A version of this paper has been published in: *Journal of Universal Computer Science*, 14, pp. 3202-3216, 2008.

Evaluation of Agent-based Interactive Systems: Proposal of an Electronic Informer Using Petri Nets

Chi Dung Tran, Houcine Ezzedine, Christophe Kolski
(LAMIH, University of Valenciennes and Hainaut Cambresis, France
{ChiDung.Tran, Houcine.Ezzedine, Christophe.Kolski}@univ-valenciennes.fr)

Abstract: The evaluation of interactive systems has been an active subject of research for many years. Many methods and tools have been proposed but most of them do not take architectural specificities of agent-based interactive systems into account. In addition, electronic informers are popular evaluation tools but current ones have often some limits. In order to solve these problems, we propose an electronic informer to evaluate agent-based interactive systems. This tool captures interaction events occurred in agent-based interactive systems and then, based on such captured data, it realizes treatments such as calculations, statistics and generates Petri Nets (PNs) to assist evaluators in evaluating three aspects of the system: user interface, non-functional properties (e.g. response time, reliability, etc.) and user's properties (e.g. abilities, preferences, etc.). The approach has been validated by applying it to evaluate an agent-based interactive system used for the supervision of urban transport network.

Keywords: human-computer interaction, user interface (UI), evaluation, interactive systems, electronic informer (EI), software architecture, interface agents.

1 Introduction

1.1 Architecture of interactive systems

The architectural design of interactive systems is object of many researches since the eighties. Architecture of an interactive system is composed by components, the outside visible properties of these components and the relations between them [Coutaz, 01]. In general, the proposed models respect a same principle: the separation of the user interface part from functional core (application) part of an interactive application; as a result, the flexibility, reusability and maintainability of applications are increased. We can distinguish two types of architectural models:
- Functional models, such as the well-known Seeheim and Arch models [Bass, 91]; functional models split an interactive system into several functional components; for instance, the Seeheim model is made up of three logical components (Presentation (seen by the user), Dialogue Controller, Application Interface).
- Structural models, such as PAC [Coutaz, 87], and its variations, AMF [Tarpin-Bernard, 99] or MVC [Goldberg, 83] and its variations; the structural models aim at a finer breakdown. Indeed, such models regroup functions together into one autonomous and cooperative entity (often called agent). They are agent-based interactive systems that are built based on a hierarchical structure of agents in accordance with the principle of composition or communication, not on a functional division like functional models. For example, PAC model is a hierarchical structure of interactive agents; an agent in PAC model is composed by three facets: *Presentation* that connects agents to the input/output devices, *Abstraction* is responsible of functional core of the application, *Control* plays an intermediary role between the two other components and serves communications between PAC agents. Three facets (Model, View and the Controller) also compose an agent of another model, MVC.

Functional or agent-based interactive systems of structural models, each of them has their own advantages and disadvantages (not presented here because of lacking place). In order to exploit the advantages of both types of models, we propose an agent-based architectural model that borrows principles of both of them; so this model can be considered as a mixed model. The idea of a mixed model is not new but our proposed agent-based architectural model aims principally at (1) designing complex supervision systems in industrial context, (2) proposing solutions for the evaluation phase (see [Trabelsi, 04; Tran, 08]). We will present our architecture in the section 2.

1.2 Electronic Informers (EIs) in order to evaluate interactive systems

Before presenting our EI, we introduce here some other EIs and discuss about their drawbacks. It is motivation of our EI proposal.

Traditional electronic informers collect automatically, in a real situation, users' actions and their repercussions on the system, in a discreet and transparent way for the user. Then, collected data can be analysed and visualized in a synthetic representation to the evaluator. This facilitates the analysis of collected data.

One needs to distinguish between EIs and "feedback quality agents" which are 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, the state of the application when it had problem such as OS Version, Processor Type, Display Type, register, functions that were called just before the failure, etc. These data are sent to the development team members to help them detect problems and the cause of the crash more readily and then, improve the future version of the application. "Feedback quality agents" can also permit the user to report what he/she was doing with the application when the failure appears. These "feedback quality agents" only capture some information on context and state of the application when it had problem; it is completely different from EIs which capture interactions between users and the application in real situation of use for later analysis in order to make the application user-friendlier. As a result, for the evaluation of interactive applications, "quality feedback agents" are very limited comparing to EIs.

Many EIs are currently available; we can mention here some representative tools:

- WET [Etgen, 99] collects only interaction data between user and the interface without any later analysis. In fact, WET is only a method to collect interaction data, and their analysis is then manual.
- RemUsine [Paterno, 00], WebRemUsine [Paganelli, 02], MultiDevice RemUsine [Paterno, 07] use the task model CTTE [Mori, 02] to realize some analysis on collected interaction data such as calculations, summary statistics concerning executed tasks (number of executed tasks and taken time) and Web pages visited by the user and then, visualize analysis results to the evaluator in terms of diagrams. These tools can distinguish 2 types of errors from the user (precondition errors and useless actions); they can determine which task has failed (because their preconditions were not satisfied) as well as how many times each task failed, and find task patterns to identify sequences of tasks that are frequently performed by the users. In particular, MultiDevice RemUsine evaluates applications on mobile devices; so, it must capture not only interaction data but also contextual information such as user location, signal power of network, battery capacity, etc., for the evaluation.
- WebQuilt [Hong, 01] represents sequences of visited Web pages in terms of an interactive directed graph. It realizes some calculations, summary statistics such as which pages the user has consulted for the longest time, navigation ways of users as well as its frequency, etc. This tool can also help the evaluator in detecting discordance between real navigation ways and the ones expected by the designers.
- EDEM [Hilbert, 98] uses "expectation agents" to capture only interactions between the user and interface that violate "usage expectation" that has been specified before by developers. Such agents are only able to generate events at high abstract levels from the captured corresponding ones at low level. The other analysis is manual. In fact, EDEM is only a little more powerful than a method to collect interaction data.

More, these current EI tools do not take architectural specificities of an agent-based interactive system into account for the evaluation. In fact, in previous works of our research team, [Trabelsi, 04] has ever proposed an EI that takes into account architectural specificities of an agent-based interactive system; nevertheless it is not a generic tool but only a specific one. This specific EI is dedicated only to the evaluation of a specific agent-oriented interactive application called IAS (Information Assistance System) intended to manage the passenger information on a public transport network in a project called SART [SART, 07; Ezzedine, 08]. This tool cannot be used to evaluate other agent-based interactive applications because it depends on the number of agents; on the structure and the content of the specific application IAS, as well as all its treatments are specific to IAS. Furthermore, if the IAS has a certain modification or development, then there must be a corresponding modification or development in source code of this evaluation tool. Finally, this tool captures only interactions between user and interface; interactions between agents themselves are not captured. In order to solve such problems, we propose and develop a generic and configurable model of an EI that takes architectural specificities of an agent-based interactive system into account for the evaluation. Moreover, this proposed EI goes 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 ability of users to operate systems. This tool has been technically validated by applying it to evaluate an agent-based system used for the supervision of urban transport network (bus, tram). We will present this EI and a part of our experiment results, including Petri Nets generated by this tool, in the section 2.

1.3 Petri Nets

Petri Nets (PNs) were proposed in the beginning of the sixties by C.A. Petri [Petri, 62]. They allow to formalize and analyse behaviours of dynamic systems in which events and concurrence usually occur. Modelling and design of discrete event systems are successful application areas of PNs. PNs also permits to evaluate the performance of modelled systems. Three types of objects compose the elementary PNs: (1) places (depicted by circles) represent states of the system, (2) transitions (depicted by boxes or bars) represent operators that make changes of states, (3) arcs are used for connections between transitions and places. If there is a directed arc connecting a place to a transition, then this place is input place of this transition. If there is a directed arc connecting a transition to a place, then this place is output place of this transition.

In our approach, places and transitions of PNs describe 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. Our proposed EI (presented in the section 2) permit to generate PNs; these generated PNs are very useful for evaluators to do their work. The designer can also use PNs to predict different ways to realize a certain task. We will present some PNs in the section 2.

2 Proposal

2.1 Proposed agent-based architecture: basic concepts and discussion

Our agent-based architecture of interactive systems that can be considered as mixed model [Grislin-Le Strugeon, 01] divides an interactive systems into three functional components (as in Seeheim model) called respectively: *interface with the application* (connected to the application; in our application case: a supervised system), *dialogue controller*, and *interface* or *presentation* (Figure 1). Each of these components is broken down further in a structural approach in the form of agents; these components are built like three multi-agent systems (MAS) that can work in parallel, at least, at a theoretical viewpoint.

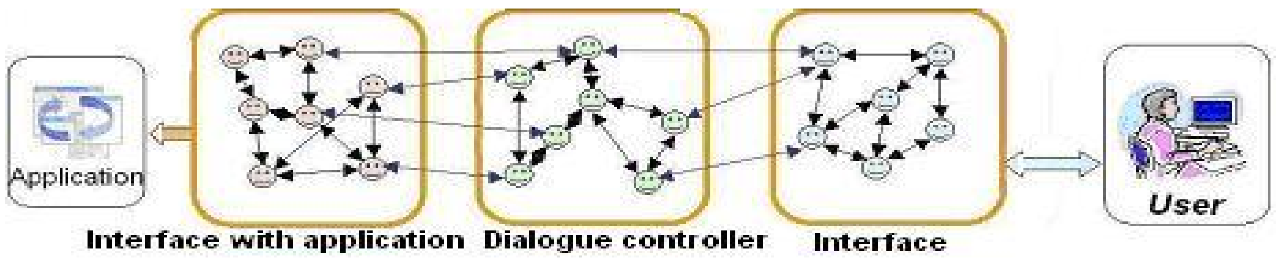


Figure 1: proposed agent-based architectural model of interactive systems

The user cannot directly access to the application agents that manipulate the field concepts of the application and that execute specific functions of application domain. One of their roles is to ensure the real time transmission of the necessary information for other agents in order to perform their tasks. The interface agents are in direct contact with the user (they can be seen by the user) through the associated UI events. These interface agents co-ordinate between them in order to: (1) intercept the user commands and then send these commands to the application through other agents; in supervision systems, this role (of interface agents) allows users (who are human operators in control room) to perform their regulations on the real dynamic process; (2) form a presentation that allows the user to gain an overall understanding of the current state of the application; in supervision systems, this helps the human operators control the process. 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 agents communicate amongst themselves in order to answer the user actions or events from the application. Each agent of this architecture associates with a set of services that are the actions this agent can execute. The communication between agents is realized by the interactions between services of agents.

In this model, three types of events can occur at two abstract levels. Events generated by interaction devices (such as a pressed keyboard button, clicked left button of mouse, etc.) are at the lowest abstract level, like the model proposed by [Hilbert, 00]. At the higher level, there are two types of events: (1) UI events (also called *eviu*-events for interactions with users) are events relating to interface objects on screen of the application such as *button_clicked*, *window_showed*, *window_closed*, *menu_item_selected*, etc. These events can only occur on interface agents. Each event of such type can correspond to one or many interaction device events at lowest level; that means there are many ways on interaction devices to trigger one UI event. (2) Executed services of agents that can occur on all three types of agents. Due to the availability of these different types of events as well as their formal description (not presented because of lacking place), we propose a generic and configurable tool, called "electronic informer" (EI) and designed to assist evaluators in analysing and evaluating interactive systems based on such architecture.

2.2 Proposed electronic informer (EI)

The proposed EI is made up of 7 main modules (Figure 2) so that the developers can modify a certain module without affecting other ones. The evaluator can configure it to evaluate different agent-based interactive systems.

This proposed EI tool exploits architectural specificities of an agent-based interactive system to evaluate it. In particular, this tool captures 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 to evaluate three aspects of an agent-based interactive system: UI, some non-functional properties such as response time, reliability, complexity, etc., and ability of users to operate systems. This exploitation will bring the evaluator better detection and understanding about problems concerning the evaluated application. The evaluator can answer more rapidly questions like: Which services of agents has often troubles (they fails or execute too long, etc.)? Which interactions between services take usually too long (that slows system down)? Which interface agents rarely interact with user or which application agents rarely work?, etc. From answers to these questions, the evaluator can determine clearly which modifications must be done to improve the system and transmit useful suggestions to the designers.

On the other hand, current EIs usually retrieve captured UI events to realize analysis on these data such as: searching, transformations, counts, statistics, and finally visualize analysis results to the evaluator. They often stop there, at this stage. The interpretation of these analysis results for the evaluation is done by only the evaluator, and of course, the evaluation results depend on the experience and ability of evaluators. Results drawn by different evaluators can be different. We intend to go further to assist evaluators in interpreting these analysis results. Each module is presented below and some experiment results are used as examples.

2.2.1 Module 1 (M1)-Collecting events from interactive systems

This module can run in background to capture events that occur (generated interaction device events, occurred UI events or events for interactions with users (*eviu)* and executed services) from all agents of the evaluated system and save them into a database that will be exploited by other modules (Figure 2). This module 1 and evaluated system can run on the same machine, or on two different ones on the network. That means the evaluation can be done remotely. The module 1 and other modules do not communicate directly with each other. The data collection and its treatment are separated.

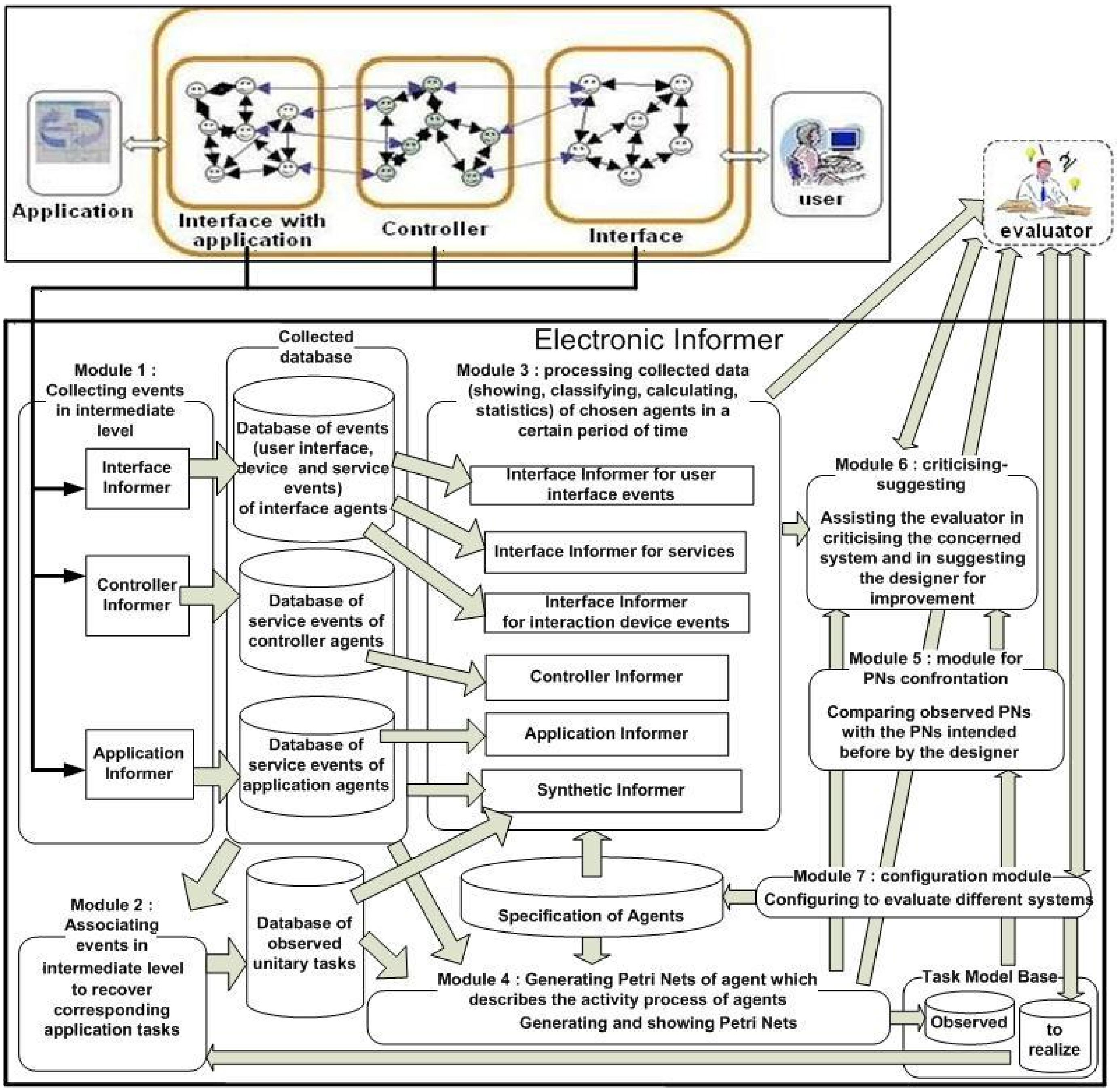

Application
Interface with application
Controller
Interface
user
evaluator
Electronic Informer
Module 1 : Collecting events in intermediate level
Interface Informer
Controller Informer
Application Informer
Collected database
Database of events (user interface, device and service events) of interface agents
Database of service events of controller agents
Database of service events of application agents
Module 3 : processing collected data (showing, classifying, calculating, statistics) of chosen agents in a certain period of time
Interface Informer for user interface events
Interface Informer for services
Interface Informer for interaction device events
Controller Informer
Application Informer
Synthetic Informer
Module 6 : criticising-suggesting
Assisting the evaluator in criticising the concerned system and in suggesting the designer for improvement
Module 5 : module for PNs confrontation
Comparing observed PNs with the PNs intended before by the designer
Module 2 : Associating events in intermediate level to recover corresponding application tasks
Database of observed unitary tasks
Specification of Agents
Module 7 : configuration module
Configuring to evaluate different systems
Module 4 : Generating Petri Nets of agent which describes the activity process of agents
Generating and showing Petri Nets
Task Model Base
Observed
to realize

Figure 2: Our proposed generic and configurable EI (electronic informer)

2.2.2 M2-Associating intermediate level events collected by the module 1 to corresponding task

3 abstract levels of events are distinguished. Along with events at two levels that were above presented, and that module 1 can capture, the events at the highest level are tasks that user or system must realize. The proposal of these highest level events helps evaluators (1) avoid work overload because of enormous quantity of tedious events at lower levels, (2) understand the motivation, intention and action processes of users and system for a certain objective. A task can be user task or system task. An user task is triggered by a sequences of user's actions in terms of UI events on interface agents, and then, these events will trigger automatically other events (UI events or services) of same or other agents to realize this task. In supervision systems, user tasks are usually corresponding to operator's activities as commander for regulations or interventions in necessary cases. A system task is automatically triggered by a service of an application agent (or interface agent in uncompleted models) and then, this service will trigger automatically other events (UI events or services) of same or other agents to realize this task. In supervision systems, system tasks are usually realized when: (1) application agents inform operators about current state of the application so that they can play their controller's role, (2) application agents detect a certain problem in the application and they inform operator about it, so that operators can intervene or regulate as commander role. M2 enables the evaluator to associate captured events at intermediate level (UI events and services) to their corresponding task called "observed task". Evaluators need only to work with services or user's actions in term of UI events on interface agents that trigger tasks; events that are triggered by these events will be automatically treated. Here is an example from the supervision application IAS for urban transport network: UI events

A version of this paper has been published in: *Journal of Universal Computer Science*, 14, pp. 3202-3216, 2008.

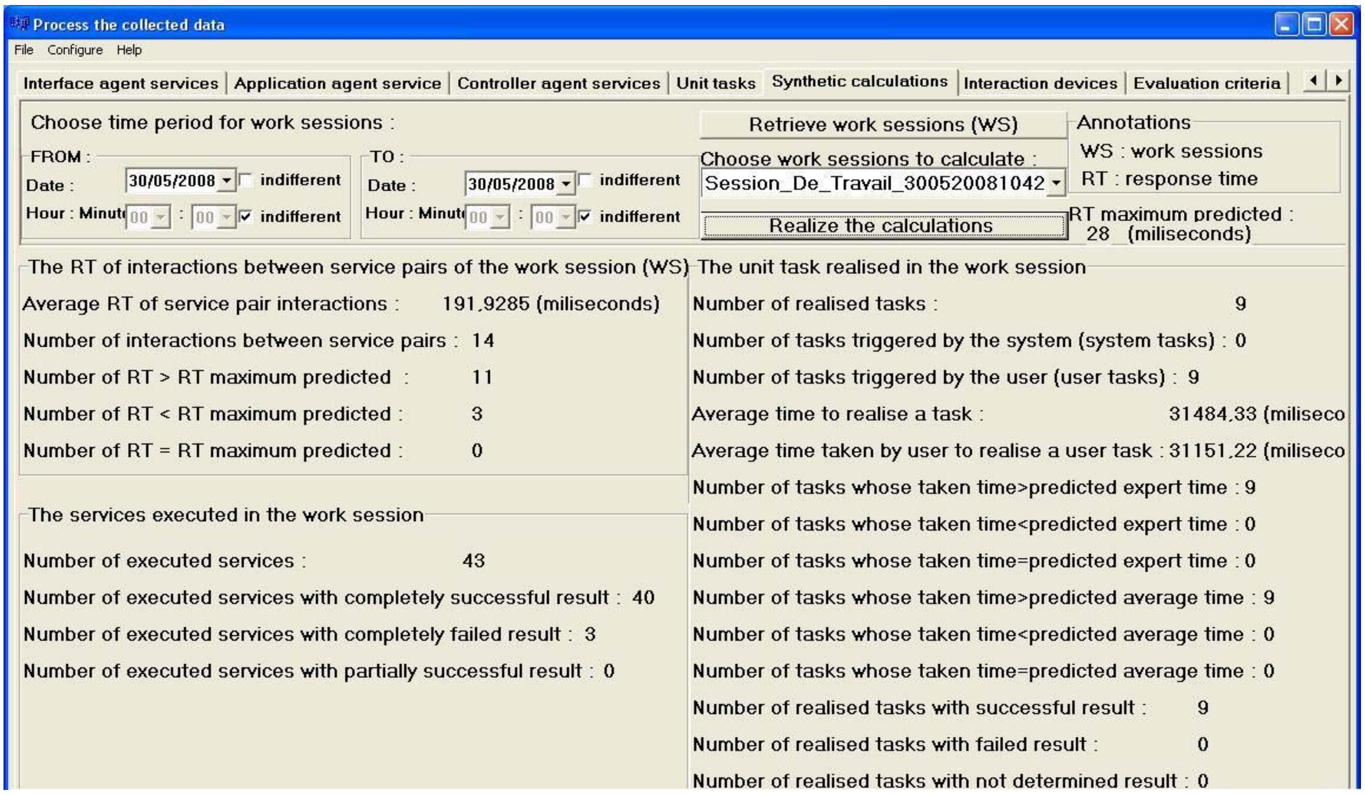
Image_Station_Click, *TextBoxMessage_TextChanged* and *buttonOK_Click* can be associated to the task *"Send a message to passengers at station"*.

2.2.3 M3-analysing data captured by the module 1 and tasks

This module realizes synthetic calculations and statistics on data of events (interaction device and UI events, tasks, services) such as number and ratio 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 a any chosen period of time with a any chosen work session and then, show these analysis results in table or graph forms. These results provide evaluators with necessary objective data to be interpreted with the indications of module 6 (that permits to associate them to determined criteria to criticise the system) and propose useful suggestions to the designers for improvements. M3 generates several screen pages. Figures 3 and 4 show two screen pages of this module containing analysis results.

2.2.4 M4 and M5-generating and comparing Petri Nets (PNs)

These two modules enable evaluators to generate PNs and compare generated PNs with each other or with theoretical PNs specified before by designers. Generated PNs called observed PNs describe 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. These generated PNs are very useful for evaluators to do their work. Thanks to them, evaluators can know in a visual manner about processes of real activities of users, of the system as well as interactions between them. These PNs can be interpreted with the indications of M6 (presented below) that permits to associate them to determined criteria to criticise the system and draw useful suggestions for improvement purpose. The evaluators can compare observed PNs to realize a certain task of a certain user with theoretical PNs (PNs predicted by the designers) for the same task or with observed PNs to realize the same task of other users. Such comparisons are very useful for evaluators to detect problems coming from the interface, the system or the users. For example, the evaluator can perceive: useless actions of users, non-optimal way chosen by users to realize tasks, service interactions that has failed, etc. The evaluator can use comparisons of PNs (concerning different users) to learn about habits of users, to compare abilities of different users, to supervise the progress of abilities of a certain user, etc. Like observed PNs with their visual views, these comparisons can also be interpreted with the indications of M6 to help evaluators do their work.


Process the collected data
File Configure Help
Interface agent services | Application agent service | Controller agent services | Unit tasks | Synthetic calculations | Interaction devices | Evaluation criteria
Retrieve work sessions (WS)
Realize the calculations

Choose time period for work sessions :

FROM : Date : 30/05/2008 ☐ indifferent
Hour : Minute 00 : 00 ☑ indifferent

TO : Date : 30/05/2008 ☐ indifferent
Hour : Minute 00 : 00 ☑ indifferent

Choose work sessions to calculate : Session_De_Travail_300520081042

Annotations
WS : work sessions
RT : response time

RT maximum predicted : 28 (miliseconds)

The RT of interactions between service pairs of the work session (WS)

Average RT of service pair interactions :	191,9285 (miliseconds)
Number of interactions between service pairs :	14
Number of RT > RT maximum predicted :	11
Number of RT < RT maximum predicted :	3
Number of RT = RT maximum predicted :	0

The services executed in the work session

Number of executed services :	43
Number of executed services with completely successful result :	40
Number of executed services with completely failed result :	3
Number of executed services with partially successful result :	0

The unit task realised in the work session

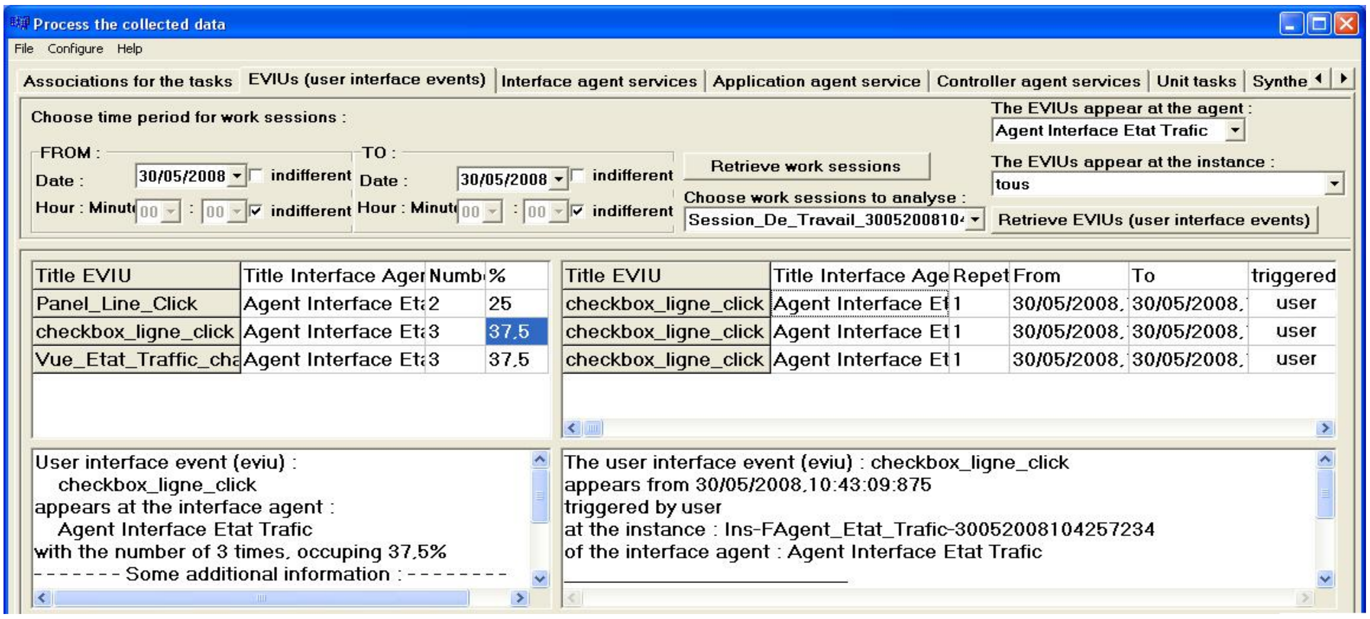
Number of realised tasks :	9
Number of tasks triggered by the system (system tasks) :	0
Number of tasks triggered by the user (user tasks) :	9
Average time to realise a task :	31484,33 (miliseco
Average time taken by user to realise a user task :	31151,22 (miliseco
Number of tasks whose taken time>predicted expert time :	9
Number of tasks whose taken time<predicted expert time :	0
Number of tasks whose taken time=predicted expert time :	0
Number of tasks whose taken time>predicted average time :	9
Number of tasks whose taken time<predicted average time :	0
Number of tasks whose taken time=predicted average time :	0
Number of realised tasks with successful result :	9
Number of realised tasks with failed result :	0
Number of realised tasks with not determined result :	0

Figure 3: One screen page of M3: calculations (number of successful tasks, average response time, comparison between real and predicted calculations, etc.)

Figures 5 and 6 illustrate (1) two generated PNs, from two male users (user #4 and #6) who have realized the task *"Send a message to passengers at station"* (using the application IAS) in our experiment and (2) a theoretical PN (please note that eviuM,N-I stands for the UI event M of the interface agent N, sM,N-I stands for the service M of the interface agent N). From these two PNs, the evaluator can perceive that:

- the user #4 has chosen the optimal way among three different ones (marked by the symbol *** in theoretical PN) to realize the task; he had no useless actions but when he typed the message to send, he could not type continuously it. Indeed, he typed some characters and stopped, and continued to type, etc. These were bad manipulations marked by the symbol * in his PN on the left. The evaluator can perceive such manipulations by the UI event *eviu7,3-I (Textbox_Message_TextChanged)* repeated 5, 12, 8, 14, 14 times. Of course, these are not very good because the time taken to realize the task can be longer and the module 3 verified it. The time taken to realize this task is longer than the time predicted by the designer to realize it in both cases of average user or expert user.

- The user #6 has chosen the optimal way, too but he has done many useless actions (marked by rectangle and arrow on the Figure 6): first, he clicked on *Image Vehicle* on the screen through UI event *Image_Vehicle_Click* to open the window of properties of this vehicle, and then, he did so many other actions on this window, but these were wrong actions of this user because he had to send a message to a station, not to a vehicle, so, he has clicked on the button Cancel on this window through UI event eviu16,4-I(*button_Cancel_Click)* to close it. In consequences, the UI event eviu9,4-I (*Window_Properties_Vehicle_Hidden)* was triggered and he had to return to the event *Image_Station_Click* again. The way this user has chosen to realize this task was the same as the way the user 4 on the figure 5 has chosen (marked by the symbol *** in theoretical PN of the figure). But this user #6 has the same bad manipulations, too. He even typed much more discretely than user #4 (marked by the symbol * in his PN on the figure 6. For the reason of clarity, the figure 6 only illustrates a part of PN of this user.) and M3 let the evaluator know that the time taken to realize this task is much longer than the time taken by the user #4. The evaluator can evaluate that user #4 is better than user #6, at least, for this task.


Process the collected data
File Configure Help
Associations for the tasks
EVIUs (user interface events)
Interface agent services
Application agent service
Controller agent services
Unit tasks
Synthe
Retrieve work sessions
Retrieve EVIUs (user interface events)

Choose time period for work sessions :

FROM : Date : 30/05/2008 ☐ indifferent
Hour : Minut 00 : 00 ☑ indifferent

TO : Date : 30/05/2008 ☐ indifferent
Hour : Minut 00 : 00 ☑ indifferent

Choose work sessions to analyse : Session_De_Travail_300520081O

The EVIUs appear at the agent : Agent Interface Etat Trafic

The EVIUs appear at the instance : tous

Title EVIU	Title Interface Ager	Numb	%
Panel_Line_Click	Agent Interface Et	2	25
checkbox_ligne_click	Agent Interface Et	3	37,5
Vue_Etat_Traffic_cha	Agent Interface Et	3	37,5

Title EVIU	Title Interface Age	Repet	From	To	triggered
checkbox_ligne_click	Agent Interface Et	1	30/05/2008,	30/05/2008,	user
checkbox_ligne_click	Agent Interface Et	1	30/05/2008,	30/05/2008,	user
checkbox_ligne_click	Agent Interface Et	1	30/05/2008,	30/05/2008,	user

```
User interface event (eviu) :
   checkbox_ligne_click
appears at the interface agent :
   Agent Interface Etat Trafic
with the number of 3 times, occuping 37,5%
------- Some additional information : --------
```

```
The user interface event (eviu) : checkbox_ligne_click
appears from 30/05/2008,10:43:09:875
triggered by user
at the instance : Ins-FAgent_Etat_Trafic-30052008104257234
of the interface agent : Agent Interface Etat Trafic
```

Figure 4: One of screen pages of M3: Statistics of occurred UI events on a chosen interface agent for chosen period of time with a chosen work session in table form

2.2.5 M6-Critising and suggesting for improvements

The modules 2, 3, 4, 5 realize analysis on collected data. These analysis results must be interpreted by evaluators to evaluate the system. We intend to assist evaluators in doing it: this module 6 aims at this purpose. M6 associates analysis results of other modules to an open list of determined criteria which can be ergonomic criteria or quality attributes. In fact, [Bernonville, 06] has presented a method called ErgoPNets that enables evaluators to associate PNs, corresponding with human-computer interactions, to ergonomic criteria (from [Bastien and Scapin, 94]) in order to show usability problems. Although there is something common, this method is different from ours:

- ErgoPNets currently works with ergonomic criteria proposed by Bastien and Scapin, whereas the criteria list of M6 is open: evaluators can add, update, or delete criteria if necessary. It is important because there are many sources of ergonomic rules, usability models from different authors and organisations (ISO 9241, ISO/IEC 9126, [Smith and Mosier, 86], [Bastien and Scapin, 94], [Abran et al., 03], etc.) with additions, updates, and integrations.
- ErgoPNets with its criteria aims only at ergonomic problems of UI. This is only one of three aspects at which M6 aims. Moreover, specificities of agent-based interactive systems are not taken into account in ErgoPNets.

A version of this paper has been published in: *Journal of Universal Computer Science*, 14, pp. 3202-3216, 2008.

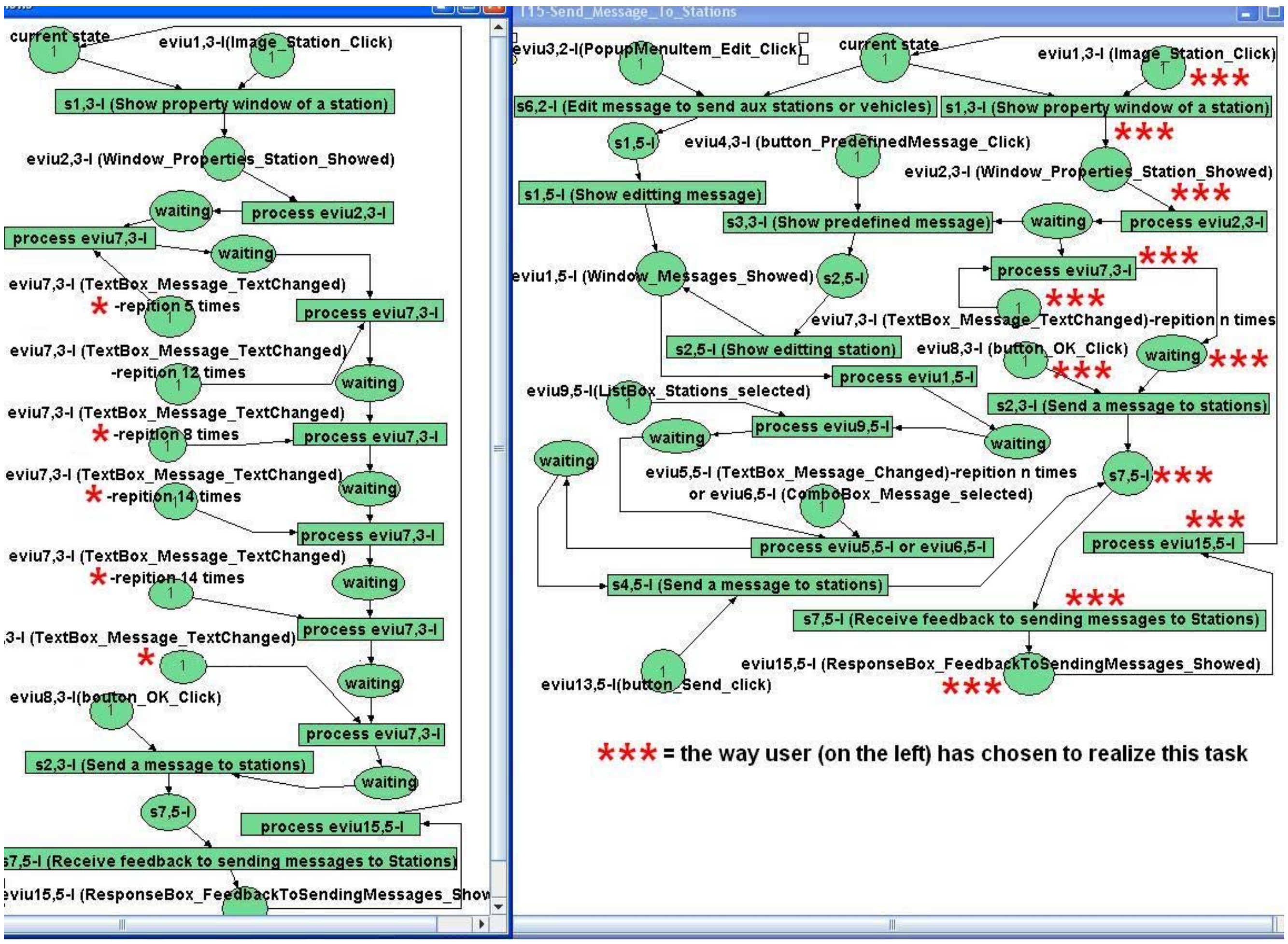

current state
eviu1,3-I(Image_Station_Click)
s1,3-I (Show property window of a station)
eviu2,3-I (Window_Properties_Station_Showed)
waiting
process eviu2,3-I
process eviu7,3-I
eviu7,3-I (TextBox_Message_TextChanged)
-repition 5 times
-repition 12 times
-repition 8 times
-repition 14 times
eviu8,3-I(bouton_OK_Click)
s2,3-I (Send a message to stations)
s7,5-I
process eviu15,5-I
s7,5-I (Receive feedback to sending messages to Stations)
eviu15,5-I (ResponseBox_FeedbackToSendingMessages_Showed)
T15-Send_Message_To_Stations
eviu3,2-I(PopupMenuItem_Edit_Click)
eviu1,3-I (Image_Station_Click)
s6,2-I (Edit message to send aux stations or vehicles)
s1,5-I
eviu4,3-I (button_PredefinedMessage_Click)
s1,5-I (Show editting message)
s3,3-I (Show predefined message)
eviu1,5-I (Window_Messages_Showed)
s2,5-I
eviu7,3-I (TextBox_Message_TextChanged)-repition n times
s2,5-I (Show editting station)
eviu8,3-I (button_OK_Click)
process eviu1,5-I
eviu9,5-I(ListBox_Stations_selected)
s2,3-I (Send a message to stations)
process eviu9,5-I
eviu5,5-I (TextBox_Message_Changed)-repition n times
or eviu6,5-I (ComboBox_Message_selected)
process eviu5,5-I or eviu6,5-I
s4,5-I (Send a message to stations)
s7,5-I (Receive feedback to sending messages to Stations)
eviu13,5-I(button_Send_click)
*** = the way user (on the left) has chosen to realize this task

*Figure 5: Compare PN of user #4 (left) to realize the task "Send a message to passengers at station" with theoretical PN predicted by designer (right) for this task. The symbol *** on theoretical PN (right) illustrate the way this user (left) has chosen. The symbol * on PN of the user (left) illustrate bad manipulations of this user*

At this moment, these associations between analysis results of other modules and determined criteria of the module 6 are not formalized yet; that means current M6 can only give to evaluators an indication, a direction about how to interpret analysis results to assist them in evaluating agent-based interactive systems at all their three aspects above mentioned (UI, some non-functional properties, user's properties) and in proposing necessary modifications. This is beginning step towards formalizing and then, automating these associations as much as possible in the future. As illustration, we present here some criteria and corresponding evaluations. Because of lacking place, we cannot present all criteria or go to the details of criteria.

- In agent-based interactive systems, transmissions of information between the application agents and interface agents through control agents are very important. A slowness or delay of these transmissions can cause severe consequences, especially in industrial supervision systems because operators must understand quickly the current state of the real process that they are supervising and their commands must be sent to this real process in time if necessary, especially, in the case of malfunction. In order to have timely transmission of information between agents, service interactions between agents must be taken into account. Our EI also contributes to take into account this problem through calculating response time between services of agents: it does not only process interactions between interface and user like other EIs. Response time (RT) between services, one of the most important dynamic important attributes to evaluate and measure the real performance of a MAS [Lee, 04], is measured by the time taken from service request to service provision. The "*Response Time*" is also one of criteria used by M6. M3 can let the evaluator know service pairs whose RTs take long, this module also calculate average RT between services, the number of service pairs whose RTs are longer or faster than an determined acceptable threshold. The criterion RT can be evaluated using these results. If RT of the majority of service pairs are longer than the acceptable threshold, then the evaluator can advise the designers to revise the system, especially revise service pairs whose RTs often take longer than the acceptable threshold with high frequency. The improvement of RT of such service pairs will improve the overall performance of the system. We can extend here about which factors influence RT of systems? Different coordination patterns between agents influence differently the performance of a MAS according to [Lee, 04]; but in fact, RT is influenced not only by coordination patterns between agents but also by used interaction techniques. If designers cannot change coordination patterns between agents to improve RT because of constraints, then the evaluator can advise them to modify interactions techniques for the same purpose. These techniques are key determinants for properties such as reliability, performance [Mehta, 00; Spitznagel, 01]. There exist many interaction techniques in software systems today (see [Mehta, 00]). Designers must consider these problems when they design an agent-based interactive system.

- A criterion of M6 to evaluate UI is *"legibility"*. This criterion can be evaluated by the number of occurrences of UI events such as Zoom In, out, scroll bar, resizing windows, etc. The results are calculated by M3. If such event occurs with high frequency, then the evaluator can make a question about legibility of UI.
- In order to evaluate the application IAS, a specific criterion: *"The rapidity of regulators to find necessary stations or vehicles"* has been added to M6. When regulators want to use IAS to command drivers or inform passengers at a station or in a vehicle about a problem or event, the UI must help evaluator to find rapidly corresponding stations or vehicles on screen. It is important criterion specific to IAS because it affects work productivity of regulators and insures timely regulation. In our experiment, M3 and PNs generated by M4 let us know that except a few "super users", the others have taken a long time (some users even take about 2 minutes according to M3) for useless actions before they could do the first necessary action to send a message. Clearly, the others have difficulty to find the necessary stations, vehicles to which they must send messages and they have taken a long time to find them. Of course, this criterion of the interface IAS is not highly evaluated and some ideas to modify the interface IAS for the improvement have appeared.

2.2.6 M7-configuring EI

this module enables the evaluator to configure electronic informer to evaluate different agent-based interactive 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 users or system can realize for a certain goal and some other configuration parameters.

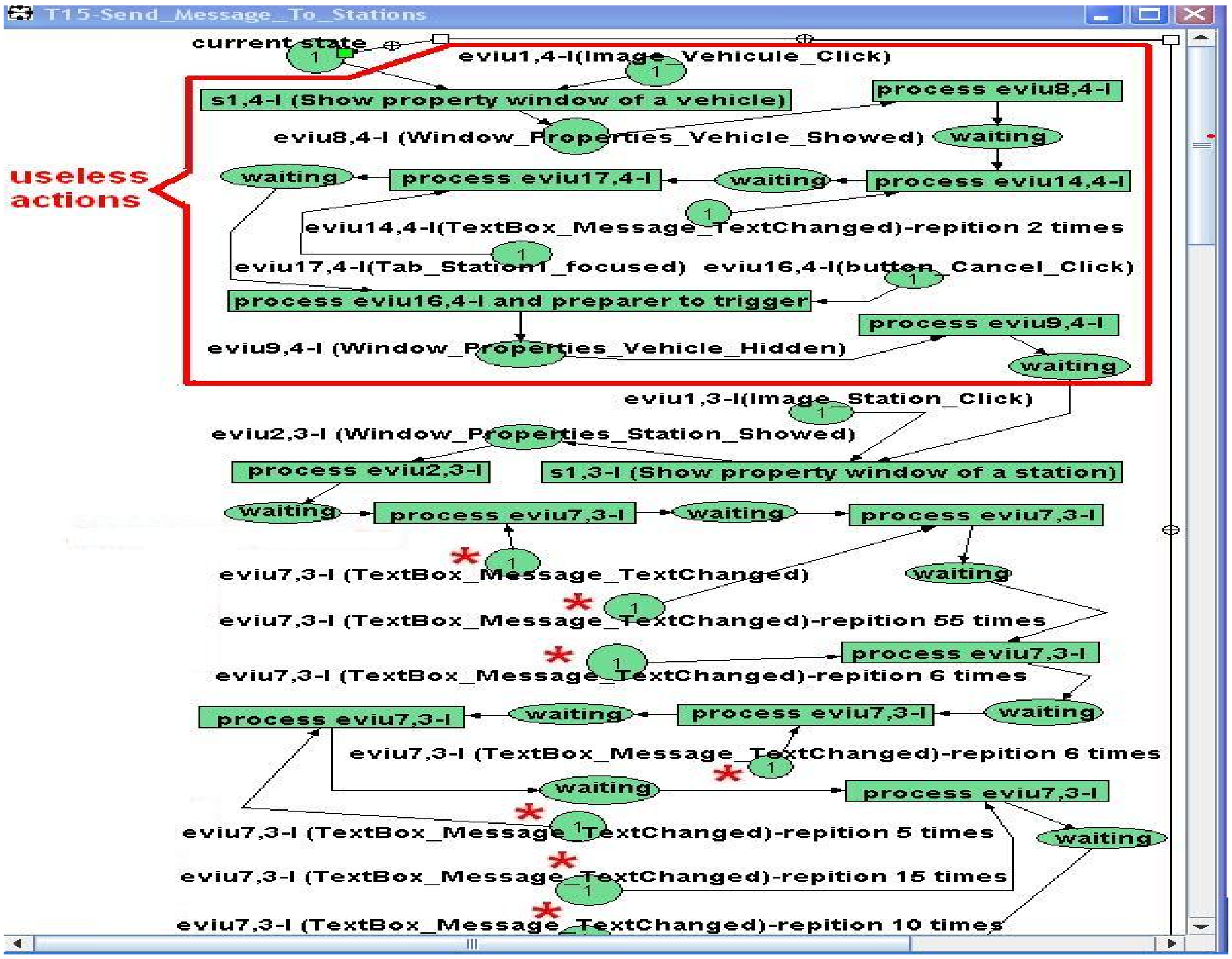

T15-Send_Message_To_Stations
current state
useless actions
eviu1,4-I(Image_Vehicule_Click)
s1,4-I (Show property window of a vehicle)
process eviu8,4-I
eviu8,4-I (Window_Properties_Vehicle_Showed)
waiting
process eviu17,4-I
process eviu14,4-I
eviu14,4-I(TextBox_Message_TextChanged)-repition 2 times
eviu17,4-I(Tab_Station1_focused)
eviu16,4-I(button_Cancel_Click)
process eviu16,4-I and preparer to trigger
process eviu9,4-I
eviu9,4-I (Window_Properties_Vehicle_Hidden)
eviu1,3-I(Image_Station_Click)
eviu2,3-I (Window_Properties_Station_Showed)
process eviu2,3-I
s1,3-I (Show property window of a station)
process eviu7,3-I
eviu7,3-I (TextBox_Message_TextChanged)
eviu7,3-I (TextBox_Message_TextChanged)-repition 55 times
eviu7,3-I (TextBox_Message_TextChanged)-repition 6 times
eviu7,3-I (TextBox_Message_TextChanged)-repition 6 times
eviu7,3-I (TextBox_Message_TextChanged)-repition 5 times
eviu7,3-I (TextBox_Message_TextChanged)-repition 15 times
eviu7,3-I (TextBox_Message_TextChanged)-repition 10 times

*Figure 6: PN of user #6, with the task "Send a message to passengers at station". The symbol * on this PN illustrate bad manipulations of this user*

3 Conclusions and Future Work

We have briefly introduced some typical architecture of two types of models: functional and structural models for interactive systems. Then, we have presented our mixed agent-based architectural model that borrows principles of both types in order to exploit their advantages. This model aims at supervision systems in industry. After mentioning and criticising some existing electronic informers (EIs), we have presented our generic and configurable electronic informer (EI), composed by 7 modules and designed to assist evaluators in analysing and evaluating agent-based interactive systems. This electronic informer has been technically validated during an experiment in our laboratory with ten test participants. This is final stage of the SART project and it helps us find improvements for the IAS. In future researches, the distribution of work and of knowledge between agents in each component or even, between agents in all three components of our architectural model must be deepened because such distribution (and service interactions) affects the speed of information transmission between agents. Criteria for this distribution should be clearly determined, especially in the case of interactive supervision systems. On the other hand, associations between criteria of the module 6 and analysis results of other modules should be formalized and then, automated as much as possible. It is not a simple problem because it get involved to the determination of a model to evaluate qualities of agent-based interactive systems at all their three aspects: UI, non-functional properties and user's properties.

In order to evaluate systems more exactly and completely, many different methods should be combined. For example, "*legibility*" of UI of the system as a criterion in the module 6 can be evaluated by the number, frequency of occurrences of UI events such as Zoom In, out, scroll bar, resizing windows, etc., from the module 3 but it can also be evaluated using TFWWG (tools for working with guidelines) to retrieve parameters of static presentation of the UI such as: size and color of text, position and size of buttons, textbox on forms, etc. These tools use objective data for the evaluation. The third way to evaluate this criterion is methods using subjective data such as questionnaires or interviews. Different evaluation tools such as: our EI, a TFWWG, a tool to create questionnaire, etc., can be combined in an integrated environment that could be object of another research perspective.

Acknowledgements

The present research work has been supported by International Campus on Safety and Intermodality in Transportation, the Nord-Pas-de-Calais Region, the European Community (FEDER). The authors gratefully acknowledge the support of these institutions.

References

[Abran et al., 03] Abran, A., Khelifi, A., Suryn, W., Seffah, A.: Consolidating the ISO Usability Models, Proceedings of 11th International Software Quality Management Conference (Springer), Glasgow, Scotland, UK, 23 – 25 April 2003.

[Bass et al., 91] Bass, L., Little, R., Pellegrino, R., Reed, S.: The Arch Model: Seeheim revisited, Proceedings of User Interface Developpers'Workshop, Seeheim, 1991.

[Bastien and Scapin, 94] Bastien, J.M.C., Scapin, L.D.: Evaluating a user interface with ergonomic criteria, research report, INRIA, N 2326, France, 1994.

[Bernonville, 06] Bernonville, S., Leroy, N., Kolski, C., Beuscart-Zéphir, M.: Explicit combination between Petri Nets and ergonomic criteria: basic principles of the ErgoPNets method, Proceedings of the 25th Edition of European Annual Conference on Human Decision-Making and Manual Control, Valenciennes, France, 27-29 September 2006.

[Coutaz, 87] Coutaz, J.: 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, Stuttgart, Germany, September 1-4, 1987, p.431-436.

[Coutaz, 01] Coutaz, J., Nigay, L.: 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, 2001, pp. 207-246.

[Etgen, 99] Etgen, M., Cantor, J.: What does getting WET (Web Event-logging Tool) Mean for Web Usability?, User Experience Engineering Division AT&T Labs Middletown, NJ, 1999.

[Ezzedine, 08] 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. International Journal of Computers, Communications & Control (IJCCC), 3, pp. 281-294, 2008.

[Goldberg, 83] Goldberg, A.: Smaltalk-80, the interactive programming environnement. Addison-Wesley, 1983.

[Grislin-Le Strugeon and al, 01] Grislin-Le Strugeon, E., Adam, E., Kolski, C.: Agents intelligents en interaction homme-machine dans les systèmes d'information, In Kolski C. (Ed.), Environnements évolués et évaluation de l'IHM, Paris: Éditions Hermes, 2001, 207-248.

[Hilbert, 00] Hilbert, D. M, Redmiles, D.F.: Extracting usability information from user interface events, ACM Computing Surveys (CSUR), 32 (4), 2000, 384-421.

[Hilbert, 98] Hilbert, D. M, Redmiles, D.F.: Agents for collecting application usage data over the Internet, In Proceedings of Autonomous Agents '98.

[Hong, 01] Hong, Jason I., Heer, J., Waterson, S., Landay,J.A: WebQuilt: A Proxy-based Approach to Remote Web Usability Testing, In *ACM Transactions on Information Systems*, 2001, 263-385.

[Lee, 04] Lee, S.K., Hwang, C.S.: Architecture modeling and evaluation for design of agent-based system, The Journal of Systems and Software 72, 2004, 195–208.

[Mehta, 00] Mehta, N. R., Medvidovic, N., Phadke, S.: Towards a Taxonomy of Software Connectors, International Conference on Software Engineering, Proceedings of the 22nd international conference on Software engineering, ACM Press, 2000.

[Mori, 02] Mori, G., Paternò, F., Santoro, C.: CTTE: Support for Developing and Analyzing Task Models for Interactive System Design, IEEE Transactions on Software Engineering, August 2002, 797-813.

[Paternò, 07] Paternò, F., Russino, A., Santoro, C.: Remote evaluation of Mobile Applications, 6th International Workshop, TAMODIA 2007, Toulouse, France, Nov. 7-9, 2007.

[Paganelli, 02] Paganelli, L., Paternò, F.: Intelligent Analysis of User Interactions with Web Applications, Proceedings of ACM IUI 2002, San Francisco, CA, January 2002.

[Paterno, 00] Paternò, F., Ballardin, G.: RemUSINE: a bridge between empirical and model-based evaluation when evaluators and users are distant, Interacting with Computers, 13(2), 2000, 229–251.

[Petri, 62] Petri, C. A.: Kommunikation mit Automaten. Schriften des IIM 3, 1-128, Universität Bonn, Institut für Instrumentelle Mathematik, Bonn, 1962.

[SART, 07] 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. 2007

[Smith, 1986] Smith, S. L., Mosier, J. N.: Guidelines for Designing User Interface Software, The MITRE Coporation, Bedford ESD-TR-86-278 MTR-10090, 1986.

[Spitznagel, 2001], Spitznagel, Garlan: A Compositional Approach for Constructing Connectors, The Working IEEE/IFIP Conference on Software Architecture (WICSA'01), Royal Netherlands Academy of Arts and Sciences Amsterdam, The Netherlands, August 2001.

[Tarpin-Bernard, 99] Tarpin-Bernard, F., David, B.: AMF : un modèle d'architecture multi-agents multi-facettes, In: TSI, Vol. 18, No. 5, 1999, 555-586.

[Trabelsi, 04] Trabelsi, A., Ezzedine, H., Kolski, C.: Architecture modelling and evaluation of agent-based interactive systems, In: Proc. IEEE SMC 2004, The Hague, Oc., 2005, 5159-5164.

[Tran, 08] TRAN, Chi Dung., Ezzedine, H., Kolski, C.: A generic and configurable electronic informer to assist the evaluation of agent-based interactive systems, 7th int. conference on Computer- Aided Design of User Interfaces, CADUI'2008, Albacete, Spain, 2008.