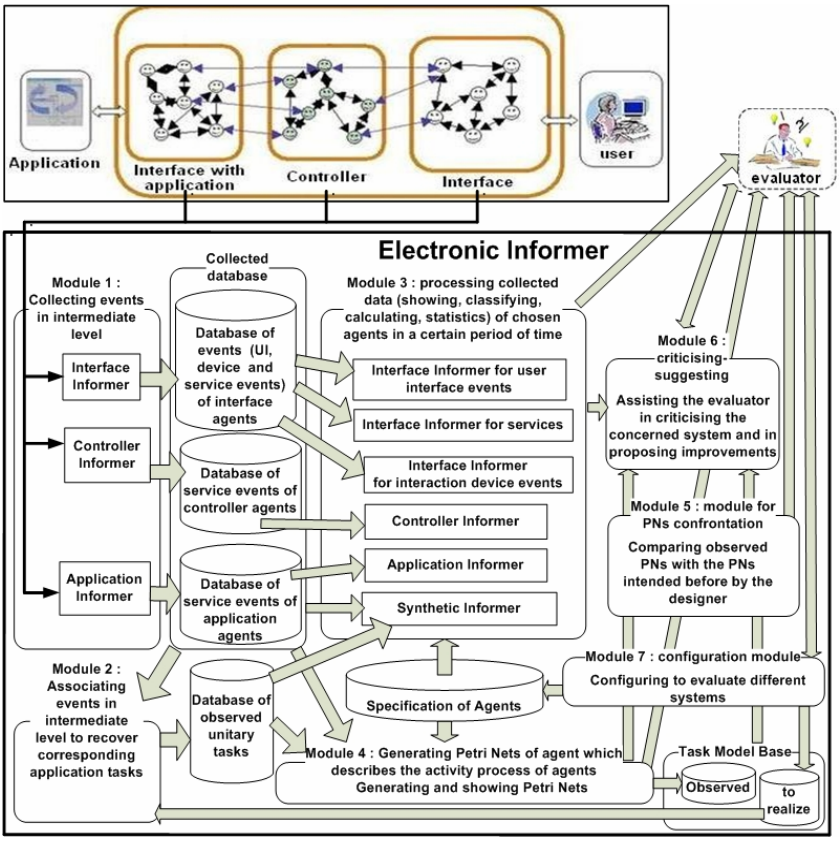

Fig. 5. Example of a tool for evaluating agent-based interactive systems [33]

properties of users to operate systems (ability, habits, preferences, progress of a certain user, etc.).

Seven independent modules compose this tool. The module 1 is responsible of capturing events that occur from all agents of the system and then, it saves them into a database that will be analyzed by other modules. The connection between this EI and the evaluated agent-based system is based on the association of each type of agents (interface agents, controller agents, application agents) with a corresponding informer. The evaluation can be remotely realized. This module 1 and the evaluated system can run on the same machine, or on two different ones on the network.

After capturing data, this EI enables the evaluator to determine tasks that user has realized (module 2). Some synthetic calculations and statistics can be realized on captured data such as the number and frequency of occurred events, average response time of service interactions, time taken to realize a task, number of successful or failed tasks, etc., of any chosen agent or all the agents in any chosen period of time. These analysis results will be showed to the evaluator using tables or graphs (module 3). The tool also enables the generation of Petri Nets (PNs) and the evaluator can

compare PNs (module 4 and 5). A generated PN describes user's actions in terms of UI events (that have ever occurred on interface agents) and system's actions in terms of executed services of agents in order to realize a certain task. Generated PNs are called *observed PNs* or *real PNs*. The evaluator can compare *real PNs* to realize a certain task of a certain user with *theoretical PNs* predicted by the designers for the same task or he/she can do the comparison between *real PNs* to realize the same task of different users. Exploiting formal aspect of the PNs, such comparisons are very useful for evaluators to detect problems of the interface, the system or the users such as: bad or useless actions of users, non-optimal way chosen by users to realize tasks, failed service interactions, properties of users (habits, evaluation and comparison of abilities of different users, supervision of the progress of abilities of a certain user, etc.). The analysis results of the module 3, the generation and comparison of PNs, all these results can be interpreted with the indications of module 6 (that enables the association with an open list of determined criteria) to help evaluators critique the system and propose useful suggestions to the designers for improvements.

This tool is representative of a new generation of tools dedicated to agent-based interactive systems. A lot of research is still necessary in this domain (adaptation to different application fields and architecture models, helps in real time…).

6 Conclusion and Perspectives

Since the eighties, many models and approaches are proposed in the literature concerning so-called *distributed* or *agent-based* architectures of interactive systems. By lack of place, it was just possible to propose a brief overview of this domain, about (1) general agent-based architecture models, (2) models dedicated to CSCW systems, (3) interactive systems based on web services, (4) evaluation of interactive systems using agent-based architecture. Many research and development perspectives can be now envisaged. Currently, general agent-based architecture models are mainly used at the conceptual level. They allow good design of application, minimizing dependencies and improving maintainability of applications. They need now to be more largely used at the implementation level. Their inclusion into integrated development environments, such as Eclipse for example, might be the next step to allow tools to be developed. Help for software design, simulation, and evaluation are the main topics that are to be addressed.

Capillary cooperative systems need important context adaptation. These mechanisms are more easily elaborated in hybrid architectures using agents in several layers. The benefits of autonomous behavior and independence of agent-based systems constitutes an important advantage. Many researches concern currently context-aware interactive systems; different types or generations of adapted agent-based architecture models have to be progressively proposed. Agent-based systems might help to dynamically compose web services. In this way they can support dynamic adaptation of workflow systems.

Many research problems have also to be studied and solved regarding the evaluation of agent-based interactive systems.

Acknowledgements. The present research work has been supported by CISIT, the Nord-Pas-de-Calais Region, the European Community (FEDER). The authors gratefully acknowledge the support of these institutions.

References

1. Pfaff, G.E.: User interface management system. Springer, Heidelberg (1985)
2. Bass, L., Little, R., Pellegrino, R., Reed, S.: The Arch Model: Seeheim revisited. In: Proceedings of User Interface Developers Workshop, Seeheim (1991)
3. Goldberg, A.: Smalltalk-80, the interactive programming environment. Addison-Wesley, Reading (1983)
4. Coutaz, J.: PAC, an Object-Oriented Model for Dialog Design. In: Bullinger, H.-J., Shackel, B. (eds.) Proc. Interact 1987, 2nd IFIP International Conference on Human-Computer Interaction, Stuttgart, Germany, September 1-4, 1987, pp. 431–436 (1987)
5. Ouadou, K.: AMF: Un modèle d'architecture multi-agents multi-facettes pour Interfaces Homme-Machine et les outils associés (in French), PhD Thesis, ECL, Lyon (1994)
6. Goschnick, S., Sterling, L.: Shadowboard: an Agent-oriented Model-View-Controller (AoMVC) architecture for a digital self. In: Proc. Int. Workshop on Agent Technologies over Internet Applications (ATIA 2001), Tamkang University, Taipei, Taiwan (2001)
7. Jambon, F.: From Formal Specifications to Secure Implementations. In: Kolski, C., Vanderdonckt, J. (eds.) Computer-Aided Design of User Interfaces (CADUI 2002), pp. 43–54. Kluwer Academics, Dordrecht (2002)
8. Nigay, L.: Conception et modélisation logicielles des systèmes interactifs: application aux interfaces multimodales (in French), PhD Thesis, Joseph Fourier Univ., Grenoble (1994)
9. Texier, G., Guittet, L., Girard, P.: The Dialog Toolset: a new way to create the dialog component. In: Stephanidis, C. (ed.) Universal Access in HCI, pp. 200–204. Lawrence Erlbaum Associates, Mahwah (2001)
10. Depaulis, F., Maiano, S., Texier, G.: DTS-Edit: an Interactive Development Environment for Structured Dialog Applications. In: Kolski, C., Vanderdonckt, J. (eds.) Computer-Aided Design of User Interfaces (CADUI 2002), pp. 75–82. Kluwer Academics, Dordrecht (2002)
11. Francis, J., Girard, P., Boisdron, Y.: Dialogue Validation from Task Analysis. In: Duke, D.J., Puerta, A. (eds.) Eurographics Workshop on Design, Specification, and Verification of Interactive Systems (DSV-IS 1999), Braga, Portugal, pp. 205–224. Springer, Heidelberg (1999)
12. Baron, M., Girard, P.: SUIDT: Safe User Interface Design Tool. In: International Conference on Intelligent User Interfaces Computer-Aided Design of User Interfaces (IUI-CADUI 2004), Madeira, Portugal, pp. 350–351. ACM Press, New York (2004)
13. Balme, L., Demeure, A., Barralon, N., Coutaz, J., Calvary, G.: CAMELEON-RT: A Software Architecture Reference Model for Distributed, Migratable, and Plastic User Interfaces. In: Markopoulos, P., Eggen, B., Aarts, E., Crowley, J.L. (eds.) EUSAI 2004. LNCS, vol. 3295, pp. 291–302. Springer, Heidelberg (2004)
14. Calvary, G., Daassi, O., Coutaz, J., Demeure, A.: Des widgets aux comets pour la plasticité des systèmes interactifs. Revue d'interaction Homme-Machine 6(1), 33–53 (2005)
15. Ellis, C.A., Gibbs, S.J., Rein, G.L.: Groupware: some issues and experiences. Communication of ACM 34(1), 39–58 (1991)

16. David, B., Chalon, R., Vaisman, G., Delotte, O.: Capillary CSCW. In: Stephanidis, C., Jacko, J. (eds.) Human-Computer Interaction Theory and Practice, LEA, pp. 879–883 (2003)
17. Ellis, C.A., Wainer, J.: A Conceptual Model of Groupware. In: Proceedings of CSCW 1994 Conference, pp. 79–88. ACM Press, New York (1994)
18. Patterson, J.F.: A taxonomy of architectures for synchronous groupware applications. In: Workshop on Software architectures for cooperative systems CSCW 1994. ACM SIGOIS Bulletin Special Issue Papers of the CSCW 1994 workshops, vol. 15(3) (April 1995)
19. Dewan, P., Choudhary, R.: Coupling the User Interfaces of a Multiuser Program. ACM Transactions on Computer-Human Interaction 2(1), 1–39 (1995)
20. Tarpin-Bernard, F.: Architectures logicielles pour le travail cooperatif (in French), PhD Thesis, Ecole Centrale de Lyon, France (1997)
21. Laurillau, Y.: Conception et réalisation logicielles pour les collecticiels centrées sur l'activité de groupe: le modèle et la plate-forme Clover (in French), PhD Thesis, Joseph Fourier University, Grenoble (2002)
22. Tarpin-Bernard, F., Samaan, K., David, B.: Achieving usability of adaptable software: the AMF-based approach. In: Seffah, A., Vanderdonckt, J., Desmarais, M.C. (eds.) Human-Centered Software Engineering, Software Engineering Models, Patterns and Architectures for Human-Computer Interaction, Springer, Heidelberg (2009)
23. Idoughi, D.: Contribution à un cadre de spécification et conception d'IHM de supervision à base de services web dans les systèmes industriels complexes, application à une raffinerie de sucre (in French), Ph.D. Thesis, University of Valenciennes, France (2008)
24. Li, Y., Shen, W.-m., Ghenniwa, H., Lu, X.: Model-Driven Agent-Based Web Services IDE. In: Wang, S., Tanaka, K., Zhou, S., Ling, T.-W., Guan, J., Yang, D.-q., Grandi, F., Mangina, E.E., Song, I.-Y., Mayr, H.C. (eds.) ER Workshops 2004. LNCS, vol. 3289, pp. 518–528. Springer, Heidelberg (2004)
25. Paolucci, M., Sycara, K.: Autonomous Semantic Web Services. IEEE Internet Computing 7, 34–41 (2003)
26. Yang, H., Chen, J., Meng, X., Zhang, Y.: A Dynamic Agent-based Web Service Invocation Infrastructure. In: Proceedings of the First Int. Conf. on Advances in Computer-Human Interaction, Sainte Luce, Martinique, pp. 206–211 (2008)
27. Bellifemine, F., Caire, G., Poggi, A., Rimassa, G.: JADE: A software framework for developing multi-agent applications. Lessons learned, Information & Software Technology 50(1-2), 10–21 (2008)
28. Nielsen, J.: Usability Engineering. Academic Press, Boston, MA (1993)
29. Hilbert, D.M., Redmiles, D.F.: Extracting usability information from user interface events. ACM Computing Surveys 32(4), 384–421 (2000)
30. Trabelsi, A., Ezzedine, H., Kolski, C.: Architecture modelling and evaluation of agent-based interactive systems. In: Proc. IEEE SMC 2004, The Hague, pp. 5159–5164 (2004)
31. Tarby, J.-C., Ezzedine, H., Rouillard, J., Tran, C.D., Laporte, P., Kolski, C.: Traces using aspect oriented programming and interactive agent-based architecture for early usability evaluation: Basic principles and comparison. In: Jacko, J.A. (ed.) HCI 2007. LNCS, vol. 4550, pp. 632–641. Springer, Heidelberg (2007)
32. Ezzedine, H., Bonte, T., Kolski, C., Tahon, C.: Integration of traffic management and traveller information systems: basic principles and case study in intermodal transport system management. Int. J. of Comp., Com. & Control (IJCCC) 3, 281–294 (2008)
33. Tran, C.-D., Ezzedine, H., Kolski, C.: A generic and configurable electronic informer to assist the evaluation of agent-based interactive systems. In: 7th international conference on Computer-Aided Design of User Interfaces, CADUI 2008, Albacete (June 2008)