A version of this paper has been published in:

International Journal of Human-Computer Studies, 71 (6), pp. 725-761, 2013.

The final version is available in ScienceDirect.

EISEval, a Generic Reconfigurable Environment for Evaluating Agent-based Interactive Systems

Chi Dung TRAN[a], Houcine Ezzedine[b,c,d], Christophe Kolski[b,c,d,*]

[a]LIMA Laboratory, DRT, CEA List, Atomic Energy and Alternative Energies Commission, Saclay, France
[b]Univ Lille Nord de France, F-59000 Lille, France
[c]UVHC, LAMIH, F-59313 Valenciennes, France
[d]CNRS, UMR 8201, F-59313 Valenciennes, France

[*]*Corresponding author: Prof. Christophe Kolski christophe.kolski@univ-valenciennes.fr*

Abstract

The evaluation of interactive systems has been an active subject of research for many years. Many methods have been proposed, but most of them do not take the architectural specificities of an agent-based interactive system into account, nor do they focus on the link between architecture and evaluation. In this paper, we present an agent-based architecture model for interactive systems. Then, based on this architecture, we propose a generic, reconfigurable evaluation environment, called EISEval, designed and developed to help evaluators analyze and evaluate certain aspects of interactive systems in general and of agent-based architecture interactive systems in particular: User Interface (UI), non-functional properties (e.g., response time, complexity) and user characteristics (e.g., abilities, preferences, progress). System designers can draw useful conclusions from the evaluation results to improve the system. This environment was applied to evaluate an agent-based interactive system used to supervise an urban transport network in a study organized in laboratory.

Keywords: evaluation, user interface, electronic informer, ergonomic criteria, agent-based architecture, interactive systems, human-computer interaction (HCI).

1. Introduction

The evaluation of interactive systems, in terms of utility and usability (Bastien and Scapin, 1995; Nielsen, 1993; Shneiderman, 1998), has been an important research topic since the 1980s (Sears, 2003). A user interface (UI) aims at increasing user comfort, satisfaction and productivity and decreasing the possibility that users can make errors while they are interacting with the UI. For this reason, the evaluation is very important to help designers understand user difficulties when interacting with UI and propose future improvements.

1.1. Motivations

Many evaluation tools exist today. However, these tools have several limitations and are inconvenient. Indeed, most traditional tools attempt to generally evaluate the UI of interactive systems but do not take architectural specificities of agent-based interactive systems into account when evaluating them. Moreover, these tools usually only aim at evaluating the UI of interactive systems without considering other aspects of such systems. For instance, the non-functional properties of an agent-based interactive system (e.g., agent response times, reliability, design complexity) are not considered although these properties can be very useful for detecting the system's problems. Moreover, traditional Electronic Informer (EI) tools (i.e., tools that capture data during interactions between the user and the UI in real situations so that the data can be analyzed later) show analysis results (e.g., statistics, classifications) in visual forms to evaluators, who then must interpret these results to identify the problems with the UI and suggest improvements to designers. There is no assistance or indication to help evaluators do their work. Other tools, like TFWWG (*Tools For Working With Guidelines*) (Vanderdonckt and Farenc, 2000), mainly try to evaluate the static aspects (e.g., position, size and color of elements, text fonts) of an user interface but do not apply objective use data. In consequence, more complete evaluation environments may be envisaged.

1.2. Proposal

We propose an evaluation environment called EISEval (Environment for Interactive System Evaluation). Although EISEval's activity respects EI principles, it can remedy the aforementioned drawbacks of traditional evaluation tools in general, and of EIs in particular, as well as provide a more complete evaluation. Moreover, EISEval also uses ergonomic criteria as well as other quality criteria (e.g., response time between agents, complexity of MAS design) to help evaluators interpret captured objective data and evaluate agent-based interactive systems.

In short, its objective is to provide evaluators with several benefits: in addition to evaluating UI (like other tools), EISEval helps evaluators assess other system aspects that help improve the understanding of the drawbacks of the target system as well as of the users' difficulties, thereby enabling improvements to be suggested. Furthermore, EISEval also helps evaluators interpret analysis results in order to assess the system and suggest improvements, whereas many EIs lets evaluators interpret such results on their own. Moreover, the specificities of our proposed agent-based architecture model for evaluating interactive systems using this environment taken into account by EISEval.

1.2. Organization of the paper

This paper is organized as follows. Section 2 presents a brief state of the art for the architectures of traditional interactive systems and agent-based interactive systems. At the end of this section, we also introduce our architecture model for agent-based interactive systems. Section 3 discusses the research related to evaluation tools. The main contribution and content of this paper is our proposal - the EISEval evaluation environment, presented in the section 4. Section 5 presents the application of EISEval for evaluating an agent-based system used to supervise an urban transport network in a study organized in laboratory with a set of human subjects. Section 6 reports the summary of this study results and discuss about it. Finally section 7 presents our conclusions and prospects for future research.

2. Architecture models FOR interactive systems: a parameter to consider for interactive system evaluation

The architecture of interactive systems is not a new research topic in the HCI field. Architecture models help system designers to design, develop and validate interactive systems. Since the 1980s, several architecture models have been proposed (Bass et al., 1991; Coutaz, 1987; Goldberg, 1983; Pfaff, 1985; Tarpin-Bernard and David, 1999, etc.) to help designers build interactive applications. According to [Coutaz and Nigay 2001], an architecture model is defined by a set of structures that include components (e.g., modules, services, processes, procedures, applications, objects), the outside visible properties of these components (e.g., required resources, provided services, performance) and the relationships between them. Some well-known architectures and their evolutions are examined first. The presentation of architecture models is divided into two sections: basic architecture models (section 2.1) and advanced models built based on the basic ones (section 2.2). Section 2.3 introduces our architecture model for agent-based architecture interactive systems.

2.1. Basic architecture models proposed in the literature

In general, the models proposed in the literature respect the principle of an explicit separation between the two parts of an interactive application: the interface that has direct contact with users and the application that is related to core functions. This separation is useful for developing and improving applications, allowing system designers to modify one part without affecting the other. As a result of this separation, the application's flexibility and maintainability are increased. In spite of this common point, the difference between models is clear.

We distinguish two main types of architecture models – functional and structural:

- **Functional models** – Such models decompose an interactive system into several independent functional components. Two well-known models of this type are Seeheim (Pfaff,1985) and ARCH (Bass et al., 1991). The Seeheim model splits an interactive system into three functional components: *Presentation* (interacting with the user), *Application Interface* (related to the functional core) and *Dialogue Controller* (intermediary between the previous two components). The ARCH model refines the relationship between three components of the Seeheim model by adding two additional components: *Functional core adaptor* and *Logical interaction*.
- **Structural models** – Such models provide decomposition, whose the granularity is much finer than the functional ones. They group functions together into one autonomous, cooperative entity, often called an agent. Some representative models of this type are PAC (Coutaz, 1987, 1990), MVC (Goldberg, 1983)] and AMF (Tarpin-Bernard and David, 1999). These models are agent-based architectures, respecting the principle of composition or communication, with no functional decomposition. For example, a PAC agent is made up of three facets: *Presentation*, *Abstraction* and *Control*, and an MVC agent is made up of three facets: *Model*, *View* and *Controller*. We can also mention here MVP (*Model-View-Presenter*), a specialization of MVC model (http://martinfowler.com/eaaDev/uiArchs.html).

Each type of architecture model has its own advantages and disadvantages. The functional models provide designers with an analysis strategy by breaking down a big system into different parts, but they also have some drawbacks. For example, the internal structure of components and the dialogue between them are not described, although this problem is completely managed by the designers. Another problem is that components of the functional models are too macroscopic. They provide canonical functional structures whose the granularity is too large and the functionalities are mixed in the too macroscopic components (Tarpin-Bernard and David, 1999). In general, functional models are useful as a structural framework for a rough analysis/design of an interactive system. They are generally not sufficiently fine to design complex applications, especially not industrial supervision systems, as in our case.

Unlike functional models, the structural models describe their entities (i.e., agents) and the communication between them. In addition, the granularity of their decomposition is much finer. These models thus seem to be more adaptable to complex interactive systems. Furthermore, breaking down an interactive system into several autonomous, cooperative entities can speed up the interactive system's feedback for the user. This advantage is very useful for industrial supervision systems because system users (i.e., human operators in control/command room, called regulators or supervisors) have to perform highly cognitive tasks and execute a set of operations to oversee and regulate the supervised dynamic process if it malfunctions (Moray, 1997). A long slow dialogue between supervision systems and their users risks slowing down the system, thus decreasing productivity.

In spite of these advantages, the structural architectures also have drawbacks. Unlike the functional models, the role and the number of the agents are not clearly specified (Coutaz, 1990). The global user interface of structural interactive systems can be difficult for designers to perceive because *Presentation* components are distributed on the various agents. Functional models solve this problem by supplying a single *Presentation* component.

2.2. Advanced models

Below we will present briefly some advanced models that derive from basic ones

2.2.1. Hybrid models

The advantages and disadvantages of functional and structural models led to hybrid architecture models, which try to exploit advantages of each of them. Several models combining functional and structural approaches have been proposed in the literature. For instance, Nigay and Coutaz combined the ARCH and PAC models to create the PAC-Amodeus model, in which the controller component is decomposed using PAC agents (Nigay and Coutaz, 1995). Guittet proposed the H^4 model, which decompose each component of the ARCH model into a hierarchy of the abstract objects (Guittet, 1995), especially, the hierarchical structure of the *Dialogue Controller* component are detailed by new concepts (token, questionnaire, interactor, task, monitor) in order to specify flows of information exchanged between the components of H^4 model.

2.2.2. Models dedicated to groupware

(Ellis, 1991) defines groupware as a computer system that assists a group of people engaged in a common task (or common goal) and which provides an interface to a shared environment. In fact, if collaborative human activities are supported by computer systems, then we have Computer Supported Cooperative Work (CSCW). Such systems are called groupware (Ellis, 1991).

Architecture models of groupware are the evolution from mono-user architectural ones. We can mention below some representatives ones: Dewan layered model (Dewan, 1998) considered as an extension/generalization of the Arch model; PAC* - a collaborative version of the PAC-Amodeus model, used for multi-users systems and especially CSCW (Calvary et al., 1997); Clover model (Laurillau, 2002) uses principles from Dewan and PAC* models for CSCW systems.

Groupware that can adapt to the settings of users and support them in their natural interactions is an active research topic in the modern age of ubiquitous networked devices, such as smartphones, personal digital assistants (PDAs), digital whiteboards, PCs, and tablets. Readers interested in their design and development can consult (Wolfe et al., 2010, 2009a, 2009b; Wu and Graham, 2007; Phillips et al., 2006; Phillips and Graham, 2003). We can also mention the work of (Garbay et al., 2012) who propose a multi-agent approach for collaborative support systems in distant tangible environments, and the works of (Lepreux et al., 2012) that enables interaction to take place between several interactive tabletops (seen as interactive systems), each usable by one or several people; this architecture enables two types of distribution between the user interfaces: centralized distribution of user interface (case in which one tabletop is master over the other) and network of distributed user interfaces.

2.2.3. Models for mixed-reality and mobile systems

Mixed reality aims at interactive systems in which real objects and computer data are mixed in a consistent manner. Paul Milgram proposed a unification of concepts using a "real – virtual continuum" from real-world to totally virtual

environments (Milgram, 1994a, 1994b, 1995). In this continuum, mixed reality is considered as intermediate stages that mix real and virtual objects and it is divided into two sub cases: augmented reality (AR) and augmented virtuality (VA) according to the proportion and role of real or virtual objects.

Architecture models for mixed-reality systems are usually created by combining a normal architecture model with an interaction model dedicated to mixed reality. We distinguish between the interaction model and the architectural one using the definition of (Beaudouin-Lafon, 2000):

- Interaction Models are a set of principles, rules and properties that guide the HCI design. They describe how to combine interaction techniques in significant and consistent manner and they define the "look and feel" of the interaction from the point of view of the user. For example the direct manipulation is a generic interaction.
- Architecture models describe the functional elements for the implementation of the interface as well as their relationships. A wide variety of implementation model are being presented in this paper.

We can mention here some interaction models dedicated for mixed-interaction modeling:

- Interaction model proposed by (Renevier, 2004a, 2004b). This model aims at designing collaborative mixed and mobile systems and it also allows representing the users and objects, their spatial relationships as well as the creation or destruction of objects.
- ASUR (Dubois, 2001, 2002c, 2003a) proposed by Dubois is a formalism used to describe entities involved in the use of a mixed-reality system in order to perform a given task. ASUR++ (Dubois, 2002a, 2002b, 2003b) is an ASUR extension for mobile and mixed systems. ASUR 2004 (Juras, 2004) is proposed by refining the S component of ASUR++, which represents the entire computer system. The latest version of ASUR and its meta-model are presented in (Gauffre and Dubois, 2011) and it provides a complete, explicit and standardized definition of ASURs.
- IRVO (Chalon, 2004a, 2004b) (Interacting with Real and Virtual Objects), an Interaction Model for Designing Collaborative Mixed Reality Systems. In fact, IRVO aims at “modeling the interaction between one or more users and the mixed-reality system by explicitly representing the involved objects and tools and their relationship”.

We can mention here some architecture models dedicated to mixed-reality systems:

- (Dubois, 2001b) adapts PAC-Amodeus to the mixed-reality systems by moving the interaction model ASUR closer to this architecture model.
- (Renevier, 2004b) presents another extension of PAS-Amodeus for mixed-reality collaborative and mobile systems, by: a) following the extensions proposed by Laurillau in the Clover architecture model in order to describe collaborative work aspects b) taking into account information about the users’ locality (their position and orientation).
- The agent-based architecture model for groupware AMF-C is extended by associating their agents to entities of the mixed interaction model IRVO in order to design collaborative and mixed reality systems (Chalon, 2004a, 2004b).
- (Dubois et al., 2011a) present ASUR-IL - a model used to describe the software architecture of mixed interactive systems. ASUR-IL is composed of two parts: 1) adapters (expressed by interaction model ASUR in order to describe the required devices and API to implement the link between the physical world and digital world) and 2) system-dependent components decomposed according to the MVC pattern (Model, View(s) and Controller(s) components)

Other papers for interested readers include (Dubois et al., 2011b) on a four-step co-design process of mixed interactive systems with an example of a mixed interactive system in a museum, (Bortolaso et al., 2011) on a method for using a mixed interaction model in creative sessions, and (Shaer et al. 2009) on a specification paradigm for designing and implementing Tangible User Interfaces. This paradigm is based on a high-level UIDL (User Interface Description Language) to provide developers from different disciplines with the means to specify, discuss and program a broad range of tangible user interfaces. This high-level description can be semi-automatically converted into programs concerning concrete TUI implementations, etc.

2.3. Proposal of new agent-based architecture models, an example

Existing hybrid architecture models still have drawbacks: they are still too imprecise to identify their agents, implementing them is tedious and difficult, and their application becomes a handicap for highly interactive interfaces (Depaulis, 2002).

In general, existing hybrid models do not take into account industrial interactive systems, such as supervision systems, where HCI is complex and highly interactive, and whose users (such as human control room operators) must supervise the real-time process and constantly perform quick and accurate manipulations. Our hybrid architecture model has already been proposed in this context and is intended for designing complex interactive systems in the industrial domain. Our model provides decomposition with fine granularity and its interface agents (presented below), (which can be added in arbitrary numbers by the system designers) can correspond to supervision or regulation functionalities.

This hybrid architecture model decomposes each functional component in an interactive system in a precise structural way. Its structure (Figure 1) has been presented in several papers (Grislin-Le Strugeon et al., 2001; Ezzedine, 2002; Ezzedine et al., 2005). We present it briefly below.

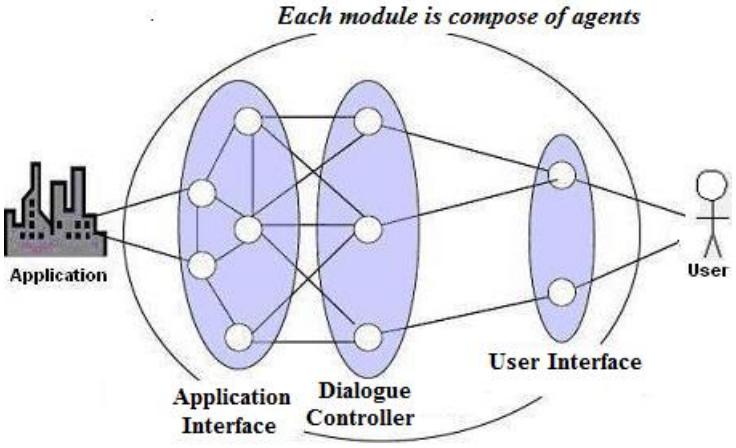

Figure 1. Structure of our hybrid agent-based architecture model

This architecture model is respectively composed of three functional components, which are the same as the Seeheim functional model: the component *Application Interface*, which connects directly to the application; the component *User Interface*, which enters in direct contact with the user; and the component *Dialogue Controller*, which is the intermediary between the two kinds of interfaces. Each component is organized structurally, each with a set of agents. We retain here the definition of agent, presented by (Grislin-Le Strugeon et al., 2001). According to it, an agent is considered as an information processing system with a set of actions it can perform, with mechanisms for input / output and the ability to represent the states (called state vector). These three components (*Application Interface*, *User Interface*, *Dialogue Controller*) can be considered as three sets of agents that are able to function in parallel, at least in theory.

- *Application Interface*: the *Application* represents the real process in physical world. For example, in our specific case of an industrial supervision system for urban transport network, the *Application* deals with the supervised vehicle network (i.e., bus, tram). The *Application Interface* component is a simulated multi-agent model of the *Application* that interacts directly with the *Application*. This component contains agents, called *application agents*, which can send information/commands to the *Application*. They can also receive information from the *Application* to transfer to the other agents in the same component or in other components. These agents manipulate domain-specific concepts and execute domain-specific functions for the *Application*. The *Application Interface* component, which cannot be directly accessed by the user, represents the functional core of the interactive system.

- *User Interface*: this component contains agents, called *interface agents*. Unlike the *application agents* in the *Application Interface* component, they can interact directly with the user, and are visible to the user, via UI events. UI events are represented by the acronym, EVIUs (EVents concerning Interface and User) in the meta-model of this architecture model, presented in the next section.

  These agents coordinate with each other in order to:

  - Intercept user commands and send them to the *Application* via the other agents (*application* and *controller* agents). In the supervision system, this allows human operators in the control/command room to send commands to the *Application*.
  - Compose a presentation that allows the user to understand the current situation of the *Application*. This allows human operators perform their tasks as supervisors of the *Application*.

  These *interface agents* are interactive agents or HCI agents. They may be simple buttons or windows whose intelligence is limited. However, they may also be intelligent interfaces. Such intelligent interfaces can be adaptable (with a possible human intervention), adaptive (without human intervention) to different use contexts (i.e., different users, physical environments, computing platforms and current activities) or execute tasks as a user assistant (Kolski and Le Strugeon, 1998). Such deep discussion about agent is beyond the scope of this paper.

- *Dialogue Controller*: this component contains *controller agents*, which are the intermediaries that insure the global coherence of the dialogue between *application agents* and *interface agents*. In particular, these agents create links between the other two components by distributing the user's commands to the *application agents* and transferring the feedback from the *Application* to the *interface agents*, and thus the user.

These three components constitute a complete model. However, an interactive system does not need to possess all of them. It is possible for an interactive system to be composed of only User Interface component and/or Application Interface component. These two cases correspond to an incomplete architecture model. Each agent provides a set of services that correspond to actions able to be executed by this agent. Each interaction between agents is realized by

service invocation between them (see section 4.2 below). Using this model, designers can add an unlimited number of interface agents to develop new representations for the user. This possibility is useful for developing complex HCI systems, such as industrial supervision systems. For example, we used this model to develop a supervision system for urban transport network, called IAS (see section 5).

The difficulty designers must face when they apply this architecture model to build an interactive system, is how can identify agents? In details, how to organize agents and distribute work and knowledge into agents in each component, especially in the case of large systems? This distribution affects many speed transmission of information between agents and it should be based on specified criteria. Indeed, in the context of complex supervision interactive systems whose users (as human operators in the control room) must supervise the process and make constant manipulation accurately and efficiently, the distribution of work and knowledge between interface agents can be based on the "natural" criterion. According to this criterion, each interface agent corresponds to a supervision or regulation functionality that operators can perform in reality. Identifying agents in this way can help operators understand more easily the HCI and perform tasks in better conditions.

Based on this architecture model, we propose a generic, reconfigurable evaluation environment, called EISEval, designed and developed to help evaluators analyze and evaluate certain aspects of interactive systems in general and of agent-based interactive systems that use this architecture model, in particular (see section 4). It aims to remedy drawbacks of traditional evaluation tools and make the evaluation of interactive systems in general and the evaluation of agent-based interactive systems that use our architecture model, in particular, more complete.

2.4. Concluding remarks on architecture models

In general, architecture models are useful for the system designers because these models guide their development of interactive systems. Indeed, such models, some of which are based on agent principles (Kolski et al., 2009), can serve as reference framework for designers. They provide designers with a generic structure in order to build interactive systems, but they are not sufficient to produce high quality interactive systems. In order to achieve this objective, evaluation is important because it allows problems and weak points in the evaluated systems to be detected.

3. Research Related to evaluation tools

For more than thirty years, interactive system evaluation has been a very active field of research. A user interface (UI) has to increase user comfort, satisfaction and productivity; it also has to decrease the possibility that users can make errors while they are interacting with the UI. Consequently, the evaluation is very important to help designers understand user difficulties when interacting with interactive systems and propose future improvements for HCI. According to Senach (Senach, 1990), all evaluation is based on a comparison of a model of the evaluated object and the reference model in order to draw conclusions about the quality of the evaluated object. Evaluators apply evaluation methods and tools to a given interactive system in order to obtain the real model of this system. Then, this model is compared to the reference model in order to help the evaluators detect weak points of the system and propose necessary modifications.

The evaluation is based on multiple criteria, but two global dimensions can be distinguished:

- *Utility* – Evaluating *utility* determines whether or not the UI allows users to achieve their objectives. It involves evaluating several properties: system performance, functional capacity, and the quality of the technical support (Nielsen, 1993).
- *Usability* – There is no standard definition for usability; instead, there are several definitions (Dix et al., 1993; Nielsen, 1993; ISO/IEC 9126-1; ISO 9241). In general, it refers to a set of many things – such as execution time, performance, user satisfaction and ease of learning (“learnability”), effectiveness, efficiency – taken together (Abran et al., 2003). The interested readers can find an overall usability engineering as well as a survey of usability tools in (Howarth et al., 2009), a summary of usability measures in (Hornbæk, 2006) and a framework for guiding and structuring, in a systematic way, activities concerning the usability problem assessment and reporting in (Andre, 2001). In hypermedia systems, we can mention another dimension: enjoyability. However, this dimension is also strongly related to the *usability* because the transparency and the friendliness of the user interface are the key issues in *enjoyability* (Yamada et al., 1995).

Nowadays, many evaluation tools exist. Before presenting our *EISEval* evaluation environment, we present below a brief state of the art of the existing evaluation tools. Among various types of evaluation tools, we are particularly interested in two important types: *tools for working with guidelines* (TFWWG) and *electronic informers* (EIs).

3.1. Tools for working with guidelines (TFWWG)

According to (Grammenos et al., 2000), the term, *guideline*, entails all forms of abstract or concrete recommendations that may be used to design or evaluate interactive software so as to produce a more efficient and user-friendly user interface. A guideline constitutes a design and/or evaluation principle for obtaining and/or guaranteeing an ergonomic user interface (Vanderdonckt, 1999). TFWWG can be tools either for accessing/retrieving guidelines or for evaluating user interface layout representations (Vanderdonckt and Farenc, 2000). TFWWG perform their evaluation using a guideline database, automatically or semi-automatically, after reading source codes or descriptions of user interface (knowledge-based tools).

Some representative tools should be mentioned: the guidelines management system proposed by Parush (Parush, 2000), SYNOP (Kolski and Millot, 1991), Sherlock (Grammenos et al., 2000), WebTango (Ivory and Hearst, 2002), DESTINE (Beirekdar, 2004; Jasselette et al., 2006), AWebHHT (Rukshan and Baravalle, 2011), ErgoCoIn (Morandini et al., 2011), EBC (Charfi et al., 2011), Ocawa (http://www.ocawa.com/fr/Accueil.htm), TAW (http://www.tawdis.net/), Dr. Watson (http://watson.addy.com), AChecker (http://achecker.ca/checker/index.php), HTML Toolbox (http://www.netmechanic.com/products/maintain.shtml), and WebXaACT (http://www.w3c.hu/talks/2006/wai_de/mate/watchfire.html).

Below is a brief presentation of some representative tools; interested readers can find details on these tools in relevant references.

- Guidelines management system proposed by Parush (Parush, 2000): Evaluators can enter new ergonomic rules, modify existing rules or lookup the system to know and study the rules necessary for carrying out their tasks instead of consulting a large paper manual. This is only an electronic system used to lookup rules instead of looking them up in a large paper manual; it does not support any UI evaluation.
- Sherlock (Grammenos et al., 2000) works with rules to evaluate WIMP interfaces (Windows, Icons, Menus, Pointing device). The description of a UI (in terms of a tree structure) is sent to Sherlock, which evaluates it based on the rules supplied by rule providers in terms of ActiveX DLLs (Dynamic Link Libraries). ActiveX DLLs also contain inspection routines to be called in order to evaluate a certain UI according to its rules and in order to detect usability violations of UI presentations, such as the incorrect position of a command button or the use of inconvenient colors.
- Three approaches, WebTango (Ivory and Hearst, 2002), AWebHHT (Rukshan and Baravalle, 2011) and DESTINE (Beirekdar, 2004; Jasselette et al., 2006) aim at evaluating web page UI usability (that relate to multiple aspects of Web pages, such as use of color, text on a page, links, fonts and images, etc.). However, each approach has its own way of representing guidelines. After representing guidelines, the evaluation tool of each approach analyzes the HTML source code of Web pages in order to evaluate them based on their guidelines' representation.

3.2. Electronic Informers

Figure 2 illustrates the principal actions of the EIs. These actions are performed in three steps:

(1) EIs discreetly and transparently capture data during interactions between the user and the UI in real use situations so that the user's activities are not interfered with. For example, the user's actions on the UI (e.g., click on a button, select a menu item) and the reaction from the UI (e.g., display/hide a window, make an alarm message appear/disappear) are captured.

(2) The captured data are stored in a database and then analyzed by the EIs. Analysis results can be different calculations (e.g., statistics), and they are often shown to evaluators in different forms (e.g., diagram, text, graph) to support their work.

(3) The model of user activity and HCI reactions can be reconstituted from the captured data and the evaluator's analysis results (Ezzedine and Abed, 1997). Called *observed model* or *real model*, this model can be compared with the model predicted and specified by the designers. Designers can use the results of this comparison to improve the interactive system. Several recent EIs have been proposed and are briefly presented below (interested readers can consult (Hilbert and Redmiles, 2000) for older EIs)**:**

- The tool family including USINE (Lecerof and Paterno, 1998), RemUSINE (Paterno and Ballardin, 2000), WebRemUSINE (Paganelli and Paternò, 2003), Multimodal WebRemUSINE (Paterno et al., 2006) and MultiDevice RemUSINE (Paterno et al., 2007). This tool family is based on using the task model CTT (Paterno et al., 1997) to support the UI evaluation. Analysis results of these tools are shown in textual or graphical form.

  USINE captures the physical actions of users on a WIMP (Windows, Icons, Mouse, and Pointers) interface and the related information (e.g., mouse coordinates, time, name and content of affected widgets,…). Then, using the CTT task model, it performs some analyses: task performance (successful or failed), types of errors committed by the user (e.g., useless actions) and statistical calculations (e.g., number of performed tasks, number of errors,…). RemUsine is similar, but it supports evaluations from a distance.

  WebRemUsine (WRU) captures three types of EVs (events) on a Web interface (user interaction EVs on a Web browser, internal EVs of the browser, and the EVs of the change of the target tasks of the user in the experiment) as well as additional information (e.g., date, IP address and name of client system,…). The performed analyses are similar to the USINE analyses and are supplemented with additional Web page information (such as the number of times a page is accessed, page visit patterns, and time taken to visit or download each page,…).

  Multimodal WebRemUSINE captures events similar to those captured by WRU, but also captures data concerning user gaze direction using an eye-tracker tool.

  Finally, MultiDevice RemUSINE captures events similar to those captured by WRU for mobile applications and also captures additional contextual information such as locality and network signal power, etc. The performed analyses are similar to those of other tools, except that they are combined with the contextual information.

- The IBOT (Zettlemoyer et al., 1999) tool captures two types of data: 1) information on screen content when the user interacts with the WIMP interface of the application and 2) low level EVs (mouse and keyboard EVs). IBOT combines these two types of information to determine corresponding EVs at the highest level. IBOT does not display analysis results. It inserts captured EVs into the event queue of the operating system to replay the actions made by the user on the interface, enabling the evaluator can see what the user has done.
- The WET (Etgen and Cantor, 1999) tool captures interactions between the user and the Web interface (e.g., mouse clicks, buttons pressed), Web browser EVs (e.g., page loads, etc.) and additional information (e.g., onscreen mouse coordinates, occurrence time, etc.). This tool cannot display analysis results because WET is simply a method for collecting EVs.
- The WebQuilt (Hong et al., 2001) tool captures data on visited Web pages, such as starting page, target page and page links clicked, but it cannot capture local interactions with Web page elements (such as button, etc.) on a client's machine. This tool can detect visited pages, frequencies of followed paths, navigation patterns, and discrepancies between the actual path and the optimal path expected by the designer. The analysis results are displayed as an interactive directed graph whose nodes represent visited pages and whose arrows between nodes represent transitions between pages.
- AppMonitor (Alexander et al., 2008): This is a Microsoft Windows-based client-side logging tool used to capture user actions in Windows applications. It performs analyses such as command use frequencies and behavioral patterns. This tool uses Windows SDK libraries to monitor both low-level interactions, such as "mouse button clicked" and "pressed keys", as well as high-level logical actions such as menu selection. The system currently supports logging in Microsoft Word and Adobe Reader, but it can be expanded to other applications according to the authors.

EIs are completely different from Quality Feedback Agents (QFAs). EIs capture interactions between users and the an application in real-use situations for later analysis in order to criticize the UI and improve it in the future. QFAs are software components used to only gather technical data about what is happening in the application whenever it crashes. This technical data is related to the context and the state of the application when it had problem (e.g., OS Version, Processor Type, Display Type, register, functions that were called on just before the failure). This data is sent to the development team to help them detect problems and the cause of the crash and then propose improvements for future versions of the application. QFAs may also allow users to report what they were doing with the application when the failure appeared. Since these QFAs only capture technical information to send to development team, QFAs are very limited for evaluating interactive applications, compared to EIs (Tran et al., 2008).

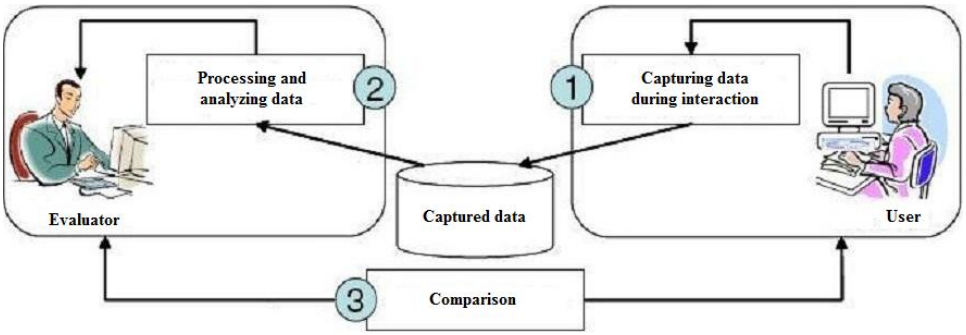

Figure 2. Principle of EIs

3.3. Conclusion

In general, the aforementioned tools focus on evaluating usability. The of these tools do not discuss utility because utility refers to the design's functionality and answers the question, "Does it do what users need it to do?" (Nielsen, 2003). As a result, the methods used to evaluate utility are generally surveys, expert reviews, etc. (according to http://wiki.cas.mcmaster.ca/index.php/Different_measures_for_evaluation).

Each type of tool evaluates a UI in different ways. TFWWG tools mainly try to evaluate the static aspects of user interface (e.g., position, size and color of elements, text fonts). In addition, the TFWWG evaluation is not based on objective use data. Since EIs are based on objective data captured from the interactions between users and the UI in real situations, EIs allow the user interfaces (including their dynamic behaviors) to be evaluated objectively for real situations. We designed EISEval as an extensive EI environment (not a TFWWG tool) based on objective data captured from the interactions between users and the UI as well as between agents themselves in real situations, so that EISEval can allow the user interfaces (including their dynamic behaviors) and other aspects of interactive systems to be evaluated objectively for real situations (see section 4). EISEval's activity is based on the EIs principles (Figure 2). However, EISEval also uses ergonomic criteria, as well as other criteria (for example, response time between agents, complexity of MAS design, etc.), to help evaluators interpret captured objective data and evaluate agent-based interactive systems. The shortcoming of traditional evaluation tools, used as the motivation of EISEval, will be presented in the next section.

4. EISEval: an environment for evaluating agent-based interactive systems

In this section, we present our evaluation environment, called EISEval (Environment for Interactive System Evaluation), which was specifically developed to evaluate interactive systems that use our agent-based architecture model (see section 2.3), although EISEval can also evaluate interactive systems based on other architecture models (as explained in the end of the section 4.2 below). This environment is organized in modules so that designers can modify a module without affecting the others.

This section is organized as follows. First, we give our motivation for developing EISEval and the shortcomings of traditional evaluation tools. Then, we present EISEval's design principles and briefly touch on the meta-model of our agent-based architecture model, which is the basis of EISEval. Finally, we introduce the multi-step process on which EISEval is based in order to evaluate interactive systems and introduce the modular structure of EISEval. Each module is also explained in this section.

4.1. EISEval's design principles and objectives

EISEval was designed to remedy the drawbacks of traditional evaluation tools in general and traditional EIs in particular. We highlight these drawbacks below:

- Most of traditional tools try to generically evaluate the UI of interactive systems, but we needed an evaluation tool that can specifically evaluate interactive systems that use our agent-based architecture model. In other words, most of traditional tools do not take architectural specificities of agent-based interactive systems into account when evaluating them.
- Traditional EIs only evaluate the UI of interactive systems. They do not consider other aspects of interactive systems, such as the non-functional properties of an agent-based interactive systems (e.g., agent response times, reliability, design complexity), although these properties can be very useful for evaluating the quality of a multi-agent system in general and an agent-based interactive system in particular (Lee and Hwang, 2004).
- After capturing HCI data, the existing EIs perform some analysis on captured data and show the analysis results (e.g., statistics, classification) in visual forms to evaluators, who then must interpret these results to identify the problems with the UI and suggest improvements to designers. There is no assistance or indications to help the evaluators do their work.

In order to remedy these drawbacks, and thus extend the possibilities of traditional EI evaluation and make the evaluation of agent-based interactive systems in particular and of interactive systems in general, become more complete, we determine certain design principles (see below) that EISEval must respect before constructing it:

- EISEval is essentially an extensive EI environment, and the EISEval activities are based on EI principles (see Figure 2). However, compared to traditional EIs, EISEval provides a more complete evaluation. Traditional EIs evaluate only one aspect of interactive systems: the UI. EISEval must allow evaluators to evaluate the various aspects of interactive systems: the UI, but also certain non-functional properties of interactive systems (e.g., agent response times, reliability, design complexity) and certain user characteristics (e.g., preferences, habits, ability to use the system, user comparisons).
- EISEval exploits the specificities of our agent-based architecture model to evaluate interactive systems using this model, which allows evaluators to better detect the problems in agent-based interactive systems, as well related elements, for example: agent services that function poorly (e.g., their execution failed or took a long time); interface agents that often or rarely interact with the user; application agents that often or rarely have problems) By detecting such problems, evaluators may have an easier job when determining the necessary improvements and then proposing them to designers.
- Although EISEval was specifically designed to evaluate interactive systems that use our agent-based architecture model, it must be able to evaluate interactive systems that use other architecture models. We will explain how to do it in the end of the next section 4.2.
- EISEval must extend traditional EIs by helping evaluators interpret analysis results to evaluate the different aspects of interactive systems.[1]
- EISEval must be generic and reconfigurable. As a result, EISEval was designed to be independent of a particular interactive system, and it can be reconfigured to evaluate different systems.
- EISEval must take the abstraction levels of events into account. Two abstraction levels are taken into account: EVIU level as well as service level (same level) and task level (higher level). These two levels are presented below in the meta-model of our agent-based architecture model.

[1] Although the current version of EISEval provides evaluators with the indications necessary to help them interpret the analysis results, one of the possibilities for future research is to extend this interpretation (see section 7).

4.2. Meta-model of our agent-based architecture model

In the section 2.3, we have presented the structure of our agent-based architecture model. In this section, we must present the meta-model of this architecture model. Indeed, EISEval mainly aims at evaluating interactive systems that are based on this architecture model. As a result, in order to propose EISEval - a generic, reconfigurable environment to help mainly evaluate agent-based interactive systems that use our architecture model, it is necessary to "*explain*" this architecture model to EISEval. In other words, EISEval must "*understand*" our architecture model, which is required for interactive systems in order to be evaluated by EISEval. For this reason, we propose a meta-model of our architecture model. This meta-model describes dynamic or behavioral aspect of our architecture model whereas the structure (presented in the section 2 above) describes its static or structural aspect.

Please note that although EISEval mainly aims at evaluating interactive systems that are based on our architecture model, it can still evaluate other interactive systems that are not based on it. This is one of our design principles (mentioned above) and we will explain how to use EISEval to evaluate interactive systems that do not use our architecture model in the end of this section.

Figure 3 shows the meta-model that we used in this paper. We used UML class diagram in order to present this meta-model briefly and visually. The design of the EISEval evaluation environment is based on this meta-model. Please note that these UML diagram classes in the meta-model do not imply or obligate their real implementations in the source code of evaluated interactive applications based on our architecture model. Indeed, these classes are only used in this paper to describe and help readers understand better, in a visual way, entities in our architecture model as their behaviors (in terms of interactions between them).

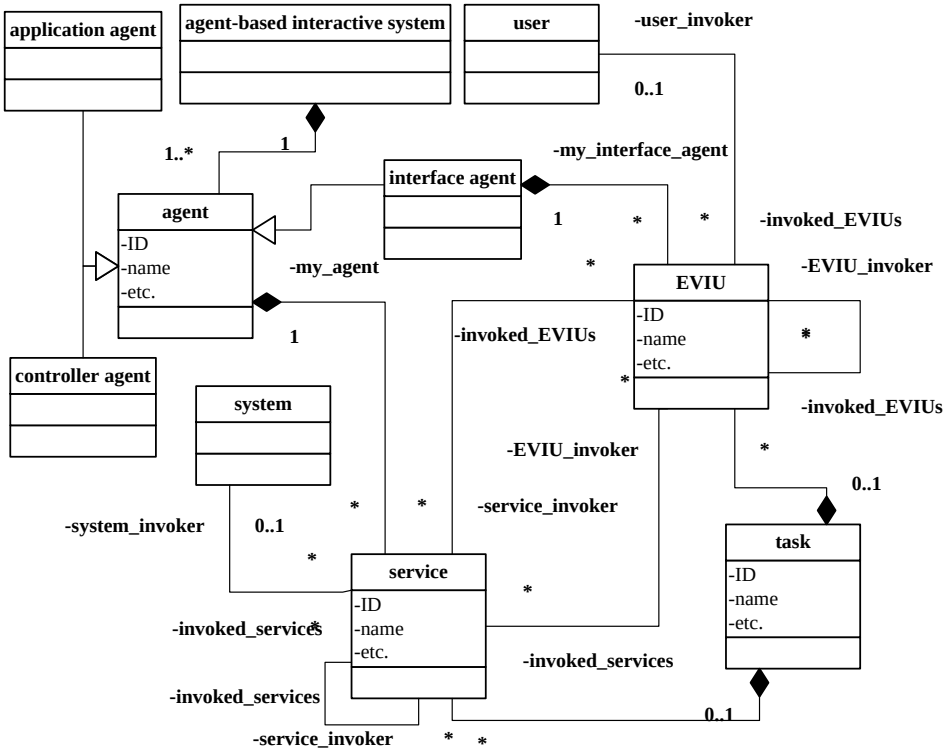

Figure 3. Meta-model of our proposed agent-based architecture model

We explain our meta-model briefly below:

- An interactive system that uses our architecture model is composed of agents. According to what we have presented in the section 2.3, an agent (class *agent* shown in Figure 3) is one of three types: *application agent*, *interface agent*, or *controller agent* (respective classes shown in Figure 3).
- Every agent (*application, interface*, or *controller agent*) manages a set of services. A *service* (class *service*) is defined as an action that can be executed by an agent.
- Every *interface agent* is associated a set of UI events, abbreviated to EVIUs in our description (EVents concerning Interface and User, class *EVIU*). Such events represent interactions between the *user* and *interface agents*.
- The interactions between *interface agents* and the user appear in terms of EVIUs, whereas interactions between agents themselves appear in terms of service invocations. Such interactions are illustrated in Figure 3.

An EVIU of an associated *interface agent* only appears if it is triggered by one of three following objects:

- The **user** – An EVIU of a certain *interface agent* can be triggered by the user through an interaction device, for example, a menu item or a screen button is selected when the user clicks the mouse button on it.
- A **service** of the same *interface agent* – For example, in an urban transport network's supervision system, when a traffic disturbance is detected, the service related to a traffic disturbance warning can show a window whose message warns supervisors of this disturbance (EVIU corresponds to the display of this window).
- Another **EVIU** – An EVIU of a given *interface agent* can be triggered by another EVIU of the same *interface agent*, for example, a window is closed when the user clicks on its *Cancel* button.

A service is only executed by an associated *agent* if it is triggered by one of three following objects:

- The **system** (class *system* shown in Figure 3) – A service of a given *interface agent* or an *application agent* can be invoked by the system, and this service is automatically executed. For example, in an urban transport network's supervision system, the service related to detecting traffic disturbances can be automatically executed when a traffic disturbance occurs (e.g., lateness or breakdown of a vehicle). Nonetheless, the services of *controller agents* cannot be invoked by the system.
- **Another service** – A service can be invoked by another service of the same agent or another agent. For example, in an urban transport network's supervision system, the service related to detecting traffic disturbances can invoke the service related to traffic disturbance warnings. Such service invocations constitute the dialogue between system agents.
- An **EVIU** – A service of an *interface agent* can be triggered by an EVIU of the same *interface agent*. For example, a click on a button can invoke the execution of a certain business function. This case happens when the user activates a certain function of the interactive system to perform his/her domain task.

Figure 4 provides an illustration of the activity of a service using a Petri Net (PN). From the current state, if a triggering event appears (system, another service or an EVIU), conditions are verified, necessary resources are available, and the service will be executed. This execution can invoke another service or an EVIU. PNs will be presented in the section 4.4.5.

A task (class *task* shown in Figure 3) is an abstract event, whereas EVIUs and services are objective data captured by EISEval. The task is situated at an abstraction level that is higher than the level of services and EVIUs. The task represents what the user and/or system must execute to accomplish some business purpose. Tasks are specific to each application domain and they usually correspond to the system's functionalities. In order to realize a task, a set of EVIUs can appears and/or services can be executed. Based on the event that initializes tasks, the model can have two types of tasks:

- *System task* – A system task is initialized by a service of an *interface agent* or of *application agent*. Then, this service can invoke another service or EVIU. There are no direct user interventions to realize this task. For example, in an urban transport network's supervision system, the task "*Warn supervisors of traffic disturbances*" is carried out by two services and one EVIU. First, this task is initialized by the service "*Detect traffic disturbances*" (service 1). Then, this first service invokes a second service "*Warn of traffic disturbances*" (service 2). This second service displays a window to warn supervisors of this disturbance (EVIU 1). In an industrial supervision system, *system tasks* are often executed when (1) the supervision system informs human operators of the current state of the *Application* (i.e., the real process under supervision) so that they can carry out their supervisory tasks, and (2) the supervision system warns human operators of a disturbance or a problem in the *Application* so that they can make regulations necessary. Such adjustments are themselves *user tasks*, presented below.
- *User Task* – Unlike the *system task*, a *user task* is initialized by an EVIU or a series of EVIUs. These EVIUs are triggered by the user, and they can invoke services or other EVIUs to accomplish this task. For example, in an urban transport network's supervision system, the task "*Send a message to passengers at station"* is carried out by a series of three EVIUs, triggered by the user in succession: *ImageStation_Click* (Clicking on the station's image on the screen), *TextBoxMessage_Changed* (Typing message content), and *buttonOK_Click* (Clicking on the button OK to send the message). The EVIU *buttonOK_Click* will invoke the associated service to send the message to passengers at the selected station. In a supervision system, user tasks often correspond to human operator activities in order to regulate the supervised process. A supervision system always provides human operators with necessary functions for such regulations.

An EVIU of an *interface agent* can only invoke a service or another EVIU of the same *interface agent*; it is not allowed to invoke a service or an EVIU of another *interface agent*. As shown below, this constraint can be described in more structural way using OCL (Object Constraint Language)[2] :

```
context EVIU

inv: if self.invoked_EVIUs->size() >= 1
then
    self.invoked_EVIUs->forAll(invoked_EVIU | invoked_EVIU.my_interface_agent = self.my_interface_agent)
endif
```

[2] A more detailed explanation of OCL can be found on the OMG website: http://www.omg.org/technology/documents/formal/ocl.htm

inv : if self.invoked_services->size() >= 1
then
self.invoked_services->forAll(invoked_service | invoked_service.my_ agent = self.my_interface_agent)
endif

Apart from making EISEval "understand" the evaluated interactive system, this meta-model and its OCL description are still useful if we aim at developing a visual development environment to help developers design, in an interactive and visual way, interactive systems that use our architecture model. The meta-model is the base to construct this development environment and OCL can be used by this environment to verify whether developer's design for a certain interactive system (based on our architecture model) is valid or not. This environment is one of our future research topics (section 7).

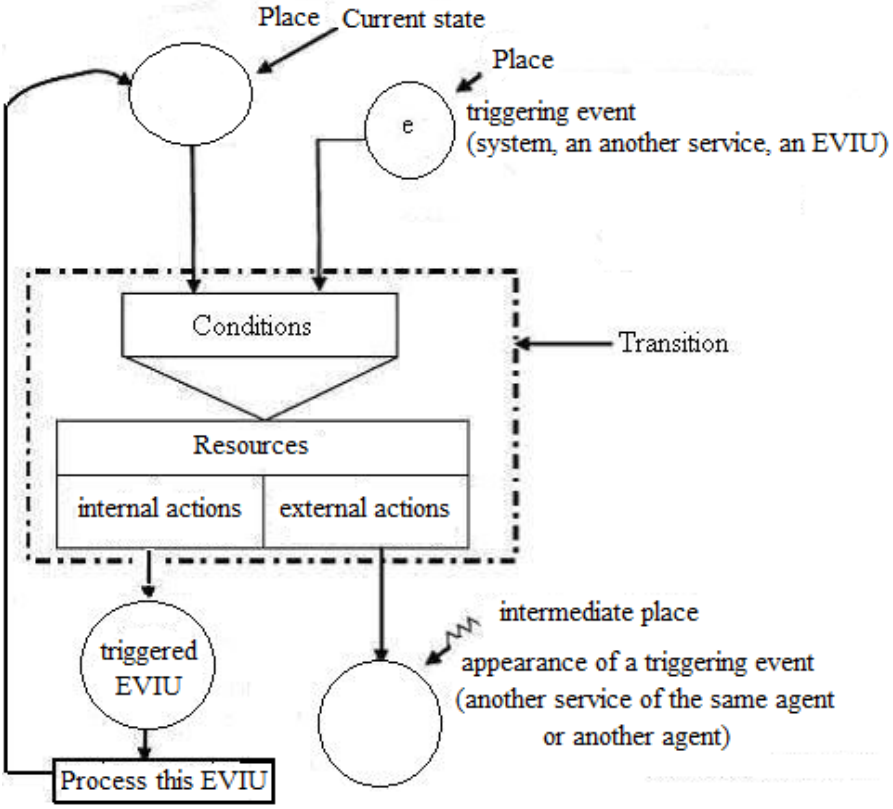

Figure 4. Activity of a service

This meta-model is the foundation for our proposed environment for evaluating agent-based interactive systems that use our architecture model. Indeed, EISEval mainly aims at evaluating interactive systems that are based on our architecture model and it takes into specificities of this architecture model into account. The meta-model of our agent-based architecture model (presented above) is the way EISEval "*understand*" the evaluated interactive system. In other word, EISEval consider that the evaluated system is based on our architecture model whose static (structural) and dynamic (behavioral) aspects are already presented above. However, EISEval must be able to evaluate interactive systems that do not use our agent-based architecture model. This is one of our design principles (mentioned above). In order to use EISEval for evaluating a certain interactive system, called system A that does not use our agent-based architecture model, we do in the following way:

We consider this interactive system A as a special case of our agent-based architecture model where A is composed of a single *User Interface* component – only one of three components of our architecture model. In other words, it can be considered that this interactive system A is based on an incomplete architecture model of our architecture model that we have mentioned in the section 2.3 above. The *User Interface* component of A also contains an only one large *interface agent*, which corresponds to the whole system.

As a result, we can use EISEval to evaluate this interactive system A in the same way we use EISEval to evaluate interactive systems that are based on our architecture model. EVIUs related to HCI interactions (between the user and the *interface agent* of A only) are captured and later analyzed by the EISEval's modules. However, if the evaluated interactive system is not based on our architecture model, EISEval is similar to traditional EIs and there are no specificities concerning our architecture model to be captured and analyzed (e.g., Interactions between agents). However EISEval is still extended by some specific and additional functions to analyze captured data. These functions are provided by EISEval' modules that will be presented later, step-by-step.

4.3. Evaluation process of EISEval

Applying EISEval to use an interactive system follows a multi-step process as presented below:

Step 1: EISEval captures and stores objective data. These data can be EVIUs (interactions between the user and UI (*interface agents*)) and/or interactions between the agents themselves in terms of their service invocations.

Step 2: EISEval must be reconfigured to evaluate a given interactive system. Reconfiguration means that some input information about the specific configuration and settings of the evaluated interactive system must be provided for EISEval so that they can be stored and used in remaining steps. Input information can be provided for EISEval by the evaluator via some user interfaces or via a configuration description file containing such necessary input information. In fact, EISEval implementation does not impose rigorous order of the step 1 and 2 and allows them to be able performed at the same time or in any order because their implementations are in two independent modules (see section below).

Step 3: Analyze captured data and show analysis results in visual forms (determining tasks from captured data, statistics, measure calculations, PN generation, etc.). This step requires a certain intervention from the evaluator (presented later). The input of this step is the data captured by step 1 as well as configuration information provided by step 2. The output of this module is the analysis results of the target system that will be the input of the step 4 below.

Step 4: Help evaluator interpret analysis results in order to evaluate aspects of target interactive system based on a list of predetermined criteria (e.g., ergonomic criteria or quality attributes). This is an open list and EISEval allows the evaluator to modify/add criteria to it for each evaluated interactive system because each interactive system can require specific criteria. The evaluation results of an interactive system can be saved for ulterior exploitation. The input of this step is the step 3's output (analysis results) as well as such a predetermined and open list of criteria (e.g., ergonomic criteria or quality attributes). The output of this module is the evaluation results of the target system based on these criteria (presented later in the sections 4.4.6 & 6 below). This step will be more detailed later in the section 5 through our study.

EISEval is designed in modular manner and these steps are implemented by EISEval's modules. We present them in the section below.

4.4. Structure of EISEval

This section presents our EISEval evaluation environment. The Figure 5 depicts seven modules of EISEval that we'll explain each of them below.

Although this environment is still able to evaluate interactive systems that use other architecture models, EISEval was specifically designed to evaluate interactive systems that use our agent-based architecture model (explained in the end of the section 4.2 above). EISEval is composed of seven modules, each of which belongs to a step of the EISEval's evaluation process (presented above). Indeed, the motivation for this module breakdown of EISEval is mainly based on these steps. Each module will be presented below. In this section, we use screenshots to illustrate the activities of these modules. These screenshots show data from our study (see section 5). The first version of EISEval was designed and developed in C++.

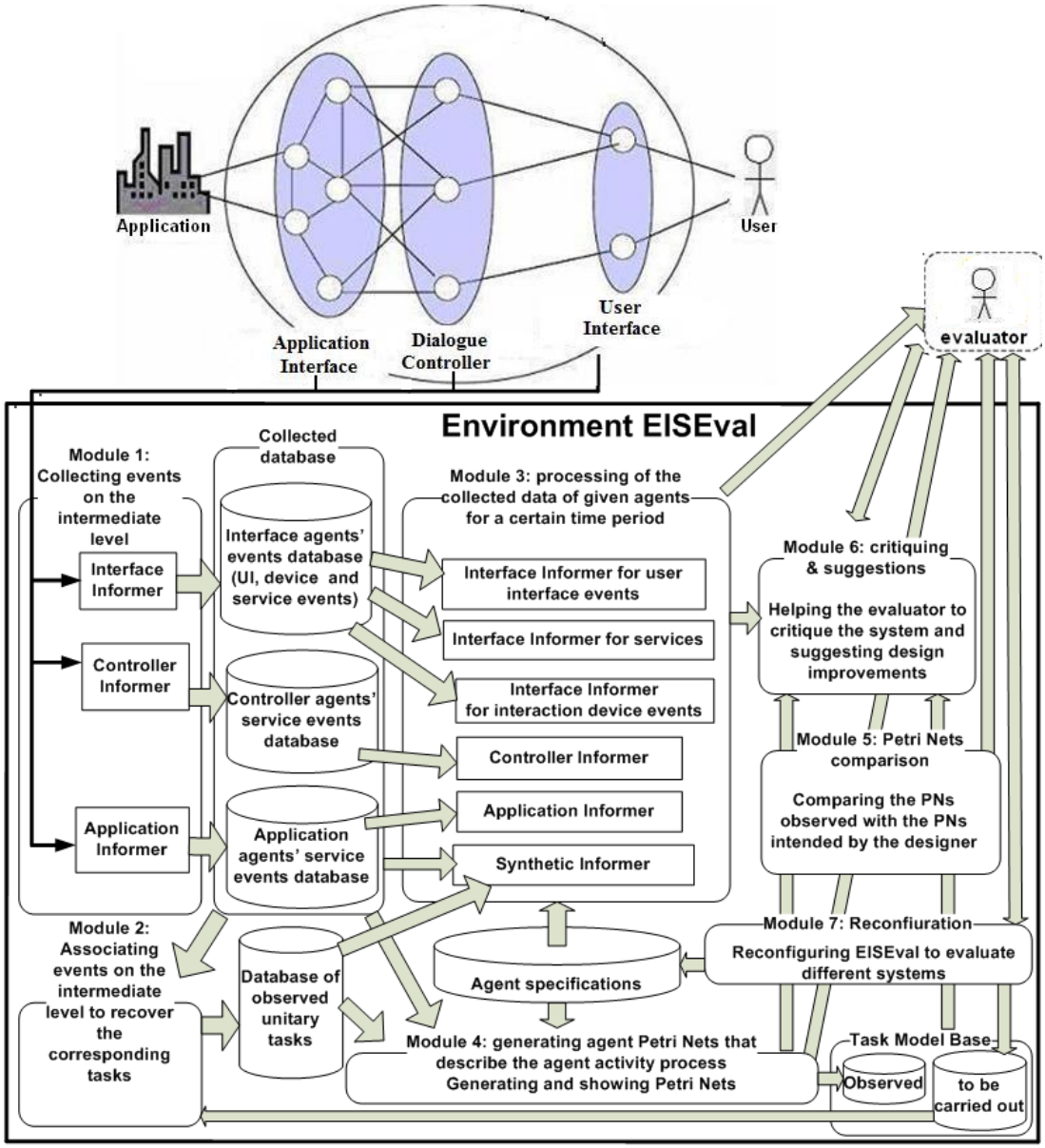

Figure 5. The EISEval seven modules

4.4.1. Module 1 (M1)

This module corresponds to the step 1 of the EISEval's evaluation process in order to capture and store objective data. Among the seven modules of EISEval, only Module 1 (M1) was developed as an individual system; the six remaining modules were integrated in a single system. The M1 and this single system can be launched individually. The M1 captures objective data that later will be analyzed by other modules of EISEval. These data involve not only HCI events (like traditional EIs) but also agent interactions. Specifically, these data can be 1) events related to interactions between the user and *interface agents* (i.e., EVIUs), and 2) execution-invocations between agents' services.

As Figure 5 shows, M1 is composed of three informers, which capture data (i.e., EVIUs, services) from respectively *interface agents*, *controller agents* and *application agents* of the interactive evaluated system, and then store these captured data in databases; the other EISEval modules can retrieve data from these databases to analyze them. When evaluators want to use EISEval to evaluate a given interactive system, M1 must be launched first, before launching the system, so that M1 can capture and store data from the evaluated system. This capture can be even performed remotely since M1 can be launched individually to capture data from the evaluated interactive system, which can run on the same machine or another one.

There are two ways to plug module 1 into the evaluated system for gathering events:

(1) M1 functions like a server and the evaluated system, as a client, connects to the M1 and sends events to it

(2) M1 receives and processes event logs from the target system.

Both ways can be performed using instrumentation code, which means a small quantity of instrumentation code is inserted into the evaluated system to output necessary data for capture. These data are sent to the M1 (which functions as a server) via a socket mechanism or are saved in logs to be processed later.

There are specific logging tools to save logs for the second method. However, we currently use the instrumentation code by inserting a very small quantity of code into the evaluated system. There are also two instrumentation approaches: source instrumentation or binary instrumentation. In this first version of EISEval, we follow source instrumentation. Instrumentation is an interesting topic, but such a discussion about it is beyond this paper. Right now, we are focusing more on analyzing captured data and interpreting analysis results to evaluate interactive systems. However, we have several ideas for improving the M1 that is in charge of capturing data. These ideas will be presented in the last section of this paper as future research.

As mentioned above (the end of section 4.2), in the case where we use EISEval to evaluate a certain interactive system, called system A which is not based on our agent-based architecture model, we consider this interactive system A as a special case of our agent-based architecture model where A is composed of a single *User Interface* component – only one of three components of our architecture model. This *User Interface* component contains an only one large *interface agent* which corresponds to the whole system and we can use EISEval to evaluate this interactive system A in the same way we use EISEval to evaluate interactive systems that are based on our architecture model.

As a result, M1 will captures and stores EVIUs related to the HCI interactions (between the user and the only interface agent of A), which later be analyzed by the other modules. In this case (where EISEval is used to evaluate interactive systems that are not based on our architecture model), the M1's data capture task is similar to traditional EIs and there are not any specificities concerning our architecture model to be captured and later analyzed by other modules (ex. Interactions between agents). However, other EISEval modules still provide evaluators with some specific and additional functions (compared to traditional EIs) to analyze captured data (presented below).

4.4.2. Module 7 (M7)

Module 7 (M7) corresponds to the step 2 of the EISEval's evaluation process and it allows evaluators to reconfigure EISEval to evaluate different interactive systems. EISEval reconfiguration means that some input information about specific configuration and settings of the evaluated interactive system must be provided for the module 7 so that they can be stored in a database and used by remaining modules (2, 3, 4 & 5). Among this input information of the module 7, some are mandatory and others are optional. This input information will be made clear later, step by step, in the sections concerning remaining modules (2, 3, 4 & 5). In this section, we only mention briefly it:

- Mandatory Information:
  - Information about *tasks* (*user tasks* and/or *system tasks*. See section 4.2 above for the *task* notion) that can be performed by the evaluated system. Indeed, the task represents what the user and/or system intend to execute in order to accomplish some business purpose and they usually correspond to the system's functionalities. When developers design and implement an interactive system, these functionalities (tasks) must be determined and developed. For example: when developers design and implement an urban transport network's supervision system, they determine and develop following tasks (functionalities): "*Warn supervisors of traffic disturbances*", "*Send a message to passengers at station*", etc. because they predict that these task can be possibly to be carried out in reality. As a results, these tasks are called *predicted tasks* or *theoretical task*s or "*to be carried out*" tasks. Predicted/theoretical tasks are used to confront with the notion *real tasks* or observed tasks that have already been carried out by the user in a real situation (see the next section for clearer explication). In order to evaluate this system, the information about these tasks must be provided for the module 7.The provided information about each *predicted task* involves its title, description and settings (ex. time predicted to realize this task of an average user in reality and time predicted to realize this task of an expert user in reality. See Figure 6 below). In the Figure 5 above, we can see the module 7 stores this information about *predicted tasks* to the repository (called *"to be carried out"*) of the *task model base*.
  - Information about agents of the evaluated systems (indeed, when developers design and implement an interactive system, its agents must be determined and developed) as well as other configuration settings (ex. predicted average response time of service invocations between agents).
- Optional information: Detailed information on each agent, ex. information on associated services, EVIUs, etc. In the Figure 5. We can see the module 7 stores information about agents (mandatory and/or optional one) to the *AS (agent specifications)* repository.

This input information will be gradually illustrated in the next sections on the remaining modules. However, how can it be provided for module 7? Indeed, this input information can be provided for the module 7 by the evaluator via some user interfaces (one of which is illustrated by the Figure 6 below) or via a description file containing all necessary input information.

In the end of the section 4.2 above, we mentioned one of our future research topics: constructing a visual development environment that helps developers interactively and visually design interactive systems that use our architecture model. Indeed, after designing an interactive system in this environment, one of functionalities of this environment is to generate a description file that contains such necessary input information to be provided for the module 7. At this moment, the evaluator still has to input the module 7 via its user interfaces. This way takes the evaluator a significant time.

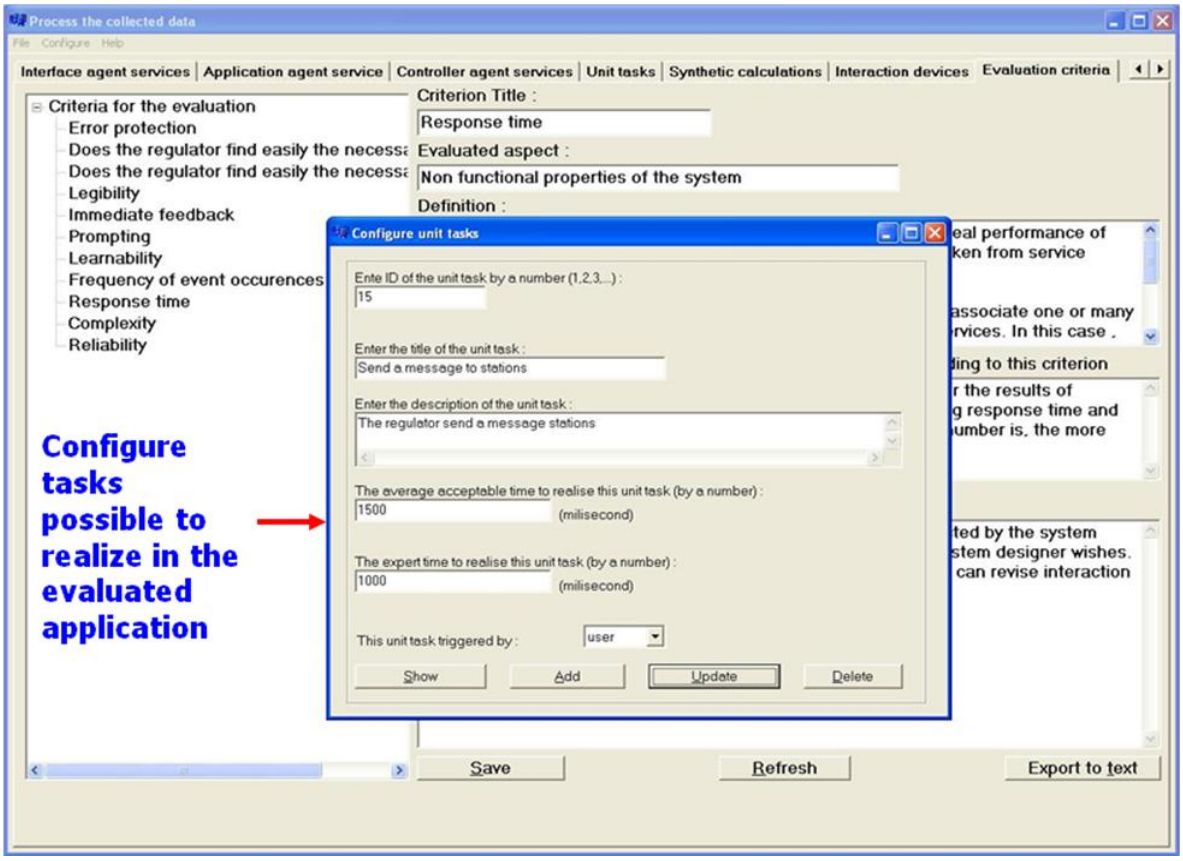

Figure 6. Screenshot of one of user interfaces of the module 7 – information about a predicted task ("tasks possible to realize" in this figure) provided for the module 7

4.4.3. Module 2 (M2)

For each interactive system, designers specify tasks that can be possibly carried out by the user/system. These tasks are called *predicted tasks* or *theoretical task*s (presented above). For example, in an urban transport network's supervision system, a *theoretical task* can be "*Warn supervisors of traffic disturbances*" or "*Send a message to passengers at station*". Before using the EISEval modules 2, 3, 4 & 5 to analyze data captured by M1, the information about the configuration of the evaluated system must be inputted into module 7 (see section 4.4.2).

Module 2 (M2) belongs to the step 3 of the EISEval's evaluation process and it allows evaluators to determine, among *theoretical tasks*, what tasks have already been carried out by the user/system in a real situation; these tasks are called *real tasks* or *observed tasks*. Indeed, in order to determine *observed tasks*, evaluators have to associate each *theoretical task* to corresponding EVIUs and services (captured by M1). All these associations are stored in the database of *observed task* (Figure 5). For example, evaluators can associate the series of three EVIUs – *ImageStation_Click* (Clicking on the station image on the screen), *TextBoxMessage_Changed* (typing message content in the text box), and *buttonOK_Click* (Clicking on the button OK to send the message) – to the *theoretical task,* "Send a message to passengers at station". Using this association, evaluators determine that the user has already carried out this task, "Send a message to passengers at station" in a real situation. M2 is used in EISEval to provide some kinds of analysis, such as identifying *theoretical tasks* that have always, or never, been carried out in reality; computing statistics for task executions; or comparing the user/system's task executions (in real situation) with the designer prediction, the *theoretical task*. This is the only intervention from the evaluator in this step 3 of the EISEval's evaluation process and we can see (in the Figure 5 above) this module 2 stores determined observed tasks to a separate database for these tasks.

4.4.4. Module 3 (M3)

Module 3 (M3) belongs to the step 3 of the EISEval's evaluation process and it retrieves the data captured by M1 (i.e., EVIUs, services) and the *real tasks* determined by M2 in order to analyze them. These analyses involve statistics (e.g., number and frequency of EVIUs, executed services, observed tasks) and measure calculations (e.g., the time taken to carry out each observed task or each service in real situation, the average response time between agent services, average time taken to accomplish a task, number of successful or failed tasks, etc.) of a given agent or all agents in a given time period. Readers can find that some measures (provided by this module 3) have already been presented in (Hornbæk, 2006). However, other measures, such as system-related measures (time response, service measures, etc.) are not included in it. Moreover, M3 not only compute measures but also compare, in quite detailed way, the measures computed from real usage with measures predicted by the system designer. In brief, M3 provide analysis results concerning statistics/measures (number or frequency of EVIUs/services/tasks; number or frequency of successful

realization of each service/task as well as all services/tasks, response time, completion time on each service and observed task, etc.) and compares these analysis results to the ones predicted by the system designer. M3 can calculate such statistics & measures for each agent or across agents.

The M3 analysis results are shown to evaluators in visual forms, such as tables and graphs. Based on the suggestions and indications of module 6 (see section 4.4.6 below), the evaluators are supported to be able to interpret analysis results of modules 3, 4 & 5 to criticize the system and propose improvements to designers. M3 generates a set of screenshots. These analysis results shown in the screenshots in Figures 7, 8 & 9 come from our study (see section 5). These screenshots show the M3's analysis results for a human subject who participated in this study.

Figure 7 shows the number and frequency of EVIUs that have occurred with a given interface agent of the evaluated system. M3 can also produce such statistics and measures about realized tasks, executed services and triggered EVIUs and then, show the results in tables or graphs. This Figure is one of M3's screenshots concerning EVIUs. These analysis results are very useful for evaluators to assess UI layout and system design. For example, knowing the frequency of EVIUs allows the evaluators to identify the UI's interactive elements that usually/rarely/never interact with users, so that the evaluators can propose improvements for UI layout.

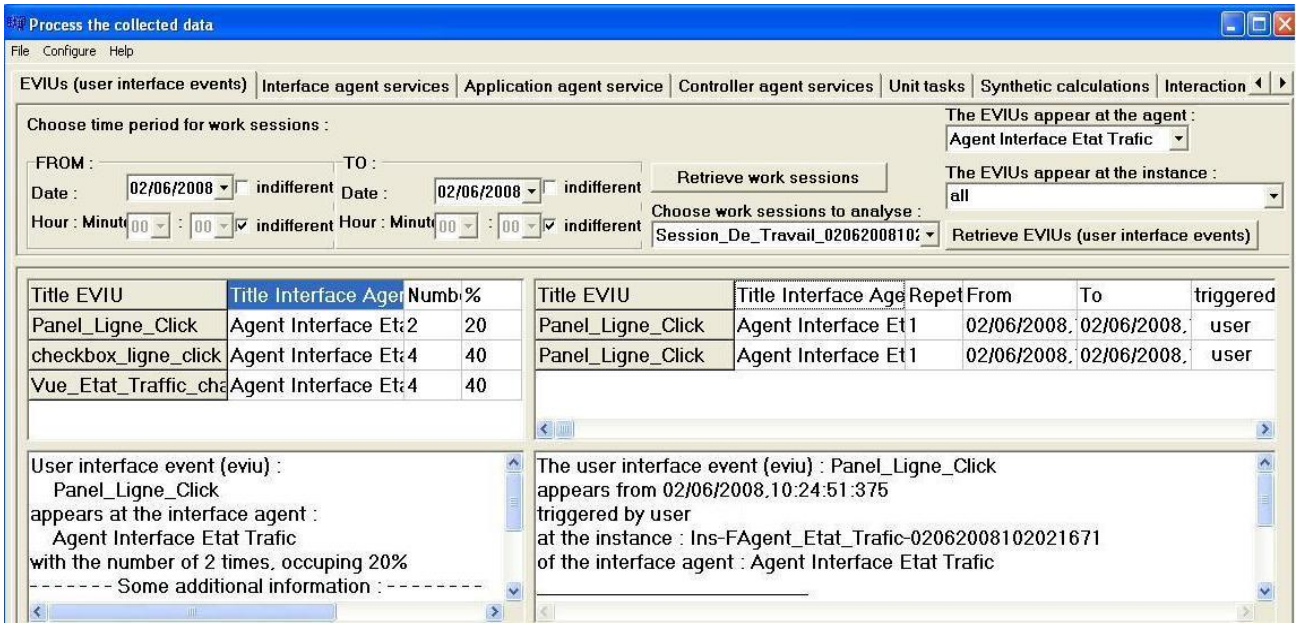

Figure 7. One of screenshots of module 3 - The number and frequency of EVIUs occurring with a given interface agent, shown in the form of a table. By clicking on each "tab" (EVIUs, services, tasks, etc.), the evaluator can see corresponding analysis results (statistics, measures) relative to EVIUs, services, tasks, etc

Figure 8 shows the number of successful or failed tasks represented with a bar graph. M3 can also produce such statistics about services (e.g., the number of successes or failures of a given service or all services). These analysis results are useful for evaluators for assessing system reliability and global system functioning.

Figure 9 shows the additional calculations that M3 can provide. These calculations relate to some measures, for example, the average response time of interactions between agent services, the number of service interactions, the average time taken to carry out a task, the number of successful or failed tasks, the number of successful or failed services, and/or the number of tasks carried out. These calculations are performed at the system level, so we call them aggregation calculations. These calculations are very useful for assessing non-functional system properties, such as speed (through the response times between services' interactions), reliability or design complexity.

Figure 8. One of screenshots of the module 3 - Statistics for results of observed tasks – number or frequency of successful realization of tasks - shown in the form of a graph

Some calculations related to tasks can be found in other EIs, such as WebRemUSINE (Paganelli and Paternò, 2003), but other calculations related to services and response time and comparisons are not possible with these EIs. Indeed, M3 can compare real information with information predicted by system designers. For example, designers can predict the time taken by an expert user and an average user, respectively, to execute each task. M3 can compare the time taken in real situation with such expert time and average time predicted by designers, and then calculates the number of tasks whose execution time is longer, shorter or equal to the expert and average times.

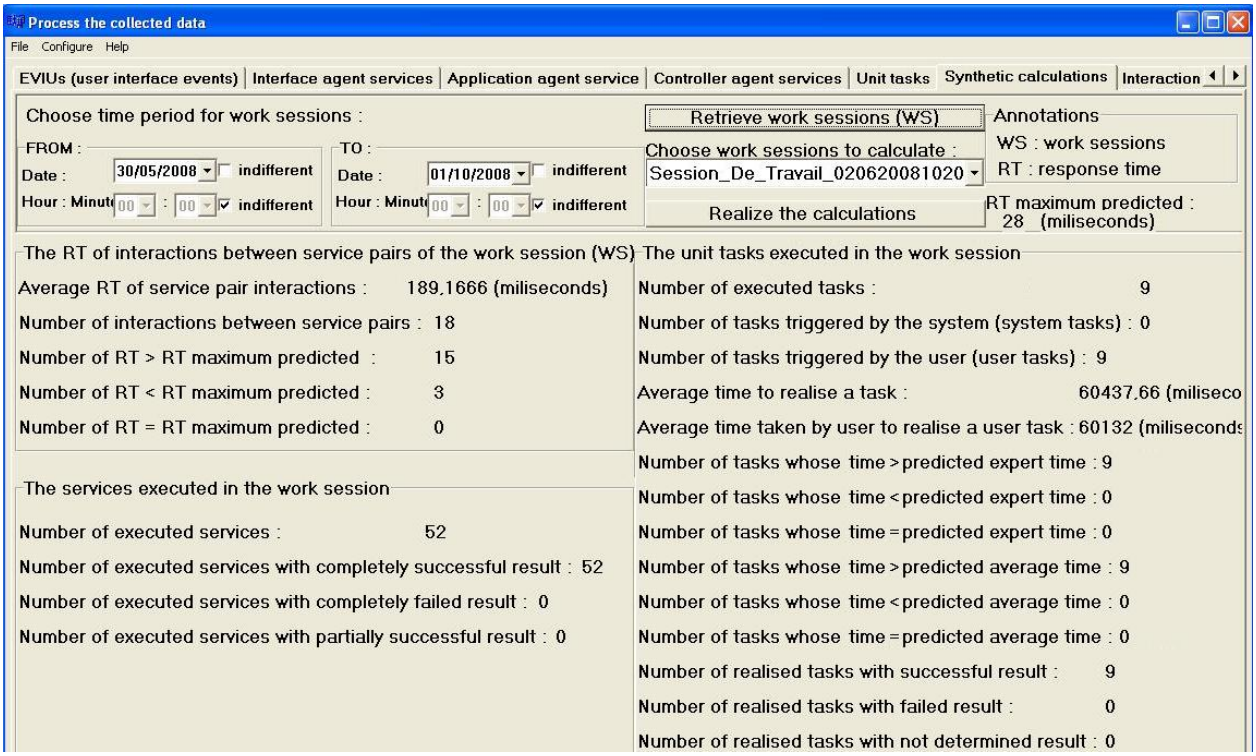

Figure 9. One of screenshots of the module 3 - Additional calculations (response time between services' interactions, average time taken by a task, results of service executions & comparisons)

Such predicted information (predicted time to execute each task as well as theoretical tasks, etc.) has been provided for the EISEval by the module 7 (see section 4.4.2) before using modules 2, 3, 4 & 5. If not, EISEval can still perform these calculations but it cannot make these comparisons.

4.4.5. Modules 4&5 (M4 & M5)

Modules 4 and 5 also belong to step 3 of the EISEval's evaluation process and they work with Petri Nets (PNs). Thus, we first present PNs and explain our choice.

Petri Nets (PNs) were proposed at the beginning of the 1960s by C.A. Petri (Petri, 1962). PNs are a graphic and mathematical tool, usually used to model and design discrete event systems formally. PNs also allow the performance of the modeled systems to be evaluated. The elementary PNs are composed of three types of objects: (1) places, represented by circles, corresponding to the system states; (2) transitions, represented by boxes or bars, corresponding to the operators who perform state changes; and (3) arcs, represented by arrows, corresponding to the connections between transitions and places. If there is a directed arc connecting a place to a transition, then this place is the input place of this transition. If there is a directed arc connecting a transition to a place, then this place is the output place of this transition.

Due to their advantages, we chose to use PNs. One of the major advantages of using PNs is that the same model is used for analyzing the behavioral properties and the performance evaluation (Zurawski and Zhou, 1994). In addition, PNs provide concurrent behavior modeling, which features "true concurrency" semantics (Genrich, 1991; Navarre et al., 2009). In EISEval, PNs are used to visually recapitulate behaviors of the users and the system agents while they were performing a certain task to facilitate the evaluation. Another way to describe a task is CTT task model (Paterno et al., 1997) that allows to a hierarchical tree of events and tasks. Comparing to CTT, PNs are more rigorous because it is based on a strict mathematical basis. Moreover, PNs allow for the representation of transitions between the system's states while it is performing a task, and PNs provide more extensions (colored PNs, temporal PNs), which can be potential developments for EISEval, whose current version only generates elementary PNs. In CTT task model, temporal relations are imposed in very rigid way, thus we find it very useful to model systems that follow a rigid business process but it is limited to use this model in supervision systems where events as well as regulation activities can be executed anytime, at the moment when the disturbances occur. However, the CTT model generation is also one of our future research topics, so we can use CTT to describe tasks whose execution follows rigid temporal relations.

As is shown in Figure 5, M4 exploits data captured by M1 and database of the *observed tasks* (i.e., *real tasks*) determined by M2, in order to generate PNs describing each *observed task*. M4 generates PNs to visually recapitulate behaviors that the users and the system agents have already performed to carry out each task. The user actions (EVIUs) and the agent actions (service executions) are represented by PNs with *places* and *transitions* to facilitate the evaluation of the system. Called *observed PNs* or *real PNs*, the generated PNs, are formally described using Petri Net Markup Language (PNML) (Billington et al., 2003; Kinder, 2004, 2005) (see also http://www.pnml.org).

These generated PNs are very useful for the evaluation because they provide evaluators with visuals that facilitate the detection of problems and inconveniences in the evaluated system. Figure 9 shows a portion of generated PNs from our study (see section 5). This is the simplest PNs among PNs generated from our study and chosen for the sake of simplicity, clarity and pedagogy. These PNs visually recapitulate behaviors that human subject 9 has already performed in order to execute task 3 ("*Send a message to passengers at station*"). Using these PNs, the evaluators can detect this subject's useless actions and errors.

M5 allows evaluators to compare the *real PNs* for a given task with the *theoretical PNs* predicted and specified by the system designer for executing this task. These *theoretical PNs* can be called "*PNs to be executed*". Evaluators can also compare the real PNs of different users, which is very useful for detecting problems with an interface, a system or users. Some examples of the problems that can be detected are: incorrect or useless user actions, non-optimal paths chosen by users to carry out tasks, failed service interactions, user characteristics and/or habits. In addition, the evaluators can assess and compare different users' abilities or supervise the progress of a given user's abilities.

There are two types of errors that evaluators can detect using PNs: system errors, which are the result from service execution failures, and user errors. Several sub-types of *user errors* can be distinguished:

- Redundant (or useless) actions – these errors do not cause any damage, but they are not necessary to carry out the task. For this type of error, the evaluators need to pay attention to the repetition of certain useless actions.
- Erroneous actions – an incorrect path (i.e., erroneous actions on the part of the user) was chosen to carry out a task. If the user persists in using this path, the task cannot be accomplished. An incorrect way consists of erroneous actions made by the user.
- Non-optimal navigation – a non-optimal path chosen by the user to carry out a given task.
- Bad actions or habits – the objective of the actions is to carry out the task, but the user has performed these actions improperly. Depending on the application domain or the evaluated system, an action/habit is considered a bad one. For example, re-entering the same text instead of using the "copy/paste" operation, typing a text that is available to be selected instead of selecting it, and typing a text in the "hunt-and-peck" method instead of using touch typing) are all examples of bad actions in the transport domain and IAS – transport supervision system evaluated in our study (see. Section 5). If such actions are performed many times of a certain user, evaluators can affirm that he/she has a bad habit.

The PNs shown in Figure 10 illustrate some user errors from our study. Indeed, at first, the human subject in our study wants to send a message to a station but he has already performed an erroneous action by choosing the wrong one. Consequently, he had to click on the *Cancel* button to close this window. Then he performed useless actions by clicking many times on the checkboxes that represent transport lines.

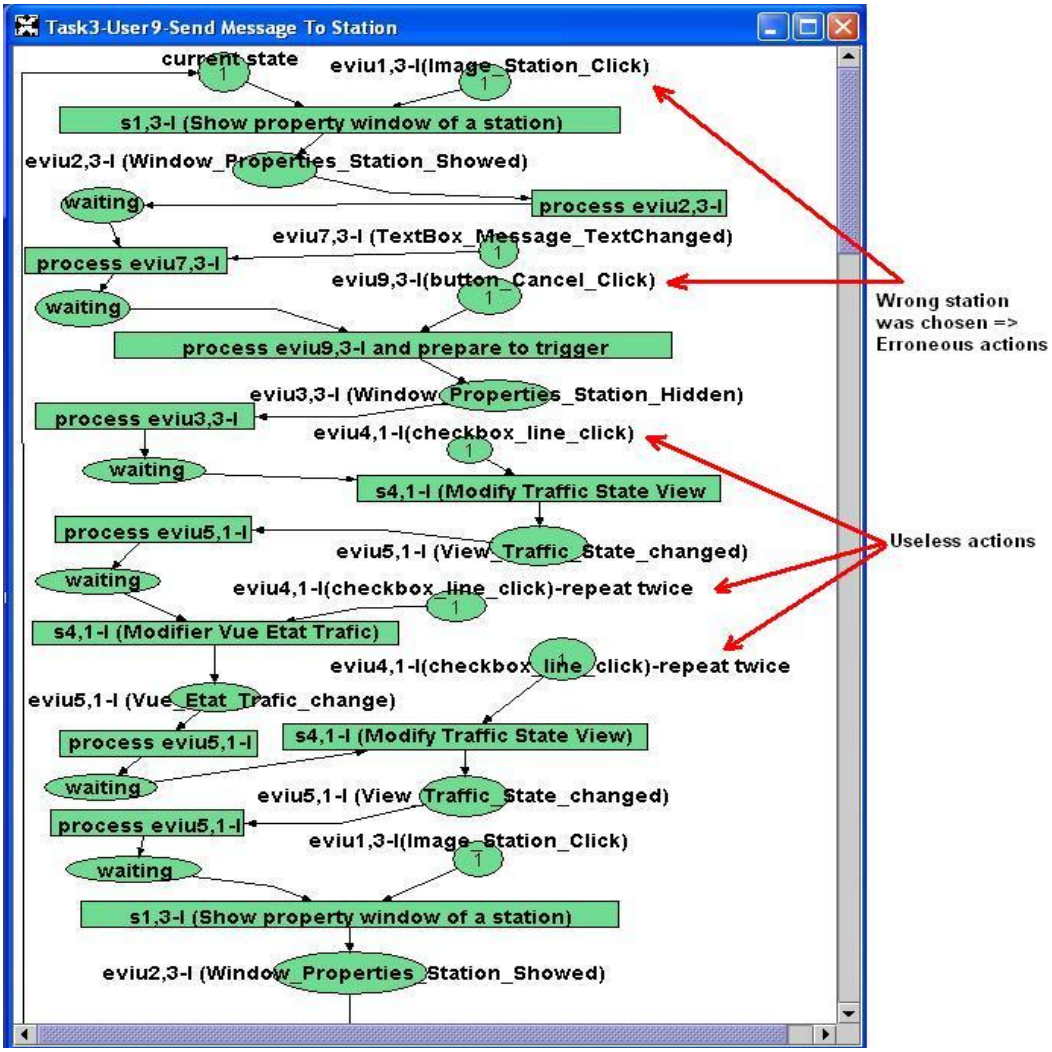

Figure 10. Petri Net generated by human subject 9's execution of task 3: "Send a message to passengers of station" in our study (eviuM,N-I: the UI of event M of the interface agent N; sM,N-I: the service M of the interface agent N)

User actions are almost sequential, whereas service executions can be either sequential or parallel. For example, in our study, the task, "*send messages to vehicles*" involves a parallel execution of two services: *send messages to passengers in a certain vehicle* and *send messages to the driver of a certain vehicle*. M5 is used to compare PNs of different users or compare *real PNs* and *theoretical PNs* predicted by the system designer or compare *real PNs* of different users. The current version of EISEval only allows PNs to be visualized so that evaluators can compare them. The improvement of the M5 takes part in our future research (section 7).

4.4.6. Module 6 (M6)

M3, M4 and M5 analyze the captured data. Evaluators then must interpret these analysis results (statistics, measure calculations, generated PNs) in order to criticize the system and suggest the necessary improvements to the system designers. This is step 4 of the EISEval evaluation process and module 6 (M6) corresponds to this step. Indeed, it provides the evaluators with the necessary indications for interpreting the analysis results from the other modules. Figure 11 shows a screenshot of this module.

The module 6 provides evaluators with an open list of predetermined criteria to help them interpret and evaluate the target system according to these criteria. An evaluation criterion supplied by the module 6 can be:

a) Ergonomic Criteria. There are many sources of ergonomic criteria, rules and style guides (Bastien and Scapin, 1993; Vanderdonckt, 1994; Smith and Mosier, 1986; etc.), however, the current version of module 6 uses mainly ergonomic criteria of (Bastien and Scapin, 1993), such as legibility, prompting, immediate feedback and error protection. The evaluator can add other ergonomic criteria if necessary, for example, in our study the evaluator added two criteria specific to the IAS – the evaluated system (see section 5 below).

b) Quality Attributes. At this moment, the module 6 supplies some quality attributes in order to evaluate some non-functional aspects of the systems, such as: response time between agents' services (calculated by measuring the time from the service request to the service provision (Lee and Hwang, 2004), system reliability (the ability of a system or a

component to perform its required functions under the stated conditions for a specified period of time (IEEE, 1990)), etc. The evaluator can add other quality attributes if necessary.

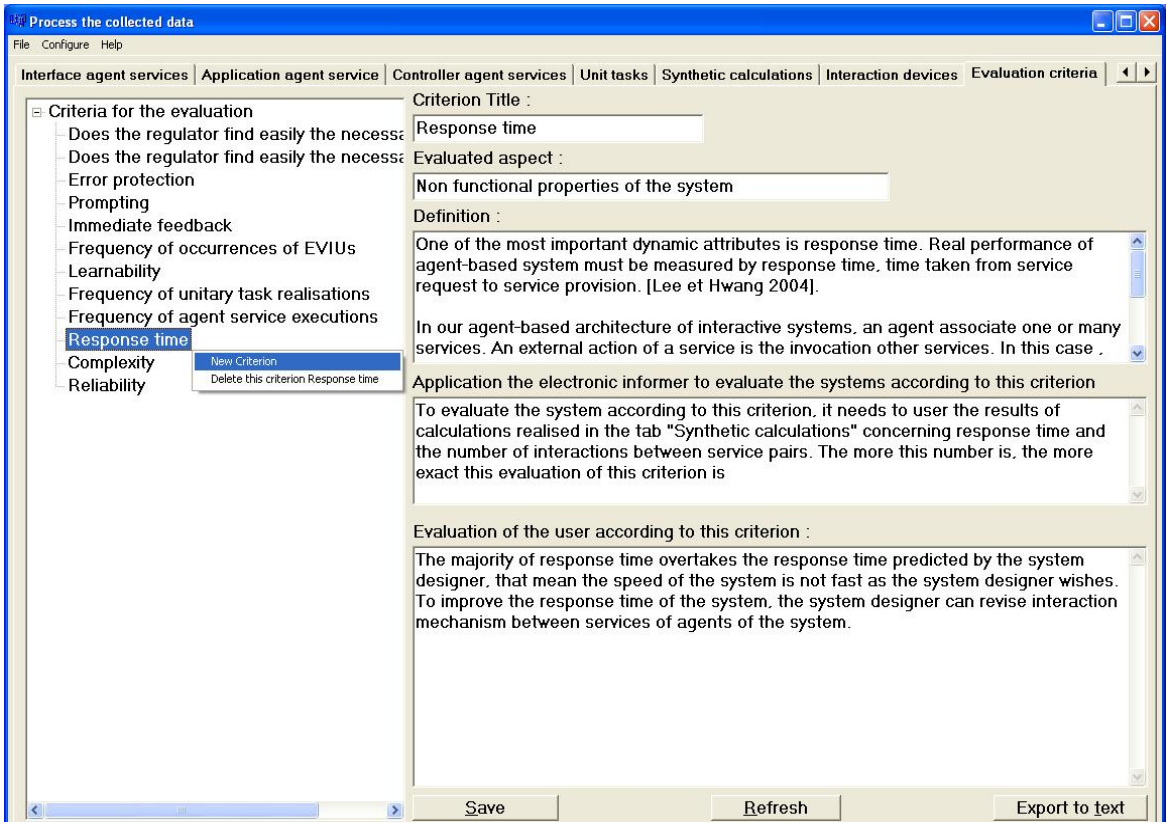

Figure 11. Screenshot produced by M6: evaluators criticize the system and propose improvements based on the provided criteria

Each criterion is composed of four parts:

1. Name of the criterion (ex. *legibility*) and evaluated aspects (UI, non-functional properties, user characteristics).
2. Definition of the criterion.

   This section provides a definition of this criterion.
3. The way of interpreting EISEval's analysis results in order to evaluate the target system based on this criterion.

   This section provides the evaluator with a concrete indication of how to interpret analysis results to evaluate the system according to this criterion (ex. the occurrence number and frequency, provided by EISEval's module 3, of user interface events (EVIUs) that allow the user to see more clearly the interface (such as *Zoom Out, Zoom In, scroll bar, change the size of views*, etc.), can be used to evaluate the system interface according to this criterion "*legibility*", etc.). In other word, this part shows the associations between this criterion and analysis results of EISEval's other modules.
4. Criticism and improvements proposed by the evaluator based on this criterion.

   This section allows evaluators to enter their criticism about the evaluated system based on this criterion and propose their suggestions for improvement (ex. through study, EISEval let us know that user interfaces events (EVIUs) (such as *Zoom Out, Zoom In, scroll bar, change the size of views*, etc.), never occurred because their frequency, provided by the EISEval's module 3, is zero). Therefore, the system's interface can be easy to be read by the user and *legibility* of this system can be appreciated, etc.). The module 6 also allows generating and saving all criticism and suggestions based on all the module 6's criteria into a document in order to be exploited later.

In brief, the current version of M6 associates the EISEval's analysis results to an open list of predetermined criteria (e.g., ergonomic criteria or quality attributes) in order to help evaluators interpret the results. Thus, the evaluators are supported to criticize the system more easily and suggest useful improvements to the designers. These criteria can be generic or specific to the evaluated system and can be related to three different system aspects: the UI, non-functional properties, and user characteristics. Evaluators can also, if necessary, add new criteria since the criteria list is open, ex. because each evaluated interactive system can require specific criteria, the evaluator can add some specific criteria to the list so that the system can be evaluated according to these criteria. The evaluator has already done this in our study (see section 5). Specific criteria often involve business concepts of the domain of the evaluated system.

At present, the associations between the evaluation criteria of module 6 and the analysis results of the remaining modules are not yet formalized. As a result, module 6 is only able to provide evaluators with indications in order to help them interpret these analysis results. In the future, it will be necessary to extend and formalize these associations as much

as possible, which would increase the automation of module 6. This is one of our future research topics (section 7). In the next section, we will present a table summarizing the generic criteria provided by module 6.

4.5. EISEval's Generic Criteria Summary

Table 1 summarizes important generic criteria provided by the M6 of EISEval to evaluate an interactive system. This list of criteria is modifiable and open so that evaluators may add new criteria. We also present here very brief additional information on the evaluation of the target system according to each criterion in our case study (after presenting the table). Please note that each analysis result (e.g., measures, frequency, time, etc.) provided by the other EISEval modules can be used to help evaluators interpret and evaluate the target system based on several criteria and each criterion can be interpreted and evaluated using several analysis results.

Table 1. Summary of some of the Generic Criteria provided by EISEval

Criterion Definition	**Way of interpreting EISEval's analysis results to evaluate the target system based on this criterion**
Legibility: Legibility concerns the lexical characteristics of the information presented on the screen that may hamper or facilitate the reading of this information (character brightness, font size, interword spacing, etc.) (Bastien and Scapin, 1993).	1) Frequency of occurrence of EVIUs that allow the user to see the interface more clearly (such as Zoom out, Zoom in and change window size,etc). If this frequency is high, then evaluators can doubt the legibility of the interface. 2) Time interval between the occurrence of an EVIU corresponding to the display of a window and the occurrence of another EVIU corresponding to the next action of the user on this window. Evaluators can find these EVIUs using generated Petri Nets (PNs). If this interval is long, they may ask the UI designer and/or the user a question: "*Why did the user take so long to perform the next action on this window?*". Legibility of information on this window may be one of the possible reasons.
Prompting: Prompting has a broader definition than usual. Here it refers to the means available in order to lead the users to making specific actions whether it is data entry or other tasks. This criterion also refers to all the means that help users to know the alternatives when several actions are possible depending on the context. Prompting also concerns status information, that is information about the actual state or context of the system, as well as information concerning help facilities and their accessibility (Bastien and Scapin, 1993).	1) Frequency of occurrence of EVIUs that correspond to the display of error messages when the user enters invalid data or executes erroneous actions. If this frequency is high, then evaluators can assume that the user has made many mistakes during his interaction with the interface and doubt this criterion of the UI. 2) Frequency of a task that a user cannot accomplish, frequency of a chosen non-optimal way to perform tasks. If users were often unable to perform certain tasks or if they often chose the non-optimal way, and the optimal path was subsequently rarely followed (when the system provides the user with several possible ways to perform a task), then evaluators can assume that bad prompting of the UI may have been one of possible reasons. 3) Frequency of occurrence of events corresponding to help consultation If the user often looks up a help document in order to use the system, then evaluators can assume that the user encountered difficulties during interactions with the system. 4) Frequency of EVIUs corresponding to useless manipulations, erroneous navigation and user actions. This type of navigation and types of actions are shown on the generated PNs. If this frequency is high, then the evaluator can assume that the prompting is not good. Evaluators can interpret the UI's ineffectiveness as leading to the user's difficulties in interpreting it (Lecerof and Paterno, 1998). In this case, for example, they can advise the system designers to add more explicit labels or necessary guidance. 5) Time intervals between the occurrence of EVIUs that correspond to the user's actions during task performance and the time intervals between executions of user tasks. When the user performs a task, a series of user actions (in terms of EVIUs) occur. If the intervals between these EVIUs are long, evaluators can interpret this as the user taking a long time to find a solution (in terms of navigation) to perform the task. Bad prompting can be one possible reason for this. The same may be considered when there is a long time interval between tasks to be performed. In both of cases, an additional evaluation method, such as a questionnaire, can be used to obtain a more complete evaluation of this criterion.
Immediate feedback: concerns system responses to users' actions. These actions may be simple keyed entries or more complex transactions such as stacked commands. In all cases computer responses must be provided, they should be fast with appropriate and consistent timing for different types of transactions. (Bastien and	Observed Petri Nets (generated PNs) that visually describe task performances and the moment where involved events (shown in PNs) occur can be used to evaluate the system according to this criterion. If EVIUs corresponding to system responses to user actions do not exist or if the time intervals between their occurrences are long then the system can be underestimated for this criterion.

Scapin, 1993).	
Design Complexity: This refers to the degree of complexity in system design (Lee and Hwang 2004). Complexity affects system speed: the more design complex the system, the more slowly the system is likely to work. In addition, it is more difficult for designers to understand and manage source code for complex systems.	Additional measurements calculated by module 3: The ratio between two measurements – the number of service interactions and the number of executed tasks – can be used to evaluate a system according to the "complexity" criterion. This ratio lets evaluators know the average number of service interactions between agents needed to execute a task. If there are too many interactions, then evaluators may consider the system design to be too complex. In short, the higher this ratio, the more complex the system design. If the design is too complex, evaluators can advise designers to reorganize the agent services to maximally reduce the number of agent interactions needed to execute a task. In general, there are still many discussions and much literature on complexity.
Response time (RT): This quality attribute is calculated by measuring the time from the service request to the service provision system (Lee and Hwang 2004).	1) The total number of service interactions 2) Average RT between service interactions 3) M3 information on service interactions have long RTs and the number of service interactions that have RTs longer/shorter than an acceptable threshold (predetermined by configuring the system using module 7) EISEval only lets evaluators know the real response time - it cannot explain why the response time is what it is (e.g., due to agent coordination patterns, interaction technique used or other reasons such as network load, system platform load or application priority, etc.). The system designer is in charge of considering these issues.
Reliability: This is the ability of a system or a component to perform its required functions under the stated conditions for a specified period of time (IEEE, 1990).	Evaluators can use the ratio between two measures: the "number of executed services with a successful result" and the "number of executed services" to evaluate the IAS according to this criterion. This ratio is called the success ratio; the higher this ratio, the more reliable the system's operation.
Improvement suggestions arising from the frequency of event occurrence (services, EVIUs, tasks). Frequency of events provided by the M3: From these frequencies, evaluators can comment on the system as well as users. They can also provide useful advice to help the designer improve the system. This criterion aims to answer the questions: "*Which events (services, EVIUs, tasks) occurred frequently?*" "*Which events occurred rarely?*" and "*Which events never occurred?*". The answers to these questions are useful for evaluating the interactive system.	1) Frequency of service execution If some services are often executed, then recommendations are made to the designer to improve the execution time of these services, thus contributing to the execution time of the overall system. It is necessary to optimize these services (by reducing execution time and memory consumption, by optimizing the code for implementing these services, etc.). 2) Frequency of task performance In section 4.2 above, we distinguished between two types of tasks: *system tasks* (initialized by a service of an interface agent or an application agent) and *user tasks* (initialized by an EVIU or a series of EVIUs). If some system tasks are often performed, then the evaluator can suggest that the designer seek to accelerate their performance time. To achieve this goal, we must speed up the execution time of involved services and the interactions between them (response time). If some user tasks are often performed, then the evaluator can suggest that the designer facilitate their implementation. To achieve this goal, the designer is advised to improve the views, related windows (e.g., the designer can add different ways to perform these tasks through shortcuts or speed keys. It is necessary to allow users to begin these tasks anywhere in the system.). As a result, the productivity of the system and its user satisfaction can be increased. 3) Frequency of EVIU occurrence: If an EVIU is often triggered by the user, then the designer may be asked to provide the user with several ways to trigger this EVIU (mouse, hotkeys). The EVIU frequency may let the evaluator know whether or not the current appearance of the UI is effective. If some EVIUs of a window occur with high frequency, then their associated widgets should be placed so that they are close to each other on the interface or so that they are within the same group on the interface. This organization reduces mouse movement and the interface becomes more efficient (Sears, 1995). If some EVIUs never occur or if they only occur rarely, then their associated widgets could be removed or hidden. For example, when you want to print a document, a few options for paper, text can be hidden in a separate option box (Sears, 1995) because these options are rarely employed by users.
User performance: User performance can be evaluated when users use the system and they can be compared, for example, in order to organize training sessions.	1) The frequency of tasks accomplished by users can be employed to evaluate this criterion. Experienced users tend to be to achieve more tasks, while novice users may have difficulties performing the tasks 2) The way(s) selected among the several possible methods provided by the designer to perform a task can also be used to evaluate this criterion.

	An experienced user tends to choose the optimal way out of the several possible ways to accomplish a task. 3) The frequency of EVIUs corresponding to useless manipulations, erroneous navigation and user actions are also exploitable. An experienced user has a tendency to commit fewer useless manipulations and erroneous actions. 4) The time taken to perform a task can also be useful (see the "Average time taken by the user to realize a user task" measure). An experienced user tends to perform tasks quickly. 5) The frequency of event occurrence corresponding to consulting help can be used to assess this criterion. The experienced user refers less frequently to help documentation.

It is important to notice that such criteria list is open. In our case study (section 5 below), the target system was evaluated according to these criteria, but it was also evaluated based on other specific criteria, such as:

- *Prompting*: In our case study, we learn that human subjects had performed useless or redundant manipulations (such as re-typing available text instead of selecting it in the box) and erroneous actions or navigation while they performed tasks. Some subjects took a long time (about 2 minutes) to begin a task. Some could not find the optimal way to perform a task, and some users even chose the longest way to perform a task.

  In general, our case study reveals that the evaluated system is not good at all with respect to this criterion.

  Improvements were also proposed after this case study for UI and user interactive functions for two *interface agents* of the evaluated system. These proposals were intended to improve the system with respect to this criterion.

- *Legibility*: EVIUs like Zoom in and Zoom out never occurred in our case study. Therefore, the evaluator can assume that the target system has compliance with this criterion in our case study.

- *Performance of users*: In our case study, subjects can be clearly distinguished and evaluated based on their abilities. Subject habits and frequent errors can also be found.

- *Suggestions made by frequency of event occurrence (services, EVIUs, tasks)*: In our case study, some EVIUs never occurred. Therefore, some proposals were made to improve the UI of *interface agents* involving these EVIUs.

- *Design complexity*: In our study, for each human subject the evaluator perceived that one or two interactions (on average) ware needed to execute a task. Evaluators can therefore assume that the design of the evaluated system is not complex.

4.6. *Concluding remarks about EISEval*

EISEval must respect certain design principles in order to remedy the drawbacks of traditional evaluation tools in general and traditional EIs in particular. In order to achieve this objective, we have proposed a meta-model of our architecture model for agent-based interactive systems. EISEval was developed based on this description, but it can also evaluate interactive systems that do not use our architecture model.

EISEval is composed of seven modules. In order to use EISEval to evaluate a given interactive system, M1 needs to be launched to capture objective data. Among seven EISEval modules, M1 is the only module developed as an individual system, and so it can be individually launched. Before analyzing these captured data, the evaluators configure EISEval using M7. Then, the data captured by M1 are analyzed by modules M2, M3, M4 & M5, which show the analysis results in various forms. Evaluators must interpret these analysis results in order to criticize the system and suggest necessary improvements to the designers, and M6 helps the evaluators do this work.

Table 2. Summary of important EISEval features

Data capture	Performed data analysis and results display	Assistance in interpreting analysis results to evaluate different aspects of the target system	Other features
Captured events are: - Logical high-level EVIUs (such as menu selection and clicked buttons) - Events at device level (such as mouse clicks, and pressed keys) (also be captured) - Service interactions between agents. These events are stored in databases for future use. Capturing methods: The first version of EISEval uses instrumentation code. A small quantity code is inserted into the target system to generate the necessary output data. This technique can be used in one of two ways: 1) The EISEval M1 functions like a server and the target system like a client, connecting an M1 and sending events to it. 2) M1 receives event logs from the target system. Improving M1 is an important part of future research topics.	Data are retrieved from the database populated by the M1 in order to perform some analyses, such as: - Measurements and statistics on frequencies, time, successes and failures, for example. - Generated Petri Nets are used to visually reconstitute the activities of the user and the target system. - Analysis results are shown to the user in tabular and/or graphical form.	- Provision of an open and modifiable list of criteria to help evaluators interpret and evaluate the target system according to these criteria. - Possibility for evaluators to define and add new criteria (both generic and specific) to the predetermined list of criteria. - Evaluated aspects: UI, certain non-functional properties of the target system and users' performance.	- EISEval is designed to be modular way (7 modules). - EISEval is reconfigured to evaluate different interactive systems. - EISEval aims at evaluating interactive systems that use our architecture model although it is still able to evaluate other systems (by considering them as a special case of our agent-based architecture model. (See the end of section 4.2)). - EISEval was used in a case study to evaluate an interactive system and we intend to organize another case study for a second system.

From this table, we can view improvements in EISEval and its differences compared with traditional evaluation tools (presented in §3). Indeed, TFWWG (Tools For Working With Guidelines) use guidelines mainly to analyze the static aspects of UIs (position of a control, color combinations) and are not based on objective data generated by real user activities. EISEval evaluates the system using criteria based on objective data. Traditional Electronic Informers generally capture HCI data to perform an evaluation of the UIs of interactive systems. They usually assess the UI aspect and do not consider other aspects. In addition, traditional EIs do not have assistance or indications to help the evaluators interpret analysis results. Improving such assistance in EISEval is one of our future research topics (section 7).

The EISEval environment was applied in a study that evaluated an agent-based supervisory interactive system in the transport domain, called Information Assistance System (IAS). In order to illustrate activities of the above modules, we used some screenshots and data from our study. We present the results of this study in the next section.

5. Case study: Applying EISEval to evaluate a supervision system for an urban transport network

We organized a study to apply EISEval to evaluate the Information Assistance System (IAS), an agent-based system used to supervise an urban transport network (buses, trams). This study involves several human subjects in order to execute some supervision and regulation tasks. Each subject has his/her own data as well as the analysis results. Using this study, the IAS was criticized and some improvements were suggested. This study also clarifies the advantages and disadvantages of using EISEval. After explaining how we set up the study (section 5), we summarize the results of this study in the section 6.

5.1. System deployment

In order to carry out the study, we deployed four systems on three connected machines that communicated via a local network. All these systems communicated through the socket mechanism. These four systems were designed and developed by the SART project (SART is the French acronym for Traffic Regulation Assistance System):

- Machine 1 The Exploitation Assistance System (EAS) was deployed on the first machine. EAS is a public transport network simulator for tram and bus traffic. It was developed in Quest (Queue Event Simulation Tool), a simulation tool with discrete events (http: // www.delmia.com). This simulator automatically determines vehicle locations and the various disturbances of vehicles in real time. This system sends the information on vehicle locations to IAS and information on the vehicles' disturbances to both IAS and DAS.
- Machine 2 Both the Decision Assistance System (DAS) and the Information Assistance System (IAS) were deployed on the second machine.
    - DAS is a software solution used to help human regulators (or operators in control room) with their decision-making and to resolve complex problems. When the DAS receives disturbance information from EAS, it provides regulators with possible regulation solutions. These regulators can choose the most appropriate solution based on their experience, or they can choose no solution and provide their own.
    - IAS is an agent-based interactive system designed to help regulators supervise the public transport network. It presents information on the current state of the traffic to regulators and allows them to send the necessary messages or commands to vehicle drivers, as well as to the passenger in the stations or inside the vehicles. This IAS is based on our architecture model and composed of six *interface agents*: *State of the Line* (Figure 12), which allows regulators to supervise a given tram or bus line visually; *Traffic Status*, which informs regulators about the status of all the vehicles on the line (e.g., disturbances statuses (such as lateness, earliness) or normal status (on time)) (Figure 13); *Vehicle*, which represents a given vehicle; *Station*, which represents a given station; *Message*, which manages and sends a message to stations and/or vehicles; and *Global View*, which shows an overview of the entire transport network. (Interested readers can consult (Ezzedine et al., 2006) and (Ezzedine et al., 2008) to see the different user interfaces of this system.).

    In our study, each human subject was required to interact with these two systems, and especially the IAS system, to execute a list of tasks in our scenario. The tasks are presented in the two tables in the next section.

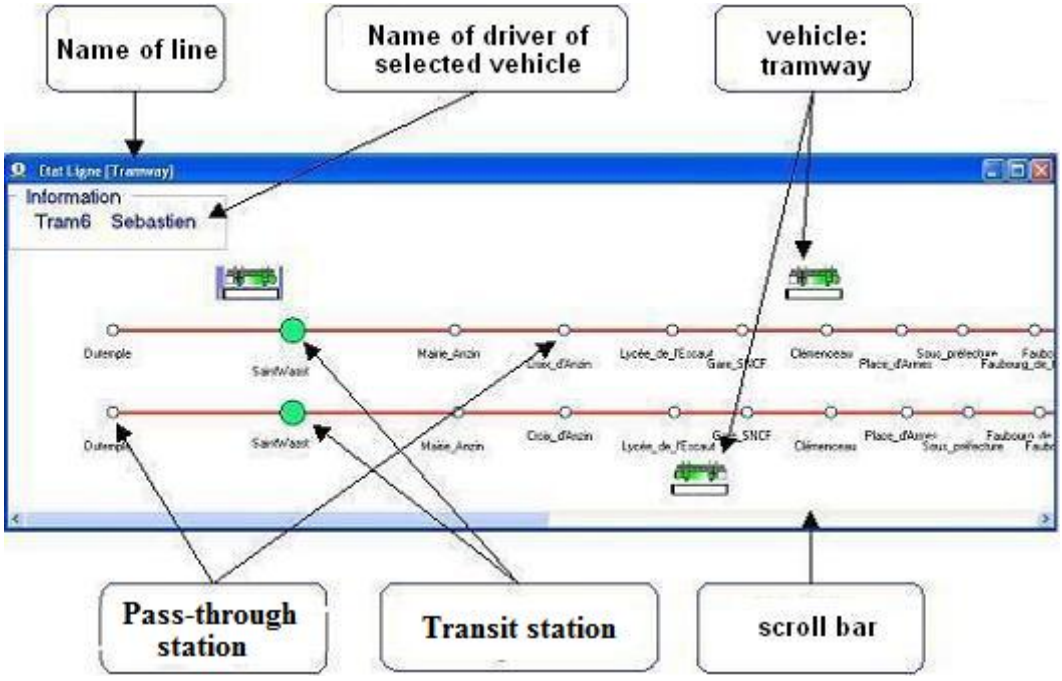

Figure 12. The IAS system's *Interface Agent*: *State of the Line*

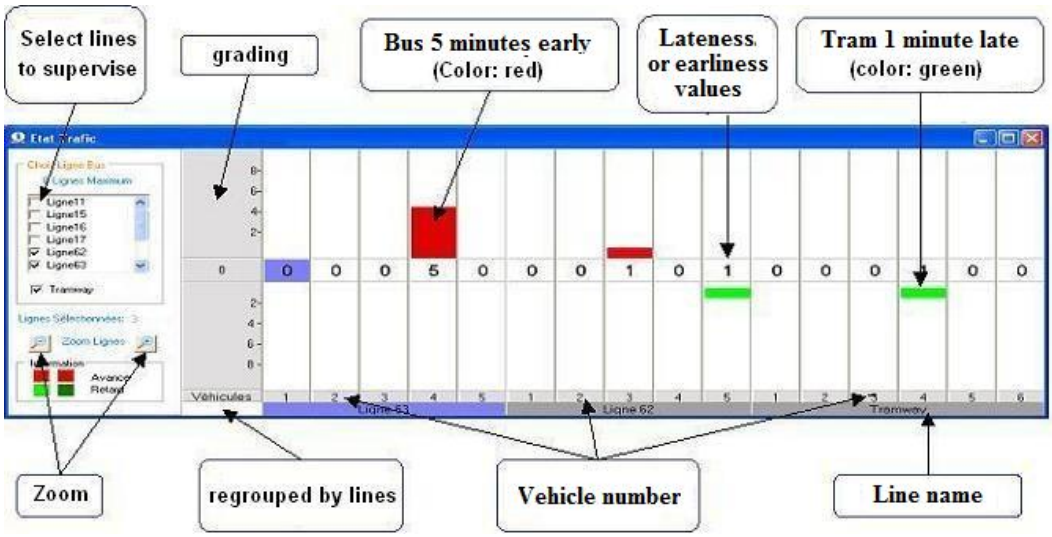

Figure 13. The IAS system's *Interface Agent*: *Traffic Status*

- Machine 3 The EISEval environment was deployed on the third machine. The evaluator used module 7 to configure EISEval in order to evaluate the IAS system (step 2 of the EISEval's evaluation process, section 4.3). EISEval's module 1 had to be launched to capture all data relative to activities of the IAS and human subjects (e.g., HCI events, service executions - step 1 of the EISEval's evaluation process, section 4.3) and store them in the database. These data will be analyzed later by other modules in order to help the evaluator criticize IAS and suggest improvements to IAS designers.

5.2. Study protocol

Ten human subjects took part in this study: one woman and nine men, ages ranging from twenty-four to thirty years, with an average age of twenty-five years. The subjects were doctoral students, engineers or future engineers in Computer Science and Automation. They were therefore quite familiar with computers, but they were not experienced users of traffic supervision programs.

During the study, EAS sent, in real-time, information on vehicle locations to IAS (through socket mechanism) that presents such information in visual and interactive interface. Each subject had to use IAS to execute nine tasks related to traffic regulations and informing drivers and passengers. Table 3 presents these nine tasks. The first six tasks were relatively simple because the subjects had to send a message to only one station or vehicle. The last three were rather complex: the subjects had to send a message to multiple stations or vehicles in the same line or different lines.

After each subject finished performing all nine tasks, we begin testing their reactions to traffic disturbances using IAS. EAS provides its user with the traffic disturbance generation functionalities and the evaluator can use them to generate traffic disturbances at any desired time. When a traffic disturbance (e.g., lateness or breakdown of a given vehicle) are generated, EAS sent information about this disturbance to both the DAS and IAS. The subjects had to use IAS to execute the necessary reactions by sending the messages to the vehicles experiencing a disturbance as well as to their next stations. Table 4 presents these necessary reactions. The subjects could also apply the solution suggested by the DAS.

We have chosen these tasks and reactions for the study because they are essential functionalities of public transport network supervision and it is necessary to evaluate whether the IAS allows user to perform them easily and or not. While the subjects were using IAS to carry out these tasks and reactions presented in two tables, EISEval's module 1 was capturing and storing all the necessary data to be analyzed later by other modules later. Module 6 helped the evaluator interpret these analysis results in order to criticize the IAS and suggest improvements to its designers. The way in which this module was used to evaluate an interactive system (IAS in this case) was already presented in previous section. We present this more explicitly in the next section.

Table 3

The nine regulation and notification tasks in the study

Tasks	**Description of tasks** (without feedback from the IAS, subjects must redo it)
T1	Send a message to the tramway station named *Railway Station SNCF*: "The next tram is going to stop 2 minutes at the station"
T2	Send these two messages to tramway vehicle N3: 1) "Stop 2 minutes at the Railway Station SNCF" for its driver 2) "We plan to stop 2 minutes at the Railway Station SNCF" for its passengers
T3	Send a message to the Station named *St Wast* on the bus line 15: "The next bus is going to stop 3 minutes at the station"
T4	Send these two messages to vehicle N4 on the bus line 15: 1) "Stop 3 minutes at the St Wast station" for its driver 2) "We plan to stop 3 minutes at the St Wast station" for its passengers
T5	Send a message to the Station named *Vaillant* on the bus line 16: "The next bus is going to stop 3 minutes at the station"
T6	Send these two messages to vehicle N5 on the bus line 16: 1) "Stop 3 minutes at the Vaillant station" for its driver 2) "We plan to stop 3 minutes at the Vaillant station" for its passengers
T7	Send a message to all three stations – *Canada*, *Ardenne*, *Concorde* – on the bus line 62: "Station will not be served tomorrow and the day after tomorrow for repairs"
T8	Send a message to all stations on all tram & bus lines: "Possible traffic disturbances on Monday of next week"
T9	Send a message to all vehicles of all tram & bus lines: "Have a good holiday!"

Table 4

Reactions to be executed by subjects when traffic disturbances happen

Disturbances	**Actions to be executed if traffic disturbances happen**
A vehicle is late for X minutes (X under or equal to 7 minutes)	1. Close the warning window related to this lateness. 2. Use the IAS to send the following two messages to this vehicle: - "X minutes late; please speed up" for its driver, and - "Be careful. The tram/bus is going to go faster to compensate its lateness" for its passengers.
A vehicle is late for X minutes (X over 7 minutes)	1. Close the warning window related to this lateness. 2. Use the IAS to send the following two messages to this vehicle: - "You are X minutes late" for its driver, and - "Attention. There will be a delay of X minutes" for its passengers. 3. Send a message to the next station visited by this vehicle: "X minutes late", and then apply the regulation solution suggested by the DAS.
Vehicle breakdown	1. Close the warning window related to this breakdown. 2. Use the IAS to send the following two messages to this vehicle: - "The repair service will arrive in 10 minutes" for its driver, and - "Bus has broken down. Sorry for this inconvenience" for its passengers.

5.3. The way of using EISEval to evaluate an interactive system

As presented above in the 4.3 section, EISEval is configured to evaluate the IAS (module 7) and the data of each human subject in our study are collected and analyzed (modules 1-5) in three first steps, with a certain intervention from the evaluator (module 2 presented above).

In the step 4, EISEval helps the evaluator interpret analysis results in order to evaluate aspects of target interactive system and the EISEval's module 6 corresponds to this step. Indeed, in our study, after that each subject uses IAS to perform all these nine tasks as well as some reactions to the traffic disturbances (generated by the EAS), according to the scenario presented above in the two tables 3 and 4, objective data of each human are collected, stored and analyzed by the modules 1-5; the evaluator used a list of evaluation criteria supplied by module 6 (M6) to interpret analysis results of the captured data, criticize the IAS system and suggest useful improvement to the IAS designers. This evaluator followed a three-steps method:

(1) the evaluator accesses to and studies each evaluation criterion of M6 to understand how the analysis results from modules 3, 4 and 5 were interpreted in order to evaluate the system based on this criterion; In other word, the evaluator understand the association between this criterion and the analysis results provided by other modules (part 3 among 4 parts of a criterion, see section 4.4.6 -module 6 above);

(2) for each evaluation criterion , the evaluator retrieves the associated analysis results to interpret them, using these M6 indications (part 3 among 4 parts of a criterion, see section 4.4.6 -module 6 above);

(3) the evaluator enters into M6 the criticisms and suggestions necessary for future improvements of the system based on each evaluation criterion (part 4 among 4 parts of a criterion, see section 4.4.6 -module 6 above).

After following this method for all the module 6's criteria, this module allows to generate and save all criticism and suggestions based on all its criteria (called, evaluation results) into a document in order to be exploited later (ex. report, see section below). The format of this document contains a list of paragraphs. Each paragraph is composed of four parts: 1) Name of criterion, 2) Definition of this criterion, 3) The way of interpreting EISEval's analysis results in order to evaluate the target system based on this criterion 4) Criticism and improvements proposed by the evaluator based on this criterion.

These four parts correspond to four parts of each criterion provided by the module 6 (see section 4.4.6 above).

6. A summary and discussion of our study results

An evaluation report has been written by the evaluator; this report presents, in very detailed way, evaluation results, based on the module 6's criteria. The main content of the report is based on the document generated by the module 6; after that the evaluator has already followed the three-step method to interpret analysis results of the captured data, criticize the IAS system and suggest useful improvement to the IAS designers (presented above). This report, in which each criterion is used to evaluate the target system, is very long and presents screenshots containing analysis results (statistics, measures, calculations of the modules 3, complicated Petri Nets of modules 4&5 because subjects had already performed many useless and erroneous actions in our study – see types of errors in the 4.4.5 section above) as well as the evaluator's criticism and improvement suggestions (in terms of text and images). We only present below, in very brief way, the reduced content of some paragraphs of this report (without images and improvement suggestions). Each paragraph corresponds to an evaluation criterion and it is composed of four-parts format (presented above). The module 6's criteria, examined in this study for IAS evaluation and presented in this report, include generic ergonomic criteria (such as legibility, prompting, immediate feedback, error protection, etc.), specific criteria added to the module 6 because of IAS's particular features, and quality attributes to evaluate non-functional properties of IAS.

As presented above, in order to evaluate the IAS system, two following criteria specific to IAS have already been added to the module 6: "*The rapidity of regulators to find necessary stations or vehicles*" and "*The rapidity and facility of regulators to treat disturbances of vehicles*". The reduced content of paragraphs concerning these two criteria, in our report, is presented below:

- Specific Criterion *"The rapidity of regulators to find necessary lines, stations or vehicles" was added to module 6*:

  Definition*:* This criterion evaluates whether the user interface (UI) of IAS allows its user to easily and fast find a line, a station or a vehicle or not. Indeed, 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 regulators to rapidly find the corresponding stations or vehicles on the screen. It is important criterion that is specific to IAS because it affects the work productivity of regulators and insures timely regulations.

  The way of interpreting EISEval's analysis results in order to evaluate the target system based on this criterion: Useless actions (view types of user errors in 4.4.5 section above) already performed by the user before finding a line, a station or vehicle can be used to evaluate the system based on this criterion. Through the Petri Nets (PNs - generated by the module 4), which reconstitute the real activities of the user and the system to perform the tasks relative to sending a message to the stations/vehicles, the evaluator can determine whether or not the user has already performed these useless actions (in the form of EVIUs) before finding what was sought (lines, stations, vehicles). If the occurrence number of these EVIUs (provided by the module 3) is high, then the evaluator may conclude that the user has difficulty to find them.

Criticism and improvements proposed by the evaluator based on this criterion: The module 3 and the Petri Nets generated by module 4 let the evaluator know that, in our study, except for a few "super subjects", the remaining subjects took a relatively long time to find the right stations/vehicles on the screen in order to send a message. According to analysis results of the module 3 for each human subject), some subjects even took about 2 minutes. This difficulty was shown by a lot of useless actions done by these subjects before they could do the first action necessary to send a message (analysis results – Petri Nets from the module 4 for each human subject). For example, the Figure 10 shows a part of Petri Net generated by module 4. This Petri Net shows the execution of task 3 by subject 9. This subject had to choose the right station to send it a message. However, he had already chosen the wrong station, so he had to click on the button "*Cancel*" to close the wrong station's property window. Then, he performed a lot of useless actions before finding the right station to send a message. A set of similar Petri Nets was generated after this study for each subject. The Petri Nets show that the subjects had difficulties finding the necessary stations/vehicles on the screen, and thus the evaluator should advise the IAS designers to revise the user interface. The improvement suggestions for the IAS's UI to facilitate user are… [presented in our report, not described here by lack of space]. These UI improvements involve modifications of UI and user interactive functions of two following *interface agents* of IAS: *State of the Line*, *Traffic Status* (see Figure 12 & 13 above for the current version of these two *interface agents*).

- Specific criterion *"The rapidity and facility of regulators to treat disturbances of vehicles*", was added to module 6

  Definition: This criterion evaluates whether IAS allows its user to treat, easily and fast, disturbances (e.g., lateness or breakdown of a given vehicle) or not, especially in cases where the time intervals between the occurrences of disturbances are short, that means the disturbances occur almost simultaneously or consecutively.

  The way of interpreting EISEval's analysis results in order to evaluate the target system based on this criterion: The evaluator may use the following analysis results provided by the module 3 to evaluate the system based on this criterion:

  - The evaluator may determine whether the user has failed the treatment of disturbances. If the number of disturbances omitted is high, then measures should be taken.
  - The time interval between the occurrence of a disturbance and the occurrence of the event corresponding to its treatment can also be used to assess this criterion. If this interval is long, and we must take the consequences.
  - If the user has performed useless actions (see types of user errors in the 4.4.5 section above) before processing a disturbance, then the evaluator can interpret the user has struggled to treat it.

  Criticism and improvements proposed by the evaluator based on this criterion: During the study, each subject had to deal with from 3 to 6 disturbances. All the subjects had to send messages to the vehicles encountering a disturbance and their next stations. Generally, the average time intervals between disturbances (generated by the EAS in our study) were relatively long (1-3 minutes), thus the subjects quite easily dealt with them. One subject omitted a disturbance (analysis results from the module 3 for each human subject) because of the short interval between two disturbances (8 seconds). From this fact, the evaluator has believed that if disturbances had occurred in succession or almost simultaneously during our study, then a lot of subjects would have omitted them. Thus, IAS does not have a very good score for this criterion. The improvement suggestion is…[presented in our report, not described here by lack of space]. This improvement involves the automation of IAS's mechanism to deal with disturbances.

We also present here the reduced content of some paragraphs concerning criteria for evaluating non-functional system properties:

- Criterion *System reliability*:

  Definition: System reliability is one of non-functional system properties and it is also one of criteria provided by module 6. Reliability is "the ability of a system or a component to perform its required functions under the stated conditions for a specified period of time" (IEEE, 1990).

  The way of interpreting EISEval's analysis results in order to evaluate the target system based on this criterion: Evaluators can use the ratio between two measurements calculated by module 3: "*number of executed services with successful result*" and "*number of executed services*" to evaluate the IAS according to this criterion (Figure 9). This ratio is called success ratio; the higher this ratio, the more reliable the system operations.

  Criticism and improvements proposed by the evaluator based on this criterion: In our study, for each human subject, it is proved to the evaluator that the IAS is reliable. His/her success ratio is always maximal (analysis results from the module 3 for each human subject). For example, the subject 4 had 100% success rate since all fifty services were executed successfully (Figure 9 above).

- Criterion *Response time* (RT):

  Definition: RT between services is one of the most important non-functional attributes to be evaluated; this attribute measures the real performance of a multi-agents system (Lee and Hwang 2004). This quality attribute is calculated by measuring the time from the service request to the service provision.

  The way of interpreting EISEval's analysis results in order to evaluate the target system based on this criterion: The module 3 provides some measures concerning the RTs between agent services: the total number of service interactions, average RT between service interactions. This module also lets the evaluator know which service

interactions have long RTs, the number of service interactions that have RTs longer/shorter than an acceptable threshold (predetermined by configuring system using the module 7). In order to evaluate this criterion of the system, the evaluator may use these measures. The higher the number of service interactions is, the higher the accuracy of this evaluation is high.

Additional discussion: We should mention here what factors influence the RTs of interactive systems. According to (Lee and Hwang 2004), different agent coordination patterns influence the performance of multi-agent systems. However, agent coordination patterns are not the only thing that influences RTs; interaction techniques also influence them. If designers cannot change agent coordination patterns to improve RT due to the constraints, then the evaluator can suggest modifying the interaction techniques for the same purpose. These techniques are a key for such properties as reliability and performance (Mehta et al. 2000; Spitznagel and Garlan 2001). Nowadays, there are many interaction techniques for software systems (Mehta et al. 2000), and designers should consider this when they design an agent-based interactive system. Please note that, EISEval only let the evaluator know the real response time, it cannot explain why the response time is like that (because of agent coordination patterns, interaction technique or other reasons such as network load, system platform load, application priority, etc.). The system designer is in charge of considering these issues after receiving evaluation results from EISEval.

Criticism and improvements proposed by the evaluator based on this criterion: In agent-based interactive systems using our architecture model, information transmissions between the *application agents* and *interface agents* through *control agents* are very important. If these transmissions slow or delayed, it can cause severe consequences, especially in industrial supervision systems like IAS, because human regulators must quickly understand the current state of the real process and their commands must be sent to this real process in time, especially in cases of malfunctions. Consequently, the speed of interactions between agent services must be evaluated.

In our study, for each human subject, the evaluator found that his/her average RT between service interactions was between 120-200 milliseconds (analysis results from the module 3 for each human subject), thus the evaluator has concluded that the IAS was fast. This is not surprising since all the agents of the IAS's current version run on the same machine (a distribution on different machines certainly lead to worse results).

However, the RT of most of the service interactions is longer than that intended by the designer (28 milliseconds configured for the EISEval through the module 7), for example, in the case of a subject 4 whose results are shown in the Figure 9 above, most of his service interactions (15/18) are longer than the acceptable threshold specified by the designer. The evaluator concludes that he system speed is not as high as the system designer had desired. The system designer can be advised to revise the system, and especially the service pairs whose RTs often take longer than what is acceptable. Improving the RT of such service pairs will improve the overall system performance.

- Criterion *Design Complexity*:

  Definition*:* it is one of non-functional system properties, and it is also one of the criteria provided by module 6. This quality attribute is used to examine the degree of complexity in the system design (Lee and Hwang 2004). Complexity affects system speed: in general, the more complex the system, the more slowly the system risks to work. In addition, a system whose the design is complex can be more difficult for designers to understand and thus more difficult for them to manage the source code.

  The way of interpreting EISEval's analysis results in order to evaluate the target system based on this criterion: In order to evaluate the complexity of an agent-based interactive system, evaluators can use the additional measurements calculated by the module 3 (Figure 9). The ratio between two measurements – the number of service interactions and the number of executed tasks – can be used to evaluate the system according to the "complexity" criterion. This ratio lets evaluators know the average number of service interactions between agents needed to execute a task. If there are too many interactions, then evaluators may consider that the system design is complex. In short, the higher this ratio, the more complex the system design. If the design is too complex, evaluators can advise the designers to reorganize the agent services in order to reduce as much as possible the number of agent interactions needed to execute a task. Additional discussion: there are still many discussions and literatures about the complexity.

  Criticism and improvements proposed by the evaluator based on this criterion: In our study, for each human subject, the evaluator perceived that an average of one or two interactions was needed to execute a task (analysis results from the module 3 for each human subject). For example, the subject 4 needed eighteen service interactions to execute nine tasks (Figure 9) – in average: two interactions/tasks. This ratio is acceptable. The evaluator could consider that IAS service organization is not very complex and that the IAS did not have to perform many interactions to execute a task.

The other ergonomic criteria (such as legibility, prompting, immediate feedback, error protection, etc.) have also been examined in our study. The evaluator's criticisms and improvement suggestions for the target system based on these criteria are already presented in the remaining paragraphs of the report. We only use two tables 5 and 6 here to summarize the results of task executions following the study (see nine tasks in the table I). Please note that these tables are only used in this paper in order to briefly present the tasks' execution results of human subjects in our study, for the sake of simplicity and briefness. Indeed, based on these tasks' execution results of human subjects, we can classify the subjects into two groups:

- Group 1 contains the subjects who were able to execute all nine tasks, including the last three complex tasks. The subjects in this group had already chosen the optimal path to execute the first six tasks, although three subjects

performed useless actions. Subject 4 chose a non-optimal path to execute task 7, and subject 9 chose a non-optimal path to execute the last three tasks. All remaining subjects of this group chose the optimal path to execute these three tasks, although useless actions still appeared.

- Group 2 contains the subjects who were unable to execute the last three complex tasks among nine tasks. They also chose the optimal path to execute first six tasks with several useless actions. While executing task 7 relative to sending a message to three different stations, they could not find the way of sending a message to all three stations in only one operation, as consequence, they sent the message to each station individually, thus they had to execute this task three times in succession, which is the longest path. Subject 3 was able to execute task 8 by following a non-optimal path, but this subject was unable to execute task 9. All remaining subjects of this group were unable to execute tasks 8 and 9.

In this paper, we have only presented some of our study results and some of the criteria provided by module 6. In fact, following this study, many good points and bad points were detected for the IAS. For each detected problem, improvements were proposed. Most of these improvements involved modifications of UI and interactive functions of three IAS *interface agents*: *State of the Line*, *Traffic Status* and *Message*. In our study, some subjects chose an incorrect or non-optimal path to execute tasks and some subjects could not accomplish every task. Thanks to these improvements, the new UI of these IAS *interface agents* allow users to find the best path to accomplish regulation and supervision tasks much more quickly. The evaluator also proposed to change the IAS's mechanism to deal with disturbances (i.e., delays or vehicle breakdowns). This new mechanism uses a queue to collect disturbances and allows automated access to vehicles encountering a disturbance (instead manual access at this moment) in order to improve and accelerate dealing with disturbances. Such improvement suggestions were presented in our report and they are not presented in this paper. During this study, the EISEval environment was also tested to detect its strengths and weaknesses. This is really useful for us to decide on future research projects.

Table 5

Group #1 subjects

<table>
<tr><th>Subject</th><th>Execution of the first six tasks. Were there useless actions or not?</th><th>Path followed when executing the task 7 (optimal path or not?). Were there useless actions or not?</th><th>Path followed when executing the tasks 8 & 9 (optimal path or not?). Were there useless actions or not?</th><th>Message “Have a good holiday!” in task 9 is perceived or not?</th><th>Average time to execute a task (in seconds)</th></tr>
<tr><td>4</td><td>Optimal path was chosen. No useless action.</td><td>Non-optimal path was chosen. No useless action.</td><td>Optimal path was chosen. No useless action.</td><td rowspan="4">No. These subjects had to re-enter this message using the keyboard when executing task 9 although this message was available in the “listbox”</td><td>60.1</td></tr>
<tr><td>8</td><td rowspan="3">Optimal path was chosen. There were useless actions and navigations (an example is shown in Figure 10).</td><td>Optimal path was chosen.</td><td rowspan="2">Optimal path was chosen. No useless action.</td><td>40.5</td></tr>
<tr><td>6</td><td>There were many useless actions before finding the right path.</td><td>27.7</td></tr>
<tr><td>9</td><td>Non-optimal path was chosen. There were many useless actions before finding the right path.</td><td>Non-optimal path was chosen. No useless action.</td><td>38.8</td></tr>
<tr><td>1</td><td>Optimal path was chosen. No useless action.</td><td colspan="2">Optimal path was chosen. No useless action.</td><td>Yes. This subject selected this message.</td><td>31.1</td></tr>
</table>

Table 6

Group #2 subjects

Subject	**Execution of the first six tasks. Were there useless actions or not?**	**Path followed when executing the task 7** (optimal path or not?). **Were there useless actions or not?**	**Execution of the tasks 8, 9. Were there useless actions or not?**	**Average time to execute a task** (in seconds)
3	Optimal path was chosen. There were useless actions and navigations.	The longest path chosen. These subjects had to repeat the same actions three times in a row. Specifically, before finding the right path, subject 7 performed useless actions.	Non-optimal path chosen to execute task 8. This subject was unable to execute task 9. No useless action	52.0
2			These four subjects were unable to execute tasks 8 & 9.	68.6
5				35.6
7				41.0
10				37.7

7. Conclusion and future research

In this paper, after examining the architecture models of interactive systems, we proposed a hybrid architecture model to combine advantages of functional and structural models. A generic and configurable environment called EISEval was also proposed to support the evaluation of interactive systems in general and of agent-based interactive systems that use our architecture model in particular. We designed EISEval as an extensive EI environment (not a TFWWG tool) based on objective data captured from the interactions between users and the UI as well as between agents themselves in real situations. EISEval's activity is based on the EIs principles. However, EISEval also uses ergonomic criteria, as well as other non-functional criteria to help evaluators interpret captured objective data and evaluate the target interactive system. EISEval can remedy the drawbacks of traditional evaluation tools in general and traditional EIs in particular. It provides some original functionalities to extend the evaluation possibilities of traditional EIs and to make the evaluation more complete. The multi-steps evaluation process and seven modules of EISEval were presented.

We conducted a study with ten human subjects. In this study, EISEval environment was applied to evaluate an agent-based interactive system called IAS, designed according to our architecture model and intended to supervise a public transport network. The results of this study were briefly described in this paper. Several strengths and weaknesses in IAS were revealed, and we proposed improvements corresponding to the weaknesses. This study also allows detecting some strengths and weaknesses in the EISEval environment.

Consequently, we propose several perspectives for future research:

- We intend to conduct other studies in which:
  - EISEval will be used to evaluate other interactive systems with other evaluators and more or less novice human subjects. We intend to let other evaluators use EISEval in order to determine whether or not EISEval is easy for them to use. Novice usability practitioners should be invited to use EISEval because difficulties are often particularly pronounced for them (Howarth et al. 2009).
  - In future studies, the target interactive systems should be in other application domains. We believe that EISEval can be applied to evaluate interactive systems whose application domains are not in transport; other studies can demonstrate this.
- As presented above (4.4.2 section), we aim at developing a visual development environment to help developers design, in an interactive and visual way, interactive systems that use our architecture model. This environment can also allow generating the description file that contains necessary input information to be provided for the EISEval's module 7. At this moment, the evaluator still has to configure EISEval by inputting the module 7 via its user interfaces; as a consequence, it takes the evaluator a significant time.
- EISEval's module 1 is responsible for capturing events occurring in the evaluated interactive system. This module was developed as an individual system that is completely independent from the remaining modules. The link between the modules is a common database. It is necessary to improve this module so that:
  - it can work effectively in the cases in which the network is not high speed At this time, the evaluated interactive system and module 1 communicate through a socket mechanism. Module 1 captures every event when it occurs in the interactive system. This capture method works very effectively in local area networks, which tend to be quite high speed. However, if module 1 and the interactive system communicate through a big network whose speed is slower (e.g., the Internet), then this data capture method is not effective any more. Indeed, in such an environment, the continuous transfer of large data quantities can slow down the network; the network can even be blocked.

- it can capture events of other types of applications (e.g., web and mobile applications) and stores the captured data in a database that can be used later by the remaining modules. At this time, module 1 captures the events occurring in interactive applications running on PC computers. If we want to apply this environment to evaluate other types of applications, then it is necessary to improve this module.

- Modules 4 and 5 generate the Petri Nets (PNs), which reconstitute the processes of the real user and system activities needed to execute tasks. After this study, we noticed that the PNs generated are often very complex because most of the subjects have performed several redundant and/or erroneous actions (see types of errors in the 4.4.5 section above). Indeed, after module 4 generates the PNs, module 5 allows the evaluator to visualize the generated PNs (of different users) and/or the designer's theoretical PNs, so that they can compare them. As a result, the evaluator was overloaded by these PNs (approximately 130 PNs were generated after the study). In the future, it will be necessary to improve module 5 to help the evaluator analyze these PNs by detecting the user's erroneous actions and useless manipulations (in terms of redundant transitions and the states of PNs). Such an improvement would facilitate evaluators' work.

- As presented above (4.4.5 section), we intend to add a new functionality to the EISEval's module 4 so that it can generate, the CTT task model (Paterno et al. 1997). This model is useful for describing tasks whose execution follows rigid temporal relations.

- At this time, the associations between the evaluation criteria of module 6 and the analysis results of the remaining modules (3, 4 & 5) are not yet formalized. As a result, module 6 is only able to provide evaluators with indications in order to help them interpret these analysis results. In the future, it will be necessary to formalize these associations as much as possible, which would increase the automation of module 6. Moreover, it is necessary to enrich the module 6 as well as other modules with other evaluation criteria as well as corresponding analysis results (measures, statistics).

- The IAS that was evaluated by EISEval is a static interactive system. In order to evaluate adaptive systems that are able to change their behavior and interfaces according to the each context of use, it is necessary to take the context into account during the evaluation. At this time, evaluators themselves are responsible for taking the context into account when they interpret the EISEval analysis results, using the criteria of module 6. In this case, in order to evaluate a given adaptive system, evaluators can use the EISEval environment to evaluate system operations and the system interface in various contexts. By interpreting analysis results, evaluators can know in what contexts errors and problems appear the least and the most; then they can suggest that the designers revise the system for the contexts in which errors and problems appear the most. As a result, the evaluators can use objective data to evaluate the quality of the system's adaptation. From this perspective, it will be necessary to take the use**d** context into account when we formalize the associations between the evaluation criteria of module 6 and the analysis results of the remaining modules.

Acknowledgements

This research was supported financially by International Campus on Safety and Intermodality in Transportation (CISIT), the *Nord/Pas-de-Calais* Region, and the European Community (FEDER). The authors gratefully acknowledge the support of these institutions.

References

Abran, A., Khelifi, A., Suryn, W., Seffah, A., 2003. Consolidating the ISO Usability Models. In: Proceedings of 11th Int. Software Quality Management Conference, Glasgow, Scotland, UK.

Alexander, J., Cockburn, A., Lobb, R., 2008. AppMonitor: A Tool for Recording User Actions in Unmodified Windows Applications. Behavior Research Methods 40(2) 413-421.

Andre., T.S., Rex Hartson., H., Belz, S.M., Mcreary, F.A., 2001. The user action framework: a reliable foundation for usability engineering support tools. International Journal of Human-Computer Studies 54(1) 107-136.

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

Bastien, JMC., Scapin, DL., 1993. Ergonomic Criteria for the evaluation of human-computer interfaces. Technical Report n°156, INRIA, Rocquencourt, 1993

Bastien, JMC., Scapin, DL., 1995. Evaluating a user interface with ergonomic criteria. International Journal of Human-Computer Interaction 7, 105-121.

Beaudouin-Lafon, M., 2000. Instrumental Interaction: an Interaction Model for Designing Post-WIMP Interfaces. In: Proceedings of ACM Human Factors in Computing Systems, CHI'2000, The Hague (Netherlands). ACM Press, pp. 446-453.

Beirekdar, A., 2004. A methodology for automating web usability and accessibility evaluation by guideline. PhD Thesis, Facultés Universitaires Notre-Dame de la Paix, Namur.

Billington, J., Christensen, S., Hee, K.V., Kindler, E., Kummer, O., Petrucci, L., Post, R., Stehno, C., Weber, M., 2003. The Petri Net Markup Language: Concepts, Technology, and Tools. In: preliminary version of a submission to the conference proceedings of the ICATPN 2003, Eindhoven, Netherlands.

Bortolaso, C., Bach, C., Dubois, E., 2011. MACS: A combination of a Formal Mixed Interaction Model with an Informal Creative Session (regular paper), ACM SIGCHI conference Engineering Interactive Computing Systems (EICS 2011), Pisa, Italy, 13/06/11-16/06/11, ACM SIGCHI, pp. 63-72.

Calvary, G, Coutaz, J., Nigay, L., 1997. From Single-User Architectural Design to PAC*: a Generic Software Architecture Model for CSCW. In: Proceedings of CHI 97. ACM publ., pp. 242-249.

Chalon, R., David, B. T., 2004. IRVO: an Architectural Model for Collaborative Interaction in Mixed Reality Environments. In: Proc. of the Workshop MIXER'04, Funchal, Madeira, January 13, 2004, CEUR Workshop Proceedings, ISSN 1613-0073.

Chalon, R., David B. T., 2004. Modélisation de l'interaction collaborative dans les systèmes de Réalité Mixte. In: Actes de la 16ème conférence francophone sur l'Interaction Homme-Machine (IHM'04). Namur, 1-3 septembre 2004. ACM Press, ISBN 1-58113-926-8, pp. 37-44.

Charfi, S., Trablesi, A., Ezzedine, H., Kolski, C., 2011. Towards ergonomic guidelines integration within graphical interface controls for the evaluation of the IS. MSLT 2011, First IEEE International Conference on Mobility, Security and Logistics in Transport (May 31 - June 1-3), Hammamet, Tunisia, pp. 76-82.

Coutaz, J., 1987. PAC, an Object-Oriented Model for Dialog Design. In: 2nd IFIP International Conference on Human-Computer Interaction, Stuttgart, Germany, pp. 431-436.

Coutaz, J., 1990. Interface homme-ordinateur, conception et réalisation. Dunod, Paris.

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

Dewan, P. 1998. Architectures for Collaborative Applications, in: Beaudouin-Lafon, M.(Ed.), Collaborative Systems. John Wiley & Sons Ltd.

Dix, A., Finlay, J., Abowd, G., Beale, G., 1993. Human-Computer Interaction. Prentice-Hall, New Jersey.

Depaulis, F., 2002. Vers un environnement générique d'aide au développement d'applications interactives de simulations de metamorphoses, Doctoral thesis (in French), Université de Potiers.

Dubois, E., Nigay, L., Troccaz, J., 2001. Consistency in Augmented Reality Systems. In: Proceedings of EHCI'01, IFIP WG2.7 (13.2) Conference, Toronto, May 2001. LNCS 2254, Spinger-Verlag.

Dubois, E., Gray, P., Nigay, L., 2002. Asur++: A Design Notation for Mobile Mixed Systems. In: Proceedings of Mobile HCI 2002. LNCS 2411, 2002, Springer-Verlag, pp. 123-139.

Dubois, E. 2002. Asur: un point de départ pour fédérer différents aspects de la conception d'un système interactif mobile. In: Colloque sur la mobilité, GDR-I3, 6 décembre 2002, LORIA, Nancy. 2 pages.

Dubois, E., Pinheiro da Silva, P., Gray, P., 2002. Notational Support for the Design of Augmented Reality Systems. In: Proceedings of the 9th international workshop conference DSV-IS'02, Rostock - Germany, June 2002, pp. 95-114.

Dubois, E., Nigay, L., Troccaz, J., 2003. Un Regard Unificateur sur la Réalité Augmentée : Classification et Eléments de Conception. Revue d'Interaction Homme-Machine 4 (1), 85-118.

Dubois, E., Gray, P., Nigay, L., 2003. ASUR++: Supporting the design of mobile mixed systems. Interacting with Computers, 15 (4), Elsevier, 497-520.

Dubois, E., Gray, P., Trevisan, D., Vanderdonckt, J., 2004. Workshop on Exploring the Design and Engineering of Mixed Reality Systems. In: Proceedings of 2004 International Conference on Intelligent User Interfaces (IUI'04), Funchal.

Dubois, E., Mansoux, B., Bach, C., Scapin, D., Masserey, G., Viala, J., 2004. Un modèle préliminaire du domaine des systèmes mixte. In: Actes de la 16ème conférence francophone sur l'Interaction Homme-Machine.

Dubois, E., Bortolaso, C., Gauffre, G., 2011. Models Transformations for Ubiquitous System Design Proceedings of International Workshop on Model-based Interactive Ubiquitous Systems (Modiquitous 2011), Pisa - Italia, 13/06/11, pp. 1-6.

Dubois, E., Bortolaso, C., Bach, C., Duranthon, F., Blanquer-Maumont, A., 2011. Design and Evaluation of Mixed Interactive Museographic Exhibits, International Journal of Arts and Technology, Inderscience Publishers, special issue Interactive Experiences in Multimedia and Augmented Environments, 4 (4), 408-441.

Ellis, C., Gibbs, S.J., Rein, G.L., 1991. Groupware: some issues and experiences. Communications of the ACM. 34 (1), 38-58.

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.

Ezzedine, H., Abed, M., 1997. Une méthode d'évaluation d'Interface Homme Machine de supervision d'un procédé industriel. Journal Européen des Systèmes Automatisés (RAIRO-APII-JESA) 7, 1078-1110.

Ezzedine, H., 2002. Méthodes et Modèles de Spécification et d'Evaluation des Interfaces Homme-Machine dans les systèmes industriels complexes (in French). HDR Thesis, University of Valenciennes and of Hainaut-Cambrésis.

Ezzedine, H., Kolski, C., Péninou, A., 2005. Agent-oriented design of human–computer interface:application to supervision of an urban transport network. Engineering Applications of Artificial Intelligence 18, 255-270.

Ezzedine, H., Trabelsi, A., Kolski C., 2006. Modelling of an interactive system with an agent-based architecture using Petri nets, application of the method to the supervision of a transport system. Mathematics and Computers in Simulation 70, pp 358-376.

Ezzedine, H., Bonte, T., Kolski, C., Tahon, C., 2008. 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, 281-294.

Garbay, C., Badeig, F., Caelen, J., 2012. Normative multi-agent approach to support collaborative work in distributed tangible environments. In: Steven E. Poltrock, Carla Simone, Jonathan Grudin, Gloria Mark, John Riedl (Eds.), CSCW '12 Computer Supported Cooperative Work, Seattle, WA, USA, February 11–15, 2012 – Companion Volume, ACM 2012, pp 83-86.

Gauffre, G., Dubois, E., 2011. Taking advantage of model-driven engineering foundations for mixed interaction design. In: Hussmann, H., Meixner, G., Zuehlke, D. (eds.) Model-Driven Development of Advanced User Interfaces. Studies in Computational Intelligence, Springer, Heidelberg (2011), 340, pp 219-240.

Genrich, H. J., 1991. Predicate/Transitions nets, in: Jensen, K., Rozenberg, G. (Eds.), High-Levels Petri Nets: Theory and Application. Springer, pp. 3–43.

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. Éditions Hermes, Paris,pp 207-248.

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

Grammenos, D., Akoumianakis, D., Stephanidis, C., 2000. Integrated Support for Working with Guidelines: The Sherlock Guideline Management System. International Journal of Interacting with Computers, special issue on "Tools for Working with Guidelines" 12 (3), 281-311.

Guittet, L., 1995. Contribution à l'Ingénierie des Interfaces Homme-Machine - Théorie des Interacteurs et Architecture H4 dans le système NODAOO (in french). Unpublished PhD Thesis, Poitiers, France.

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

Hong, J.I., Heer, J., Waterson, S., Landay, J.A., 2001. WebQuilt: A Proxy-based Approach to Remote Web Usability Testing. Journal ACM Transactions on Information Systems (TOIS) 19 (3), 263-385.

Hornbæk, K., 2006. Current practice in measuring usability: Challenges to usability studies and research. International Journal of Human-Computer Studies 64 (2).

Howarth, J., Smith-Jackson, T., Hartson, R., 2009. Supporting novice usability practitioners with usability engineering tools. International Journal of Human-Computer Studies 67 (6), 533–549.

IEEE, 1990. Institute of Electrical and Electronics Engineers, IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. New York, 1990

ISO/IEC 9126-1, 2001. Software engineering -- Product quality -- Part 1: Quality model. 2001 (JIS X 0129-1: 2003)

ISO 9241, 1998. Ergonomic Requirements for Office Work with Visual Display Terminals.

Ivory, M., Hearst, M.A., 1998. Improving Web Site Design. IEEE Internet Computing, Special Issue on Usability and the World Wide Web 6(2).

Jasselette, A., Keita, M., Noirhomme-Fraiture, M., Randolet, F., Vanderdonckt, J., Brussel, C.V., Grolaux, D., 2006. Automated repair tool for usability and accessibility of web sites. In: Proceedings CADUI 2006, Bucharest. Springer-Verlag.

Juras D., 2004. Modélisation des systèmes mixtes. Examen probatoire en informatique du CNAM, Centre de Grenoble, 14 décembre 2004, 42 pages.

Kindler, E., 2004. High-level Petri Nets – Transfer Format. Working Draft Version 0.5.0 of the International Standard ISO/IEC 15909 Part 2.(Submitted for a combined ISO/IEC SC7 WD/CD registration and CD ballot.).

Kindler, E., 2005. Software and Systems Engineering – High-level Petri Nets, Part 2: Transfer Format. International Standard ISO/IEC 15909-2. Working Draft Version 0.9.0, June 2005, (Submitted for a combined ISO/IEC SC7 WD/CD registration and CD ballot).

Kolski, C., Forbrig, P., David, B.T., Girard, P., Tran, C.D., Ezzedine, H., 2009. Agent-based architecture for interactive system design: current approaches, perspectives, evaluation. In: Human-Computer Interaction, Part I, HCII 2009 HCI International 2009. LNCS 5610, Springer-Verlag, San Diego, pp. 806–815.

Kolski, C., Millot, P., 1991. A rule-based approach to the ergonomic “static” evaluation of man-machine graphic interface in industrial processes. International Journal of Man-Machine Studies 35 (5), 657-674.

Kolski, C., Le Strugeon E., 1998. A review of intelligent human-machine interfaces in the light of the ARCH model. International Journal of Human-Computer Interaction 10 (3), 193-231.

Laurillau Y., Nigay L., 2002. Clover architecture for groupware. In: Proceedings of the 2002 ACM conference on Computer supported cooperative work, New Orleans, USA, November 16 - 20, 2002, pp 236-245.

Lecerof, A., Paterno, F., 1998. Automatic support for usability evaluation. IEEE Trans. on Software Engineering 24 (10).

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

Lepreux, S., Kubicki, S., Kolski, C., Caelen, J., 2012. From Centralized interactive tabletops to Distributed surfaces: the Tangiget concept. International Journal of Human-Computer Interaction, 28 (11), 709-721.

Milgram, P., Kishino, F., 1994. A taxonomy of mixed reality visual displays. IEICE Transactions on Information Systems, E77-D (12).

Milgram, P., Takemura, H., Utsumi, A., Kishino, F., 1994. Augmented Reality : a class of displays on the reality-virtuality continuum. SPIE Telemanipulator and Telepresence Technologies 2351.

Milgram, P., Drascic, D., Grodski, J.J., Restogi, A., Zhai, S., Zhou, C., 1995. Merging Real and Virtual Worlds. In: Proceedings of IMAGINA’95, Monte-Carlo, Monaco, 1-3 Feb. 1995, pp. 218-230.

Mehta, N.R., Medvidovic, N., Phadke, S., 2000. Towards a Taxonomy of Software Connectors. In: Proceedings of the 22nd international conference on Software engineering. ACM Press, 2000.

Moray, N., 1997. Human factors in process control, in: Salvendy, G. (Ed.), Handbook of human factors and ergonomics. John Wiley & Sons, pp. 1944-1971.

Morandini, M., de Moraes Rodrigues, R.L., Cerrato, M.V., Lordello Chaim, M., 2011. Project and Development of ErgoCoIn Version 2.0. HCII'11 Proceedings of the 14th international conference on Human-computer interaction: Springer-Verlag Berlin, Heidelberg. Design and development approaches - Volume Part 1.

Navarre, D., Palanque, P., Ladry, J.-F., Barboni, E., 2009. ICOs: A model-based user interface description technique dedicated to interactive systems addressing usability, reliability and scalability. ACM Trans. Comput.-Hum. Interaction 16 (4).

Nielsen, J., 1993. Usability Engineering. Boston, MA. Academic Press.

Nielsen, J., 2003. Usability 101: Introduction to Usability. Retrieved November, 2012, from http://www.useit.com/alertbox/20030825.html.

Nigay, L., Coutaz, J., 1995. A Generic Platform for Addressing the Multimodal Challenge. In: Proceedings of CHI'95, Human Factors in Computing Systems. ACM Press, NY, pp. 98-105.

Parush, A., 2000. A Database Approach to Building and Using Online Human Computer Interaction Guidelines, in: Vanderdonckt, J., Farenc C. (Eds.), Tools for Working with Guidelines. Springer-Verlag, London, pp. 77-84.

Pfaff, GE., 1985. User interface management system. Springer-Verlag, 1985.

Paganelli, L., Paternò, F., 2003. Tools for remote usability evaluation of Web applications through browser logs and task models. Behavior Research Methods, Instruments, and Computers 35 (3),, 369–378.

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.

Paterno, F., Mancini, C., Meniconi, S., 1997. ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models. In: Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction Pages, Interact'97, July'97, Sydney. ISBN:0-412-80950-8, Chapman & Hall, 362-369.

Paterno, F., Piruzza, A., Santoro, C., 2006. Remote Web usability evaluation exploiting multimodal information on user behavior. In: Proceedings CADUI 2006, Bucharest. Springer-Verlag.

Paternò, F., Russino, A., Santoro, C., 2007. Remote evaluation of Mobile Applications. In Task Models and Diagrams for User Interface Design. In: 6th International Workshop, TAMODIA 2007, Toulouse, France.

Greg Phillips, W., Graham, N., T.C., Wolfe, C., 2006. A calculus for the refinement and evolution of multi-user mobile applications. In Proceedings of Design, Specification and Verification of Interactive Systems (DSV-IS 2005), Springer LNCS, 2006, pp. 137-148.

Greg Phillips, W., Nicholas Graham, T.C., 2003. Workspaces: A Multi-level Architectural Style for Synchronous Groupware, in Proceedings of Design, Specification and Verification of Interactive Systems (DSV-IS 2003), Springer LNCS, 92-106, 2003.

Renevier, P., Nigay L., 2004. Notation de Conception pour les Systèmes Mixtes Collaboratifs et Mobiles. In : Actes des Premières Journées Francophones : Mobilité et Ubiquité (UbiMob'04), 1-3 juin 2004, Nice, Sophia-Antipolis, pp. 66-73.

Renevier, P., 2004. Systèmes Mixtes Collaboratifs sur Supports Mobiles : Conception et Réalisation (in french). Ph.D Thesis, Univ. Joseph Fourier – Grenoble I, 28 juin 2004. 220 pages.

Rukshan, A., Baravalle, A., 2011. A quantitative approach to usability evaluation of web sites. Proceedings of Advances in Computing Technology, London, United Kingdom.

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

Sears, A. 2003. Testing and evaluation. In J.A. Jacko, A. Sears (Eds.), The human-computer interaction handbook, Lawrence Erlbaum Associates, 1091-1092.

Sears, A., 1995. AIDE: A Step Toward Metric-Based Interface Development Tools. Proc. UIST'95, New York: CM Press, 1995

Senach, B., 1990. Evaluation ergonomique des IHM : Une revue de la littérature. Report INRIA n°1180, mars, 1990.

Shaer, O., Jacob, R.J., 2009. A specification paradigm for the design and implementation of tangible user interfaces, ACM TOCHI, 16 (4), 20, 1-39.

Smith, S.L., Moisier, J.N., 1986. Guidelines for designing user interface software. Report No. MTR-10090, ESD-TR-86-278, Bedford, MA:MITRE Corp,1986.

Spitznagel, B., Garlan, D., 2001. A Compositional Approach for Constructing Connectors, In: The Working IEEE/IFIP Conference on Software Architecture, Royal Netherlands Academy of Arts and Sciences Amsterdam, The Netherlands.

Tarpin-Bernard, F., David, B., 1999. AMF : un modèle d'architecture multi-agents multi-facettes. TSI 18 (5), 555-586.

Tran, C. D.., Ezzedine, H., Kolski, C. 2008. Evaluation of agent-based interactive systems: proposal of an electronic informer using Petri Nets. Journal of Universal Computer Science 14 (19), 3202-3216.

Vanderdonckt, J., 1994. Guide ergonomique des interfaces homme-machine. Facultés universitaires Notre-Dame de la Paix à Namur (Belgique), Presses Universitaires de Namur, Namur, 1994.

Vanderdonckt, J., 1999. Development milestones towards a tool for working with guidelines. Interacting with Computers 12(2), 81-118.

Vanderdonckt, J., Farenc, F. (Eds.) 2000. Tools for Working With Guidelines TFWWG'2000. Springer Verlag.

Wolfe, C., David Smith, J., Nicholas Graham, T.C., 2010. Fiia: A model-based approach to engineering collaborative augmented reality. In Emmanuel Dubois, Philip Gray, and Laurence Nigay, editors, The Engineering of Mixed Reality Systems, Springer, 2010, pp. 293-312.

Wolfe, C., Nicholas Graham, T.C., Greg Phillips, W., Roy, B., 2009. Fiia: User-Centered Development of Adaptive Groupware Systems, in Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems, ACM, pp. 275-284.

Wolfe, C., Nicholas Graham, T.C., Greg Phillips, W., An Incremental Algorithm for High-Performance Runtime Model Consistency, In Proceedings of Model Driven Engineering Languages and Systems (MODELS 2009), pages 357-371. Springer LNCS, 2009.

Wu, J., Nicholas Graham, T.C., 2007. Toward Quality-Centered Design of Groupware Architectures, in Proceedings of Engineering Interactive Systems, 18 pages, 2007

Yamada, S., Hong, J.K, Sugita, S., 1995. Development and evaluation of hypermedia for museum education: validation of metrics. ACM Trans. Comput.-Hum. Interact. 2 (4).

Zettlemoyer, L.S., Amant, R.S., Dulberg, M.S., 1999. IBOTS: Agent control through the user interface. In: Proceedings of the International Conference on Intelligent User Interfaces, Redondo Beach, CA,31-37.

Zurawski, R., Zhou, MengChu., 1994. Petri nets and industrial applications: A tutorial. IEEE Transactions on Industrial Electronics 41 (6), 567-583.