A Generic and Configurable Electronic Informer to Assist the Evaluation of Agent-Based Interactive Systems

Chi Dung Tran, Houcine Ezzedine, Christophe Kolski

In V. López-Jaquero, F. Montero Simarro, J.P. Molina Massó, J. Vanderdonckt (Eds.), Computer-Aided Design of User Interfaces VI, Proceedings of the Seventh International Conference on Computer-Aided Design of User Interfaces, CADUI 2008, Albacete, Spain, June 11-13, 2008. Springer 2009 ISBN 978-1-84882-205-4

A generic and configurable electronic informer to assist the evaluation of agent-based interactive systems

Chi Dung TRAN, Houcine EZZEDINE, Christophe KOLSKI

LAMIH–UMR CNRS 8530, University of Valenciennes and Hainaut-Cambrésis, Le mont Houy, F-59313 Valenciennes Cedex 9, France
{ChiDung.Tran,Houcine.Ezzedine,Christophe.Kolski}@univ-valenciennes.fr

Abstract The evaluation of user interactive systems has been an active subject of research for many years. Many methods have been proposed but most existing evaluation methods do not take the specific architecture of an agent-based interactive system into account and nor do they focus on the coupling between the architecture and evaluation phase. In this article, we propose an agent-based architecture of interactive systems that is considered as being mixed (it is both functional and structural). Based on this architecture, we propose a generic and configurable model of an evaluation tool, called "electronic informer", designed to assist evaluators in analyzing and evaluating interactive systems with such architecture.

1 Introduction

Nowadays, in spite of the existence of several methodologies for the development of interactive systems, designing, developing and assessing, in terms of utility and usability (Bastien and Scapin 1995; Nielsen 1993; Shneiderman 1998) an agent-based interactive system is still a difficult task. It is therefore necessary to provide methods, models and evaluation tools to make it easier.
The section 2 of this paper presents a brief state of the art concerning the architectures for traditional interactive systems as well as for agent-based interactive systems. The section 3 proposes an architecture that is both functional and structural. By using this architecture as a basis, in the section 4, we propose a generic and configurable model of an evaluation tool called "electronic informer"; its aims at assisting the evaluation of interactive applications of this architecture. The last section is used for our experiment and the conclusion.

2 Interactive system architectures

Architecture of an interactive system supplies the designer with a generic structure from which he/she can build an interactive application. It is a set of structures that include: components, the outside visible properties of these components and the relations between them (Coutaz and Nigay 2001). Researchers have proposed several architecture models over the past twenty years. These architecture models recommend the same principles, based on the separation between the functional core of system (application) and the human-machine interface. This separation makes modifications easier; it allows modifying the interfaces without affecting the application. Two main types of architecture can be singled out: (1) functional models, such as Seeheim (Pfaff 1985), Arch (Bass et al. 1991) and (2) structural models, such as PAC proposed by J. Coutaz (Coutaz 1987), and its variations, AMF (Tarpin-Bernard 1999) or MVC (Goldberg 1983) and its variations.

The functional model split an interactive system into several functional components. For example, the Seeheim model is made up of 3 logical components (Presentation, Dialogue Controller, Application Interface); the Arch model defines a functional breakdown of an interactive system into 5 components in which: both the presentation and interaction components are a decomposition of the presentation of the Seeheim model, the functional kernel component, the domain adapter component and the dialogue controller component. The functional models enable to separate system analysis-design difficulties by decomposing into different modules. Nevertheless, the functional models show some disadvantages. First, they do not define the internal structure of modules and the communication between them. Second, Arch and Seeheim provide canonical functional structures with big grain, the functionalities are mixed in the too macroscopic components (Tarpin-Bernard 1999); they are useful as a structural framework for a design or a rough analysis of the functional decomposition of an interactive system (Trabelsi et al. 2004). These decompositions are generally not enough to complex applications. These inconveniences make functional systems inadaptable to complex applications in general and to industrial supervision systems in particular.

The structural models aim at a finer breakdown by using structural components, and in particular those said to be distributed or agent approaches, suggest grouping the functions together into one entity, the agent. The agents of this type of architecture are then organized in a hierarchical manner according to principles of composition or communication. For example, a MVC agent is made up of three facets: Model, View and the Controller. The PAC model defines an agent using three facets: the Presentation, the Abstraction and the Control. The decomposition into many autonomous and cooperative entities enables to accelerate the feedback of system to the user. This advantage is very useful to the supervision applications that can be highly interactive; in consequence, the intensive dialogue between the user and the application can slow the system down. In spite of this advantage, the structural models have also following disadvantage: the number and the role of

agents are not made clear. Moreover, the designer of a supervision application can have difficulty in following the global interface because the interface is distributed in many facets "Presentation" of agents. The functional models can solve this problem because they provide only one "Presentation" component to represent the interface.

Our approach is intended to be mixed as its principles borrow from both types of model; it is both functional and structural.

3 Agent-based architecture proposed as a mixed model

The mixed model is a combination of these two above mentioned models in order to exploit advantages of each of them. The agent-based architecture proposed (Grislin-Le Strugeon et al. 2001) has to be considered as a such mixed model. We suggest using a division into three functional components recommended in the Seeheim model which we have called respectively: *interface with the application* (connected to the application), *dialogue controller*, and *interface* or *presentation* (this component is directly linked to the user). Each of these components can be broken down further in a structural approach in the form of agents. These components are built like three multi-agent systems and they are considered as working in parallel, at least, at a theoretical point of view.

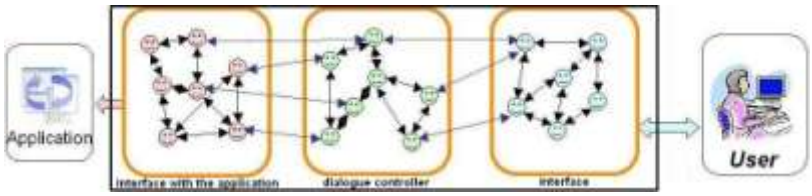

Fig. 1. Our agent-based architecture

The application agents, manipulating the field concepts of the application, cannot be directly accessed by the user. One of their roles is to ensure the real time transmission of the information necessary for the other agents to perform their task. The interface agents are in direct contact with the user (they can be seen by the user) through the associated user interface events. These agents co-ordinate between them in order to intercept the user commands and to form a presentation that allows the user to gain an overall understanding of the current state of the application. The control agents in the Dialogue Controller component provide services for both the application and the interface agents in order to guarantee coherency in the exchanges emanating from the application towards the user, and vice versa. Their role, in particular, is to link the two other components together by distributing the user commands to the application agents concerned, and by distributing the application feedback towards the interface agents concerned. All these agents communicate amongst themselves in order to answer the user actions. Each

agent of this architecture associates with a set of services that are the actions that this agent can execute. The communication between agents is realized by the invocation between services of agents. This architecture as well as its events (user interface events, services) has been formally described before proposing a generic and configurable “electronic informer”, designed to assist evaluators in analyzing and evaluating interactive systems of this architecture.

4 Proposition of an electronic informer adapted to agent-based interactive systems

An "electronic informer" is a software tool that ensures the automatic collection, in a real situation, of users’ actions and their repercussions on the system. The collection of information is done in a discreet and transparent way for the user, who must not at any time feel hampered by the presence of the informer. This is an advantage of such a tool. Objective data collected through interactions can be treated, analyzed and shown in a synthetic shape to the evaluator. This facilitates the analysis of the results. At this moment, many evaluation tools have been proposed but they have some disadvantages and limitations: (a) The current evaluation tools do not take into account the specific architecture of an agent-based interactive system (Trabelsi et al. 2004) and there are rarely propositions concerning the coupling between the architecture and evaluation phase. (b) The current electronic informers often contain some stages like collecting interaction data, and then retrieving these data to realize some analysis such as counts, summary statistics, detecting patterns, etc., and finally visualizing analysis results in a synthetic shape to the evaluator. For example, the tool WET (Etgen and Cantor 1999) collects only interaction data between user and the interface without any later analysis, the tools RemUSINE (Paterno et Ballardin 2000), WebRemUSINE (Paganelli and Paternò 2002) realize some analysis such as calculations, summary statistics concerning tasks executed and Web pages visited by the user and then, then visualize analysis results to the evaluator in terms of diagrams, WebQuilt represent sequences of visited Web pages in terms of an interactive directed graph (Hong et al.2001). After tools show analysis results, in order to identify problems of the user interface and propose useful suggestions for improvement, the evaluator must interpret these analysis results by himself without any indications or assistances. As consequence, evaluation results depend on the ability and experience of the evaluator very much. (c) The current tools working with guidelines often read source code of the user interface to determine that whether their static presentations (color or size of the text fonts, position or size of the button...) violate a set of predefined guidelines. They do not take into account of interactive behaviors of the application in terms of interactions between user and interface or between components of the application for the evaluation. (d) Some current "feedback quality agents" are only softwares used to gather data about what were happening

in the application whenever it crashes. The "feedback quality agents" only gather technical information about the context, state of the application when it had problem such as OS Version, Processor Type, Display Type, register, functions were called just before the failure, etc. And then, these data are sent to the development team to help them identify problems, the cause of the crash more readily and then, improve the future version of the application. “Feedback quality agents" can permit the user to report what he/she was doing when the failure appears to the development team. These tools only take into account of the context and state of the application when it had problem, as consequences, assistances in evaluating of interactive applications are limited compared other tools like electronic informers or tools working with guidelines.

We propose a generic and configurable model of an “electronic informer” to remedy these limitations. The first version of an electronic informer has ever been studied and developed (Trabelsi et al. 2004). However, it is not a generic tool but only a specific one. This first specific tool aims at evaluating a specific agent-oriented applicative application that is intended to manage the passenger information on a public transport system in a project called SART (SART 2007). It cannot be used to evaluate other agent-oriented applications because it depends on the number of agents, the structure and the contents of such systems. Furthermore, it also shows some inconveniences and shortcomings. We solve such problems by proposing and developing a generic and configurable model of an “electronic informer” made up of 7 main modules (Figure 2). Our tool takes into account of the architectural specificities of interactive applications of the proposed architecture. In particular, our tool collects interactions between the user and the interface agents in terms of occurred user interface events and interactions between agents themselves in terms of executed services to evaluate interactive applications. After analyzing collected data, this tool provides the evaluator with an open list of determined criteria which can be ergonomic criteria or quality attributes. These criteria are associated to analysis results to give to the evaluator clear indications and assistances in interpreting analysis results in order to identify problems of the application based on these criteria and then, he/she can propose useful suggestions to the designer for necessary modifications to improve the application. These associations should be automated as much as possible in the future version of this tool.

Module 1 (M1): This module can run in background to collect events that appear (occurred user interface events and executed services) from all agents of the concerned interactive system and save them into a database which will be exploited by other modules. This module 1 and evaluated application can be in the same or other place. That means the evaluation can be done remotely. The module 1 and the other modules do not communicate directly each other. The data collection and its treatment are separated. As a result, the module 1 can be modified without affect the other modules and vice versa.

M2: this module enables the evaluator to associate events in intermediate level (user interface events and services) with each task. Several events in intermediate level can be realized to obtain a certain task. For example, the user interface

events *Image_Vehicule_Click, TabDriver_click*, *TextBoxMessage_OnChange* and *buttonOK_Click* can be associated with the task "Send a message to the driver/voyagers" of the system intended to supervise the passenger information on a public transport system in the SART project.

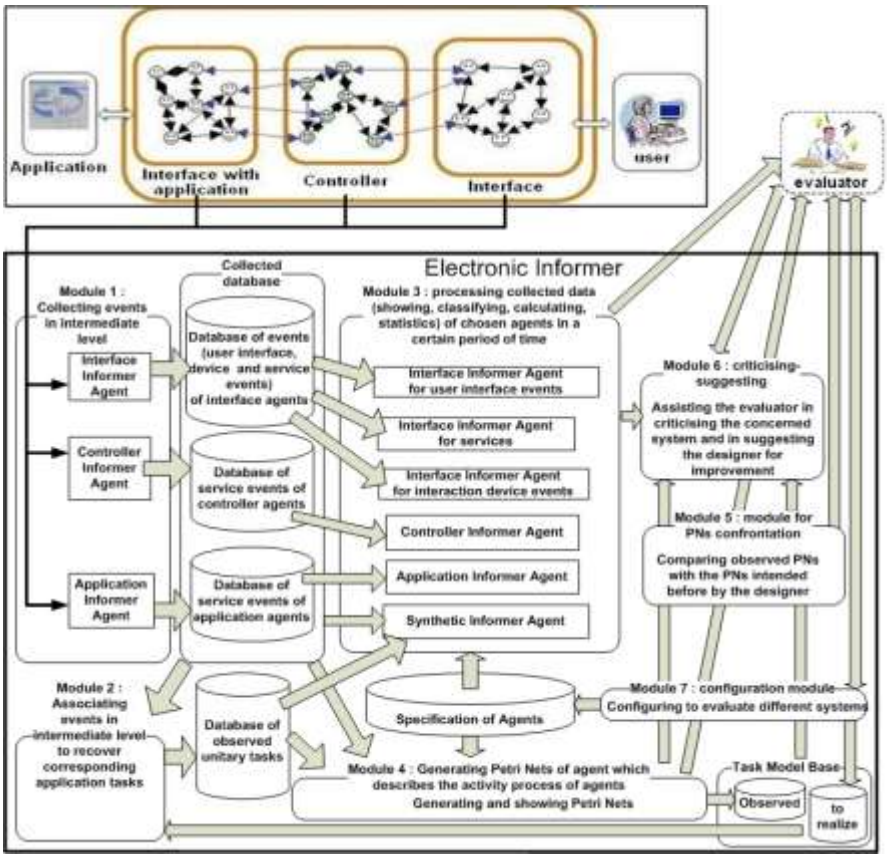

Fig. 2. Generic and configurable model of the electronic informer proposed

M3: processing collected data of a chosen agent (interface, control and application agents) or all the agents in a certain period of time and showing results in forms understandable for the evaluator. Here are examples of calculations and statistics: response time for interactions between services; time for a certain user interface event (time for typing a text box…); time for completing a service and a task; the percentage of services accomplished and furthermore, of tasks accomplished, the error's percentage, the percentage of services and furthermore, of tasks achieved per unit of time, the ratio of failure or success for interactions between services, the ration of failure or success for each or all the tasks, the ration of appearance of each user interface event of a certain interface agent, the percent-

age of execution for each service of a certain agent, and so on. The results are showed in table or graph form (figure 3, 4).

M4: generating Petri Nets (PNs) to describe activity process of agents and users in the system from collected data. Indeed, it describes process of interactions between services of different agents and process of activity of user (in terms of user interface events) to realize a certain task. We call them "observed" PNs. Generated PNs brings evaluators visual views of all real activities of the user and the concerned system and can be used for comparison purposes later. The left part of the figure 5 and the figure 6 illustrates generated PNs describing the activity process to realize the task "Send a message to stations" of the application that supervises the vehicles of urban common transport in the SART project. The generated PNs are described by PNML (Petri Net Markup Language) and are opened by the tool Renew version 2.1. In the figures 5 and 6, eviuM,N-I stands for user interface event M of the interface agent N, sM,N-I(A) stands for service M of the interface (application) agent N.

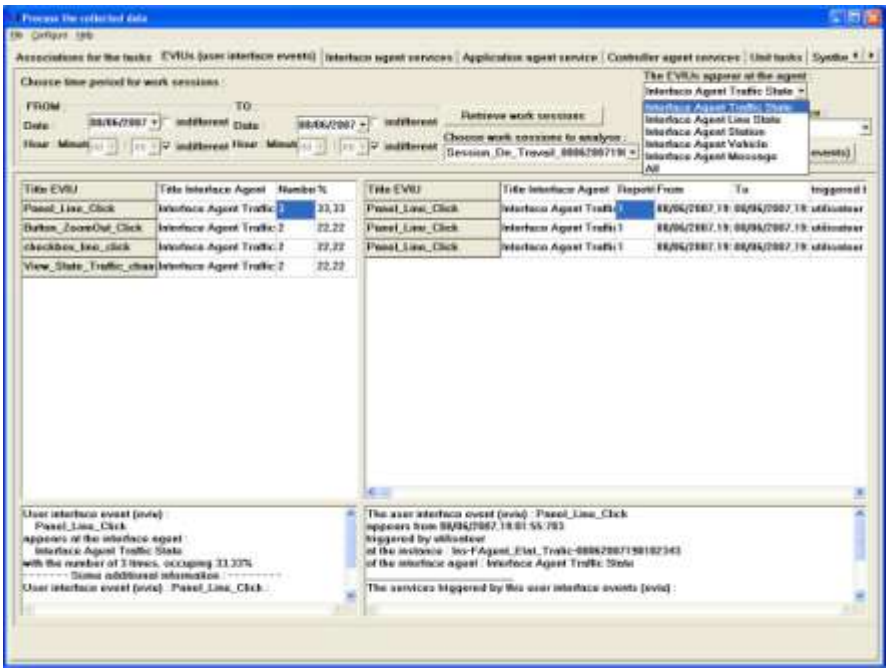

Fig. 3. One of screen pages of M3: UI events occurred in the interface agents in table form

M5 (figures 5 and 6): comparing PNs generated above with theoretical PNs that system designer has intended. This comparison assists the evaluators in detecting use errors; for ex., one can perceive that the user has done useless manipulations or chosen a non-optimal way to realize a task. These comparisons can also be used to assist the evaluator in learning habits of users or in evaluating and comparing the ability of different users to use a system.

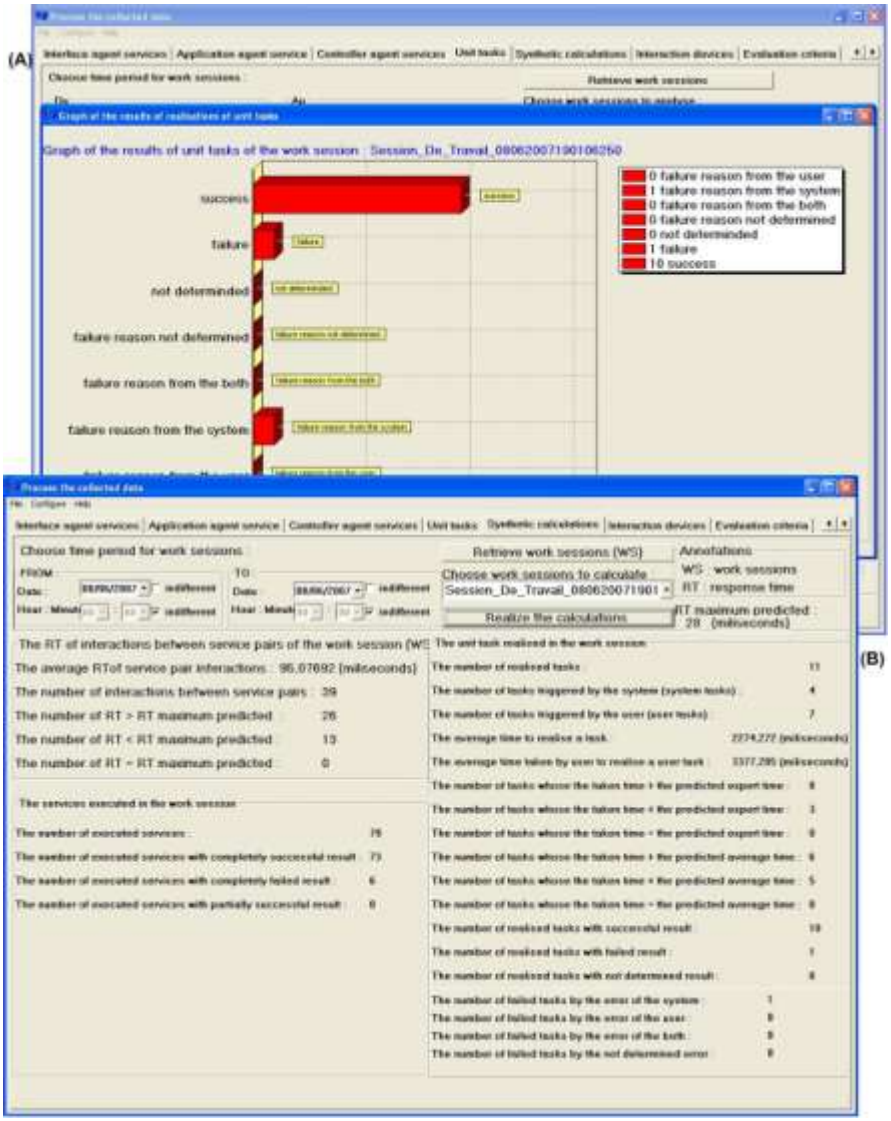

Fig. 4. Two screen pages of M3.
(A) The results of the realizations of tasks in graph form.
(B) Some calculations (the ratio of successful tasks, average response time, confrontation between real and predicted time to realize tasks, and so on).

For ex., the fig. 5 tells the evaluator about the way chosen by the user to realize the task among three ways predicted by the system designer. The fig. 6(A) let the evaluator know that the way chosen by the user 1 is optimal but the one chosen by the user 2 is not. The fig. 6(B) let the evaluator perceive two problems: (1) the user 3 realized unsuccessfully the task because of an error of the service *s2,3-I*. This

service must invoke the service *s1,2-A* but in fact, the service *s1,2-A* is not executed. One can also perceive this problem by comparing generated PNs of user 2 with theoretical PNs that system designer has intended. (2) the user 4 executed an useless manipulation through the UI event button_*Cancel_Click* (red arrow on the figure). As a result, the UI event *Window_Properties_Station_Hidden* was triggered and the user 4 had to execute the manipulation through the event *Image_Station_Click* again.

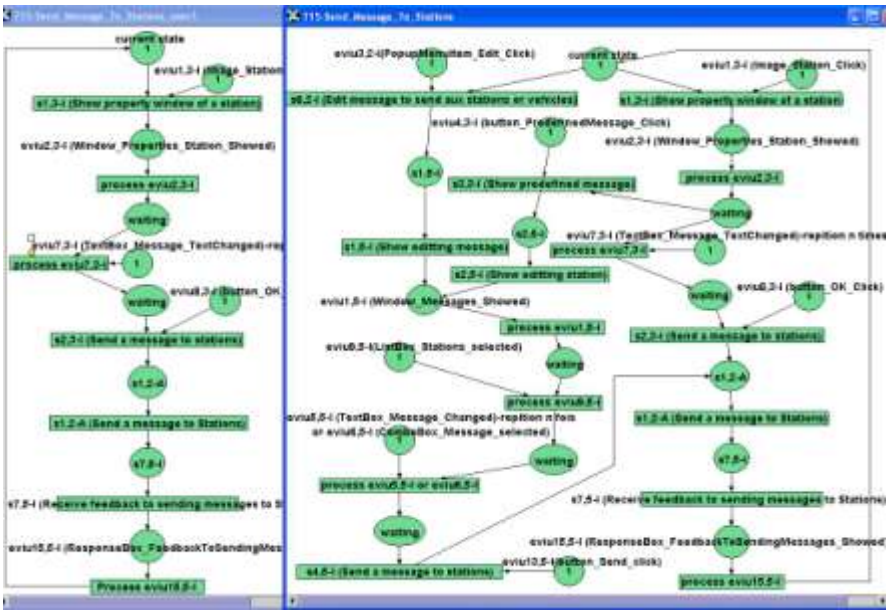

Fig. 5. Comparing generated PNs of user 1 (left side) with PNs intended by designer (right side) to realize the task "Send a message to stations"

M6: the evaluator must interpret analysis results to evaluate agent-based user interactive systems as mentioned above. M6 is responsible to assist the evaluator in doing that. As a result, the evaluator can criticize concerned system and give useful suggestions to the designer for necessary modifications to improve it. M6 is not an expert system but a tool to assist the evaluator in evaluating concerned system by providing him/her with an open list of criteria which can be ergonomic criteria or quality attributes. These criteria must respect some principles: (1) they aim at evaluating 3 different aspects of agent-based interactive systems: user interface, user ability to manipulate systems, some non-functional properties such as response time, reliability, so on. (2) These criteria must be evaluated completely or partially using the data collected and processed by the other modules. In other words, the evaluator has to be able to interpret the data collected and processed by the other modules to evaluate the systems based on these criteria. In the future, the electronic informer can be combined with other methods to evaluate more enough and exactly such criteria.

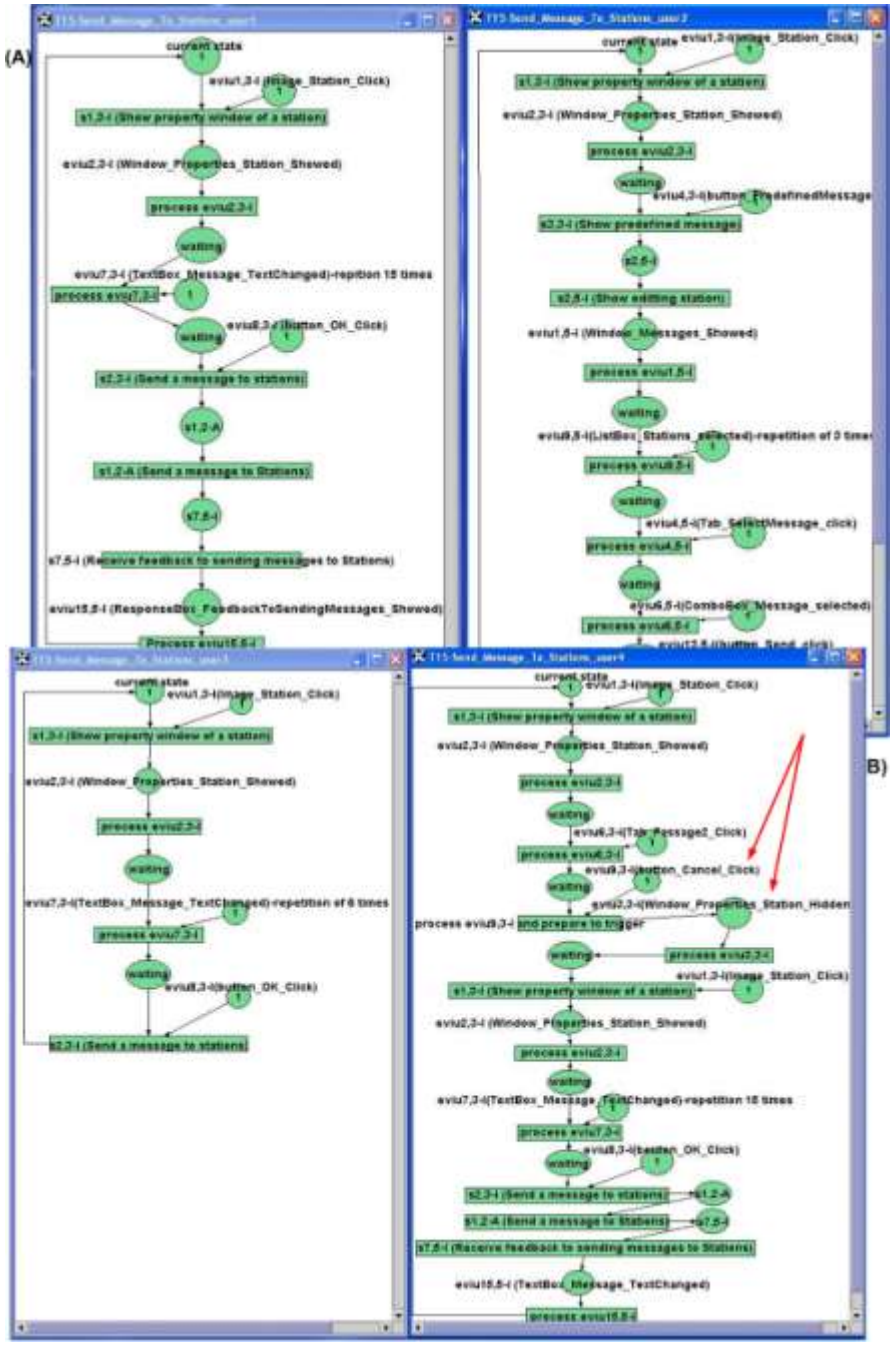

Fig. 6. Confrontation between PNs to realize the task "Send a message to stations"
(A) Comparing generated PNs of user 1 (left side) with PNs of user 2 (right side)
(B) Comparing generated PNs of user 3 (left side) with PNs of user 4 (right side)

(3) The list of criteria is open and modifiable. The evaluator has to be able to add new criteria or modify the old criteria. He/she can use the criteria from available sources or determine the criteria by himself/herself. (4) The criteria can be generic or specific for the evaluated application. For example, the criteria such as *"response time", "complexity", "immediate feedback"* are generic and can be used to evaluate all the agent-based interactive systems but the criteria such as *"Does the regulator find easily the necessary vehicle?"*, *"Does the regulator find easily the necessary station?"* are specific for the agent-based interactive system in the SART project. These specific criteria influence much the satisfaction of user of this system. The figure 7 illustrates this module. Each criterion is composed from 4 parts: *title; definition; interpretation* (explain to the evaluator how to use the collected data as well as its analysis results to evaluate this criterion); *evaluating this criterion* (according to this criterion, the evaluator enters his/her critiques for the system and suggestions to fix it). Because of lack of place, it is not possible to describe in detail this module in the paper.

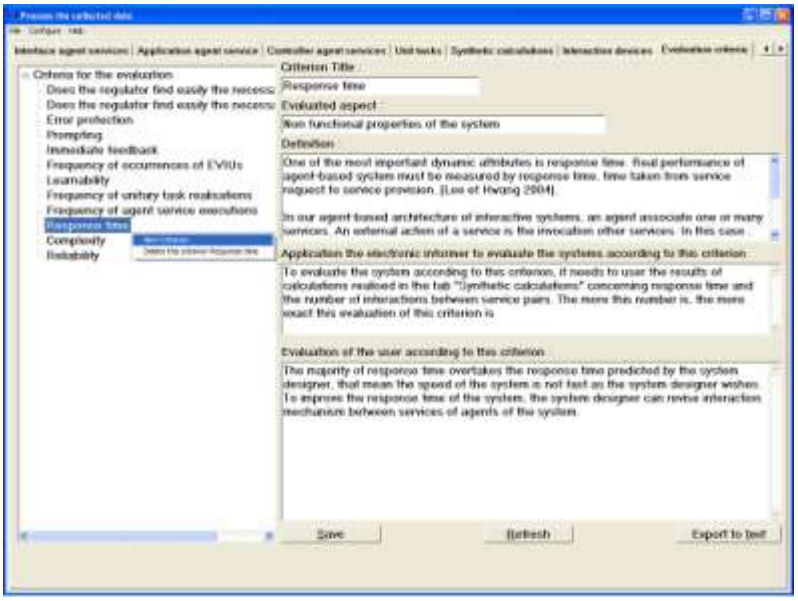

Fig. 7. Screen page of the module 6

M7: this module enables the evaluator to configure electronic informer to evaluate different agent-oriented systems by entering the specific data of the evaluated system, for example, the *Specification of Agents* that describes agents, associated services and so on., the tasks that the user can realize to reach his/her goal and some other configuration parameters.

5 Conclusion

We have presented a brief state of the art concerning interactive system architectures, and proposed a mixed architecture as well as a generic and configurable

model for assisting the evaluation for agent-oriented interactive systems. An electronic informer has been proposed and its constitutive modules have been described. We intend to combine this "electronic informer" method with other methods (questionnaire, interview…). That needs to combine data collected from "the electronic informer" and data collected from the other methods to evaluate more efficiently such systems. An experiment is planned during the first semester of 2008 at the laboratory LAMIH, University of Valenciennes and Hainaut-Cambrésis, France with about 5-10 participants. This experiment is the final stage of the SART project and it will show the advantages as well as inconveniences of this tool to help us determine the improvements in the future.

ACKNOWLEDGMENTS

The present research work has been partially supported by the "Ministère de l'Education Nationale, de la Recherche et de la Technologie», the «Région Nord Pas-de-Calais», the FEDER (MIAOU, EUCUE, SART), the ANR ADEME (Viatic.Mobilité), the PREDIM (MouverPerso).

References

Bass L, Little R, Pellegrino R, Reed S (1991) The Arch Model: Seeheim revisited. Proceedings of User Interface Developpers'Workshop, Seeheim.

Bastien JMC, Scapin DL (1995) Evaluating a user interface with ergonomic criteria, In: Int. Journal of Human-Computer Interaction, vol. 7, p. 105-121.

Coutaz J (1987) PAC, an Object-Oriented Model for Dialog Design. In: Bullinger, Hans-Jorg, Shackel, Brian (ed.), INTERACT 87, 2nd IFIP International Conference on Human-Computer Interaction, September 1-4, 1987, Stuttgart, Germany, p.431-436.

Coutaz J, Nigay L (2001) Architecture logicielle conceptuelle des systèmes interactifs, chapter 7 of «Analyse et Conception de l'Interaction Homme-Machine dans les systèmes d'information». In Kolski (Ed.), Paris: Éditions Hermes, pp. 207-246.

Etgen M, Cantor J (1999) What does getting WET (Web Event-logging Tool) Mean for Web Usability? User Experience Engineering Division AT&T Labs Middletown, NJ, USA 1999

Goldberg A (1983) Smaltalk-80, the interactive programming environnement. Addison-Wesley.

Grislin-Le Strugeon E, Adam E, Kolski C (2001) Agents intelligents en interaction homme-machine dans les systèmes d'information. In: Kolski C. (Ed.), Environnements évolués et évaluation de l'IHM, pp. 207-248, Paris: Éditions Hermes.

Hong IJ, Heer J, Waterson S (2001) WebQuilt: A Proxy-based Approach to Remote Web Usability Testing. In *ACM Transactions on Information Systems*, 2001, pp. 263-385.

Nielsen J (1993) Usability Engineering, Academic Press, Boston.

Paganelli L., Paterno F (2002) Intelligent Analysis of User Interactions with Web Applications. In Proceedings of ACM IUI 2002 (San Francisco, CA, January 2002).

Paterno F, Ballardin G (2000) RemUSINE: a bridge between empirical and model-based evaluation when evaluators and users are distant. Interacting with Computers, 13(2), 229–251.

Pfaff GE (1985) User interface management system. Springer-Verlag.

SART (2007) Système d'Aide à la Régulation de Trafic du réseau de transport valenciennois et de ses pôles d'échanges. Final report, co-operative project SART, INRETS, France, Dec.

Shneiderman B (1998) Designing the user interface : strategies for effective human-computer interaction. Addison-Wesley.

Tarpin-Bernard F, David B (1999) AMF : un modèle d'architecture multi-agents multi-facettes. In: TSI, Vol. 18, No. 5, 555-586.

Trabelsi A, Ezzedine H, Kolski C (2004) Architecture modelling and evaluation of agent-based interactive systems. In: Proc. IEEE SMC 2004, The Hague, October, pp. 5159-5164.