Vers un environnement générique et configurable pour l'aide à l'évaluation des systèmes interactifs à base d'agents, Application à un Système d'Aide à l'Information voyageur dans le domaine des transports commun (bus, Tram)

Chi Dung Tran

Université de Valenciennes et du Hainaut-Cambrésis

Numéro d'ordre : 09/17

THESE

pour l'obtention du

Doctorat de l'Université de Valenciennes et du Hainaut-Cambrésis

Spécialité Automatique et Informatique des Systèmes Industriels et Humains

Mention informatique

Rédigée par

Chi Dung TRAN

Ingénieur en Informatique

Vers un environnement générique et configurable pour l'aide à l'évaluation des systèmes interactifs à base d'agents

Application à un Système d'Aide à l'Information voyageur

Soutenue publiquement le 15 Juillet 2009 devant le jury composé de :

Franck Tarpin-Bernard	MdC HDR, INSA de Lyon, France	Rapporteur
Jean Vanderdonckt	Professeur, Université catholique de Louvain, Belgique	Rapporteur
Ahmed Seffah	Professeur, EHL (Suisse), Concordia University (Canada)	Examinateur
Slim Hammadi	Professeur, Ecole Centrale de Lille, France	Président
Peter Wieringa	Professeur, Delft University of Technology, Pays-Bas	Examinateur
Houcine Ezzedine	Professeur, UVHC, France	Co-directeur
Christophe Kolski	Professeur, UVHC, France	Co-directeur

A mes Parents

A tous ceux qui me sont chers

Remerciements

Ces travaux de thèse ont été réalisés au Laboratoire d'Automatique, de Mécanique et d'Informatique industrielles et Humaines (LAMIH), U.M.R. C.N.R.S. 8530, à l'université de Valenciennes et du Hainaut-Cambrésis (UVHC) au sein du groupe de recherche « Raisonnement Automatique et Interaction Homme-Machine » (RAIHM) sous la direction du professeur Christophe Kolski et du professeur Houcine Ezzedine. Ces travaux s'intègrent dans le cadre du projet SART (Système d'Aide à la Régulation de Trafic) cofinancé par la région Nord-Pas de Calais et le FEDER, réunissant plusieurs partenaires : l'INRETS, le LAMIH, le LAGIS et la société TRANSVILLES.

Je tiens à remercier chaleureusement le professeur Christophe Kolski qui m'a accueilli dans son équipe de recherche RAIHM et le professeur Houcine Ezzedine qui m'a suivi et encadré tout au long de mes travaux. Ils m'ont guidé, conseillé et aidé avec enthousiasme durant ma thèse. Cette thèse n'aurait pas pu voir le jour sans eux.

Pour m'avoir fait l'honneur d'accepter d'être mes rapporteurs, je remercie beaucoup Monsieur Franck Tarpin-Bernard, MdC HDR à l'INSA Lyon, France et Monsieur Jean Vanderdonckt, Professeur à l'université Caltholique de Louvain, Belgique. Je remercie aussi beaucoup les autres membres du jury : le Professeur Amed Seffah, le Professeur Slim Hammadi et le Professeur Peter Wieringa pour m'avoir fait l'honneur d'évaluer et de juger la qualité de mon travail de thèse.

Je remercie le conseil régional Nord-Pas de Calais et le Fonds Européen de Développement Régional (FEDER) pour avoir contribué au financement de cette thèse.

Pour l'aide concernant la préparation de la plateforme logicielle d'éxpérimentation pendant ma thèse, je remercie Monsieur Jacques Darque, ingénieur à l'AIP (Atelier Inter établissements de Productique) de l'université de Valenciennes et du Hainaut-Cambrésis, de même que Mme Thérèse Bonte, ingénieur de recherche au LAMIH.

Je remercie aussi les membres du LAMIH et particulièrement les membres de l'équipe RAIHM pour les souvenirs et les moments agréables passés durant ma thèse. Je pense particulièrement aux personnes avec qui j'ai partagé le bureau 010 : Selem, Firas,Wided et Makram. Nous avons eu des discussions et des conversations utiles.

Je ne saurais oublier tous ceux qui sont restés présents quelles que soient les épreuves et qui m'ont toujours soutenu :

> Merci à mes parents qui m'ont donné la chance de poursuivre mes études et qui m'ont toujours encouragé pendant les moments difficiles.
>
> Merci à ma grande sœur à qui j'ai beaucoup manqué. Elle m'a beaucoup aidé, notamment pendant la préparation du départ en France.

Merci à tous ceux dont je n'ai pas parlé et qui mériteraient d'être cités ici.

Table des matières

Table des figures

Chapitre 1

Chapitre 2

Chapitre 3

Chapitre 4

Chapitre 5 et Annexe 1

Liste des tableaux

Liste des **acronymes**

AMF – Architecture Multi-agents multi-Facettes

DESTINE – Design & Evaluation STudio for INtent-based Ergonomic web sites

CSS – Cascading Style Sheets

DOM – Document Object Model

DTD – Document Type Definition

EISEval – Environment for Interactive System Evaluation

EvalTUGL – Evaluation Tool Using Guidelines

EVIU – EVénément Interface Utilisateur

GDL – Guideline Definition Language

GMS – Guideline Management System

HTML – HyperText Markup Language

IEISEval – Integrated Environment for Interactive System Evaluation

IHM – Interaction Homme-Machine

MESIA – Mouchard électronique dédié à l'Evaluation des Systèmes Interactifs orientés Agents

MM WebRemUSINE – Multimodal Web Remote USer INterface Evaluator

MVC – Modèle – View – Controller

OCL – Object Constraint Language

PAC – Presentation - Abstraction - Control

RemUSINE – Remote USer INterface Evaluator

RdP – Réseau de Petri

SAD – Système d'Aide à la Décision

SAE – Système d'Aide à l'Exploitation

SAI – Système d'Aide à l'Information voyageurs

SART – Sytème d'Aide à la Régulation de Trafic

TdR – Temps de Réponse

UIDL – User Interface Description Language

UIML – User Interface Markup Language

UML – Unified Modeling Language

UsiXML - USer Interface eXtensible Markup Language

USINE – USer INterface Evaluator

WebRemUSINE – Web Remote USer INterface Evaluator

WET – Web Event-logging Tool

WIMP – Windows, Icons, Menu, Pointer

WML – Wireless Markup Language

XiML – eXtensible Interface Markup Language

XML – Extensible Markup Language

XUL – XML-based User interface Language

Introduction générale

L'évaluation des systèmes interactifs est un domaine de recherche très actif depuis plus d'une trentaine d'années. En parallèle, le domaine de l'interaction homme-machine accorde une grande importance aux architectures des systèmes interactifs (architectures fonctionnelles, architectures structurelles dites aussi à base d'agents et architectures hybrides). En effet, les modèles d'architecture visent à guider les développeurs dans leur conception, développement des systèmes interactifs. Grâce à ces modèles, la réutilisabilité, la flexibilité des applications sont augmentées et la maintenance est facilitée. Cependant, ils ne sont pas suffisants pour produire des applications de qualité. Afin d'atteindre ce but, l'évaluation est essentielle.

L'évaluation des systèmes interactifs permet de mettre l'accent sur les défauts de conception ; elle peut aussi aider les concepteurs à améliorer les systèmes interactifs déjà conçus en facilitant la détection de leurs problèmes et défauts. On considère qu'une interface utilisateur doit augmenter le confort, la satisfaction et la productivité des utilisateurs, et aussi diminuer les risques d'erreurs que les utilisateurs peuvent commettre pendant leur utilisation du système. De nombreuses méthodes et outils d'évaluation ont été proposés pour aider les concepteurs à détecter les problèmes de l'interface utilisateur, à comprendre les difficultés que les utilisateurs rencontrent lorsqu'ils interagissent avec les systèmes interactifs, et à améliorer les IHM. Pourtant, la plupart des méthodes d'évaluation traditionnelles ne prennent pas en compte les spécificités des architectures à base d'agent des systèmes interactifs pour l'évaluation. Cette exploitation des spécificités de telles architectures apporte plusieurs avantages. En effet, elle peut aider l'évaluateur à mieux comprendre les problèmes du système interactif et à déterminer les améliorations nécessaires. En raison de l'apparition des systèmes interactifs à base d'agent, il y a aussi de nouveaux besoins en termes d'outils d'aide à l'évaluation de ces systèmes. Nous nous situons dans ce contexte.

Les mouchards électroniques sont des outils très utilisés pour évaluer les systèmes interactifs. Ils capturent automatiquement des données objectives durant l'interaction entre l'utilisateur et le système interactif (les actions des utilisateurs et leurs répercutions sur le système) en situation de travail réelle. Ces données capturées sont analysées ultérieurement d'une manière manuelle ou automatique. Les résultats d'analyse (classifications, statistiques, recherche de patterns d'interactions, etc.) sont affichés sous formes compréhensibles pour aider les évaluateurs à effectuer leurs tâches. Plusieurs électroniques mouchards ont été proposés depuis une vingtaine d'années mais ils disposent encore de limitations.

Afin de remédier à ces problèmes, dans cette thèse, nous proposons et développons un environnement générique et configurable appelé EISEval (Environnement for Interactive System Evaluation) pour aider à évaluer les systèmes interactifs à base d'agents. Cet environnement prend en compte les spécificités de l'architecture à base d'agent des systèmes interactifs pour l'évaluation. Il peut aussi étendre les possibilités effectuées par les mouchards traditionnels et il fournit en plus quelques fonctionnalités originales ajoutées par rapport à ces mouchards traditionnels. Après la capture des interactions en lien avec le système interactif à base d'agents (entre les utilisateurs et le système et entre les agents eux-mêmes), cet environnement effectue les analyses des données capturées comme des calculs de quelques mesures, des calculs statistiques, des classifications et la génération de Réseaux de Petri (RdPs) permettant de reconstituer les processus des activités réelles de l'utilisateur et du système pour réaliser les tâches. Ces résultats d'analyse sont affichés sous forme visuelle et compréhensible pour l'évaluateur afin de l'aider à évaluer différents aspects d'un système interactif : l'interface, des propriétés non fonctionnelles du système (par exemple le temps de réponse, etc.) et des propriétés de l'utilisateur (la facilité à utiliser le système, la comparaison

entre les performances d'utilisateurs, les progrès en termes de performance lors de l'utilisation du système, etc.). L'environnement fournit aussi à l'évaluateur les indications nécessaires pour l'aider à critiquer tous ces aspects et proposer des améliorations au concepteur. Cette approche a été validée par une expérimentation dans laquelle cet environnement a été appliqué pour un prototype de Système d'Aide à l'Information voyageurs conçu pour aider des opérateurs humains (appelés régulateurs) en salle de contrôle à superviser le transport public terrestre Valenciennois (Réseau de bus et Tramway). Ce mémoire de thèse est structuré en cinq chapitres.

Le premier chapitre présente les modèles d'architecture des systèmes interactifs. Deux grands types de modèles d'architecture sont distingués : les modèles *fonctionnels* et *structurels*. Quelques modèles d'architecture représentatifs parmi les plus connus de ces deux grands types (comme Seeheim, Arch, PAC, MVC, AMF) et des modèles d'architectures hybrides (comme H4, PAC-Amodeus) sont introduits. Après une discussion sur ces grands types de modèles d'architecture, notre architecture mixte à base d'agents qui emprunte les principes de ces deux types de modèles d'architecture pour exploiter leurs avantages est présentée. Une discussion sur cette architecture termine ce chapitre.

Le deuxième chapitre concerne l'évaluation des systèmes interactifs. D'abord, les principes de base de l'évaluation des systèmes interactifs, ainsi que des classifications des méthodes d'évaluation sont introduits. Ensuite, un état de l'art des outils d'aide à l'évaluation est présenté. Parmi plusieurs types d'outils d'aide à l'évaluation, nous nous intéressons surtout à deux types : des outils utilisant des règles ergonomiques (ergonomic guidelines) et des mouchards électroniques. Nous examinons, sans souci d'exhaustivité, des outils representatifs (en lien éventuel avec des aspects méthodologiques) concernés par ces deux types. Ces méthodologies et outils examinés sont apparus durant la dernière décennie comme WebTango, DESTINE, Sherlock, WebRemUSINE, MultiDevice RemUSINE, IBOT, WET, WebQuilt, MESIA, etc. Enfin, des tableaux dressent une synthèse et permettent la confrontation de ces méthodologies et outils en se basant sur différentes dimensions.

Le troisième chapitre est consacré à notre contribution à ce domaine de recherche, sous la forme de l'environnement d'évaluation EISEval. D'abord, les limitations du mouchard MESIA proposé lors de travaux précédents de notre équipe sont présentées. Les limitations des mouchards électroniques traditionnels sont discutées dans la suite. Puis, les principes de notre environnement sont expliqués. Le respect de ces principes permet à cet environnement EISEval de prendre en compte les spécificités de l'architecture à base d'agent des systèmes interactifs, d'étendre les possibilités offertes par les mouchards électroniques traditionnels et d'aller plus loin qu'eux pour évaluer les différents aspects d'un système interactif cités plus haut. Afin d'atteindre cet objectif, une description formelle de l'architecture à base d'agents des systèmes interactifs est proposée et présentée ici. Cet environnement est proposé sur la base de cette description. Il fournit des fonctionnalités originales par rapport aux mouchards traditionnels. Enfin, les sept modules indépendants qui composent cet environnement sont présentés. Le principe de couplage entre l'environnement EISEval et le système interactif à évaluer est abordé également. Quelques données provenant de l'expérimentation détaillée dans le chapitre suivant seront utilisées pour illustrer les activités de ces modules.

Le quatrième chapitre présente dans un but de première validation, un cas d'utilisation de l'environnement EISEval. Il prend la forme d'une expérimentation qui a été réalisée au sein de notre laboratoire LAMIH avec dix sujets. D'abord, les trois sous-systèmes du système interactif se situant dans le contexte lié au projet SART sont introduits. Ce système aident les régulateurs en salle de contrôle à accomplir leur tâche en mode normal et en mode dégradé de fonctionnement du réseau du trafic. Ensuite, un de ces trois sous-systèmes, le système interactif à base d'agents SAI (Système d'Aide à l'Information) – un système d'aide à

l'information voyageurs du transport public terrestre Valenciennois (Tram, Bus) est présenté plus en détails. Puis, la configuration de déploiement, le scénario de l'expérimentation et la démarche d'évaluation d'un système interactif en utilisant l'environnement EISEval sont successivement décrits. Pendant cette expérimentation, l'environnement a été appliqué pour évaluer le système SAI. Cette expérimentation a permis à l'évaluateur de critiquer le SAI et de proposer les améliorations nécessaires à son concepteur. Enfin, les résultats d'évaluation sont détaillés. Ils prennent la forme de problèmes détectés liés au système SAI, suite à son utilisation par les dix sujets, ainsi que d'améliorations suggérées par l'évaluateur au concepteur du SAI. L'évaluateur effectue ses activités d'évaluation à l'aide plus particulièrement d'un des modules de l'environnement EISEval. Dans le cas de cette expérimentation, nous avons joué le rôle de l'évaluateur.

Dans le dernier chapitre, un ensemble de perspectives de recherche seront exposées. Ces perspectives sont relatives au système SAI, aux modules de l'environnement EISEval et à l'environnement intégré plus global nommé IEISEval (Integrated Environment for Interactive System Evaluation) que nous visons. Ces perspectives visent l'amélioration du système SAI et de quelques modules de l'environnement EISEval pour augmenter l'automatisation de celui-ci. Dans ce chapitre, nous proposons un environnement intégré plus global permettant de combiner différentes méthodes (y compris l'environnement EISEval actuel) pour l'aide à l'évaluation des systèmes interactifs.

Chapitre 1 : Architecture des systèmes interactifs

Sommaire

1. Introduction

La conception d'architecture des systèmes interactifs n'est pas un nouveau sujet de recherche dans le domaine de l'Interaction Homme-Machine (IHM). Elle a pour objectif d'assister les développeurs dans la conception, le développement et la validation des systèmes interactifs. Depuis une vingtaine d'années, plusieurs modèles d'architecture sont proposés [Bass *et al*., 1991 ; Coutaz, 1987 ; Goldberg, 1983 ; Pfaff, 1985 ; Tarpin-Bernard et David, 1999], etc., pour aider les concepteurs à construire les applications interactives. D'après [Coutaz et Nigay, 2001], un modèle d'architecture est défini par :

- des composants qui sont les *unités d'abstraction* dont la nature dépend de la structure considérée dans le processus de conception architecturale (par exemple, un composant peut être un service, un module, une bibliothèque, un processus, une procédure, un objet, une application, etc.) ;
- les propriétés extérieurement visibles d'un composant définissant ses *comportements attendus* (par exemple, les services fournis, les ressources requises, ses performances, ses mécanismes de synchronisation, etc.) ;
- et les relations que ces composants entretiennent. Le comportement observable de chaque composant doit faire partie de l'architecture parce qu'il détermine les interactions possibles de ce composant avec d'autres composants.

D'après cette définition, un système peut avoir plusieurs structures correspondant chacune à un point de vue, donc à une finalité ou classe de problèmes précis à résoudre. Un logiciel peut être représenté par plusieurs plans et schémas destinés chacun à un corps de métier et dépendants de l'étape du processus de développement, comme en architecture du bâtiment. [Coutaz et Nigay, 2001].

En général, les modèles proposés respectent le principe de séparation entre les deux parties d'une application interactive : la partie « interface » qui est en contact direct avec l'utilisateur et la partie « application » qui concerne le noyau fonctionnel [Coutaz et Nigay, 2001 ; Dragicevic, 2004 ; Texier, 2000]. Cette séparation est utile pour le développement et l'amélioration des applications, et on peut modifier une partie sans affecter l'autre. Enfin, on considère que la flexibilité et la maintenabilité des applications sont en conséquence augmentées. Malgré ce point commun, la différence entre les modèles est claire. En effet, nous distinguons deux grands types de modèles d'architecture : les modèles *fonctionnels* et les modèles *structurels*. Ceux-ci sont successivement décrits.

2. Modèles d'architectures fonctionnels

Les modèles d'architecture fonctionnels visent à décomposer un système interactif en plusieurs composants fonctionnels indépendants. Nous présentons ici deux modèles très connus : le modèle Seeheim [Pfaff, 1985] et ARCH [Bass *et al*., 1991].

2.1. Modèle d'architecture Seeheim

La figure 1.1 illustre le modèle d'architecture dit de Seeheim faisant suite à un workshop s'étant tenu dans cette ville. Ce modèle décompose un système interactif en trois composants logiques :

- Le composant *présentation* est en contact direct avec l'utilisateur. Ce composant peut, d'une part, traduire les actions provenant de l'utilisateur (clic souris, touches du clavier, parole, joystick, etc.) en langage machine, puis, les transférer au composant *contrôleur de*

dialogue. D'autre part, il peut présenter les sorties perceptibles par l'utilisateur (affichage graphique ou textuel, sons, vibrations de joystick, etc.)

- Le composant *Interface avec l'Application* traduit les sorties du composant *Contrôleur de Dialogue* en langage spécifique pour dialoguer directement avec l'application, il peut recevoir de l'information de l'application pour la transférer au composant *Contrôleur de Dialogue*.
- Le composant *Contrôleur de Dialogue* joue le rôle d'intermédiaire entre les deux autres composants *Interface avec l'Application* et *Présentation* qui sont indépendants.

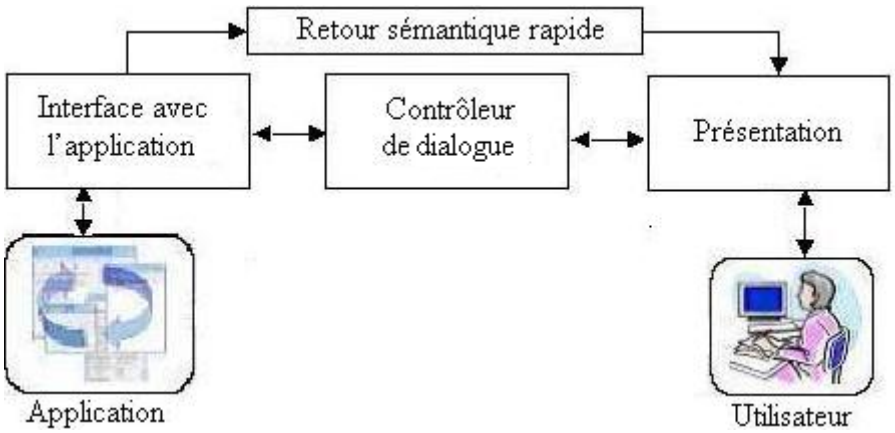

Figure 1.1 : Modèle d'architecture Seeheim [Pfaff, 1985]

Le *retour sémantique rapide* est utilisé pour que les composants *Présentation* et *Interface avec l'Application* puissent dialoguer entre eux directement, par exemple, si l'utilisateur exécute l'opération de glisser-déposer, alors il est utile pour modifier instantanément l'apparence de l'icône cible pour indiquer si l'opération est valide. Dans ce cas, le composant *Présentation* peut accéder directement aux informations sémantiques du noyau fonctionnel du composant *Interface avec l'Application*.

Ce modèle a inspiré des variantes. Les lecteurs intéressés peuvent consulter [Karsenty et Weikart, 1991] pour le modèle de Seeheim étendu, [Dance et *al*., 1987] pour le modèle de Seeheim modifié ou [Hudson, 1987] pour prendre connaissance de la variante proposée par cet auteur.

2.2. Modèle d'architecture ARCH

La figure 1.2 illustre le modèle d'architecture ARCH. Nous pouvons considérer le modèle ARCH comme un raffinement du modèle de Seeheim. En effet, deux composants intermédiaires sont ajoutés aux composants de base de Seeheim pour créer le modèle ARCH :

- Le composant *Adaptateur du domaine* gère le lien entre le composant *Contrôleur de dialogue* et le composant *Noyau fonctionnel*. L'objectif est d'assurer l'indépendance entre ces deux composants. Ce composant *Adaptateur du domaine* est utile pour régler les différences de modélisation des objets conceptuels entre le composant *Contrôleur de dialogue* et le composant *Noyau fonctionnel*. Les *objets du domaine* sont des données qui concernent directement ou indirectement le composant *noyau fonctionnel* ; par exemple, il peut s'agir du résultat d'une fonction du noyau fonctionnel ou du résultat d'une requête ou d'une interrogation dans une base de données.

- Le composant *Présentation* se charge du lien entre le composant *Interaction* et le composant *Contrôleur de dialogue.* Ce composant fournit au composant *Contrôleur de dialogue* les objets de présentation conceptuels indépendants de toutes les boîtes à outils pour assurer l'indépendance entre le composant *Contrôleur de dialogue* et le composant *Interaction* qui est responsable des interactions physiques avec l'utilisateur au moyen des interacteurs (*widgets*) d'une boite à outils spécifique. Cette indépendance permet aux concepteurs de choisir en toute liberté les composants *Interaction* proposés à l'utilisateur. Les *objets d'interaction* sont des instances spécifiques à la boîte à outils choisie et ils utilisent des techniques d'interaction et de visualisation spécifiques. Les *objets de présentation* sont des objets d'interaction conceptuels décrivant les événements produits par l'utilisateur et les données qui lui sont présentées ; ils sont indépendants des boîtes à outils, des méthodes d'interaction et de visualisation.

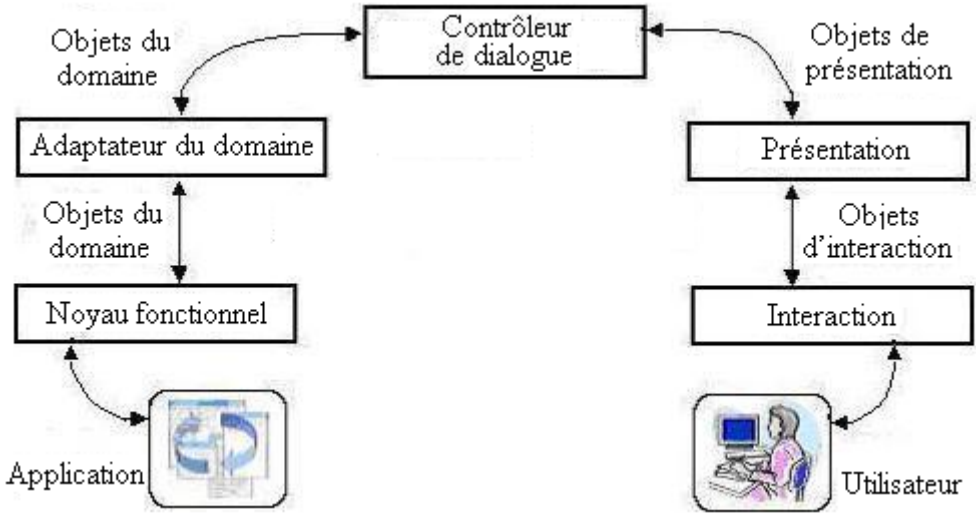

Figure 1.2 : Modèle d'architecture ARCH [Bass et al., 1991]

Ce modèle d'architecture ARCH possède une séparation plus claire entre les composants que dans le cas du modèle Seeheim mais il impose aussi des transitions d'information plus lourdes entre les composants parce que ces dialogues exigent les analyses, les transformations nécessaires des informations échangées entre les composants.

Les lecteurs intéressés par une variante de ce modèle peuvent consulter [UIMS, 1992] pour le méta-modèle *Slinky* inspiré du nom de ce jouet. Dans ce modèle, les fonctionnalités du système interactif peuvent glisser d'un composant de l'ARCH à l'autre en fonction des objectifs des développeurs. Par exemple, on peut faire transiter des valeurs de données directement entre le composant *noyau fonctionnel* et le composant *présentation* sans passer, ni par le composant *adaptateur du domaine*, ni par le composant *contrôleur de dialogue*, et ceci afin de satisfaire la contrainte de temps d'exécution si nécessaire [UIMS, 1992].

3. Modèles d'architecture structurels

Les modèles d'architecture structurels fournissent une décomposition avec un niveau de granularité beaucoup plus fin que les architectures fonctionnelles. Ils visent à regrouper les fonctions dans des entités autonomes et coopératives. Les architectures structurelles sont construites avec des entités respectant le principe de composition ou de communication ; on n'y retrouve pas de décomposition fonctionnelle comme dans les architectures fonctionnelles.

Nous présentons successivement trois modèles d'architecture très connus : PAC [Coutaz, 1987, 1990], MVC [Goldberg, 1983] et AMF [Tarpin-Bernard et David, 1999].

3.1. Modèle d'architecture PAC

La figure 1.3 illustre le modèle d'architecture PAC (Présentation, Abstraction, Contrôle) proposé par J .Coutaz [Coutaz, 1987, 1990].

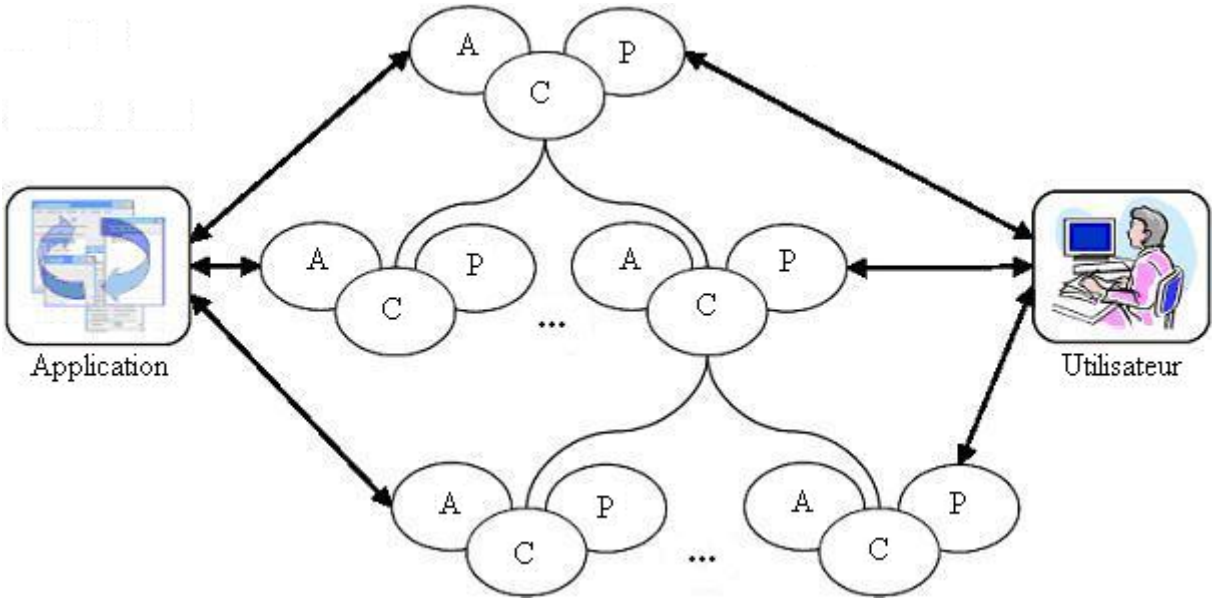

Figure 1.3 : Modèle d'architecture PAC [Coutaz, 1987]

Le modèle d'architecture PAC décompose une application interactive en une hiérarchie d'entités appelées agents, et ceci de manière récursive. Chaque agent est composé de trois composants appelés facettes :

- La facette *Présentation*, connectée aux dispositifs d'entrée et de sortie. Cette facette est responsable de la transmission des informations à l'utilisateur à travers les dispositifs de sortie (affichages graphiques et textuels sur l'écran, sons du haut-parleur, vibrations du joystick, etc.) et de recevoir les commandes de l'utilisateur à travers les dispositifs d'entrée (souris, touches du clavier, parole, joystick, etc.). La présentation d'une application interactive correspond à l'union des présentations des agents de cette application.
- La facette *Abstraction* est responsable du noyau fonctionnel de l'application. Cette facette permet d'exécuter les fonctions spécifiques au domaine de l'application.
- La facette *Contrôle* assure d'une part, le dialogue entre les facettes *Abstraction* et *Présentation*. D'autre part, elle permet le dialogue entre les agents PAC de l'application.

Les lecteurs intéressés par les variantes du modèle d'architecture PAC peuvent consulter [Calvary *et al*., 1996] pour le modèle PAC+3, qui est un modèle d'architecture générique pour les systèmes multi-utilisateurs, ou [Daassi *et al.*, 2003] pour le modèle REPAC qui permet le raffinement de la facette présentation en un agent PAC pour une meilleure structuration de la facette présentation.

3.2. Modèle d'architecture MVC

La figure 1.4 illustre le modèle d'architecture MVC (Modèle, Vue, Contrôleur).

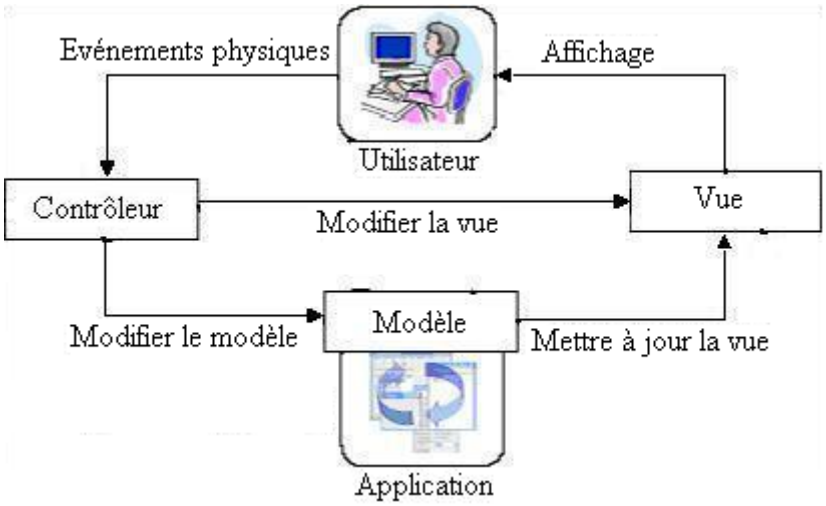

Figure 1.4 : Trois facettes du modèle d'architecture MVC [Goldberg, 1983]

Le modèle d'architecture MVC a été proposé pour implémenter les applications en langage Smalltalk. Dans ce modèle, trois facettes relativement indépendantes communiquent par envoi de messages, ce qui est conforme à l'idée de Smalltalk :

- *Modèle* : cette facette représente le noyau fonctionnel de l'application ainsi que les données métier. Le *Modèle* possède une liste des facettes *Vue* (cf. ci-dessous) enregistrées dans le *Modèle* pour recevoir la notification lors du changement du *Modèle*. Si l'état interne de la facette *Modèle* est modifié, alors les facettes *Vue* concernées sont notifiées pour une mise à jour ; la cohérence entre ces deux facettes est ainsi assurée.
- *Vue* : cette facette représente un affichage de la facette *Modèle* à l'utilisateur. En général, cette facette est composée des objets graphiques ou textuels qui sont perceptibles par l'utilisateur. Comme précisé ci-dessus, cette facette est toujours notifiée et mise à jour lorsque l'état interne de la facette *Modèle* est changé.
- *Contrôleur* : cette facette reçoit les actions provenant de l'utilisateur à travers les dispositifs d'interaction comme la souris ou le clavier. Cette facette interprète ces événements utilisateurs ; puis elle peut modifier :
    - l'état de la facette *Modèle*. Par exemple, si l'utilisateur veut modifier les données métier de l'application, alors cette facette *Contrôleur* contacte directement la facette *Modèle* pour les modifier (comme l'illustre la figure 4 ci-dessus).
    - une facette *Vue* concernée. Par exemple, l'utilisateur peut utiliser les fonctionnalités comme zoomer (zoom avant, zoom arrière) sur l'interface pour obtenir un affichage désiré. Dans ce cas, cette facette *Contrôleur* contacte directement la facette *Vue* concernée pour modifier la présentation de l'interface (comme l'illustre la figure 1.4).

Ce modèle d'architecture MVC peut avoir plusieurs facettes *Contrôleur* ou plusieurs facettes *Vue* qui peuvent se connecter à une même facette *Modèle*. Une facette *Modèle* peut être visualisée sous forme de représentations variées correspondant aux différentes facettes *Vue*. Les lecteurs intéressés par des variantes du modèle MVC peuvent consulter [Girard et Ribette, 2001] pour le modèle d'architecture MVC2 ou [Goschnick et Steding, 2001] pour le modèle ShadowBoard. Le modèle MVP (Model-View-Presenter) dérivé du modèle MVC peut être consulté à http://www.martinfowler.com/eaaDev/uiArchs.html.

3.3. Modèle d'architecture AMF

AMF [Tarpin-Bernard, 1999 ; Ouadou, 1994] est un modèle d'architecture multi-agents et multi-facettes d'une application interactive, influencé par le modèle PAC. Les agents de ce modèle sont organisés hiérarchiquement selon des règles de composition comme dans le modèle PAC. En effet, un agent parent contient un certain nombre d'agents enfants. En conséquence, pour toute application interactive modélisée avec l'architecture AMF, il doit exister un agent racine qui est l'ancêtre commun de tous les agents de l'application ; tous les autres agents de l'application sont les composants directs ou indirects de cet agent ancêtre qui est généralement appelé "application" [Tarpin-Bernard, 1999].

Chaque agent de cette architecture contient des sous-agents (agents enfants), des administrateurs de contrôle et un nombre illimité de facettes illustré par les figures 1.5a, 1.5b. Normalement, on utilise des relations d'emboîtement (figure 1.5a) pour illustrer les relations entre les d'agents. Si les schémas sont trop complexes, et donc difficiles à lire, alors on utilise une représentation relationnelle sous forme de graphes (figure 1.5b) pour les illustrer.

Chaque facette est un composant contenant l'ensemble des données et traitements relevant d'une même thématique fonctionnelle ; par exemple, la facette *Présentation* contient les éléments concernant l'interface utilisateur, la facette *Abstration* contient les éléments concernant le noyau fonctionnel de l'application, la facette *Contexte* contient les éléments concernant le contexte, la facette *droit d'utilisateur* contient les éléments concernant les droits d'utilisateur, etc. [Tarpin-Bernard et David, 1999]. Dans les autres architectures, chaque agent contient un nombre fini de facettes (trois pour PAC) mais dans cette architecture, le nombre de facettes est, en théorie, illimité. La décomposition du modèle AMF est plus fine que PAC. Les figures 1.5a, 1.5b illustrent la structure d'un agent du modèle d'architecture AMF.

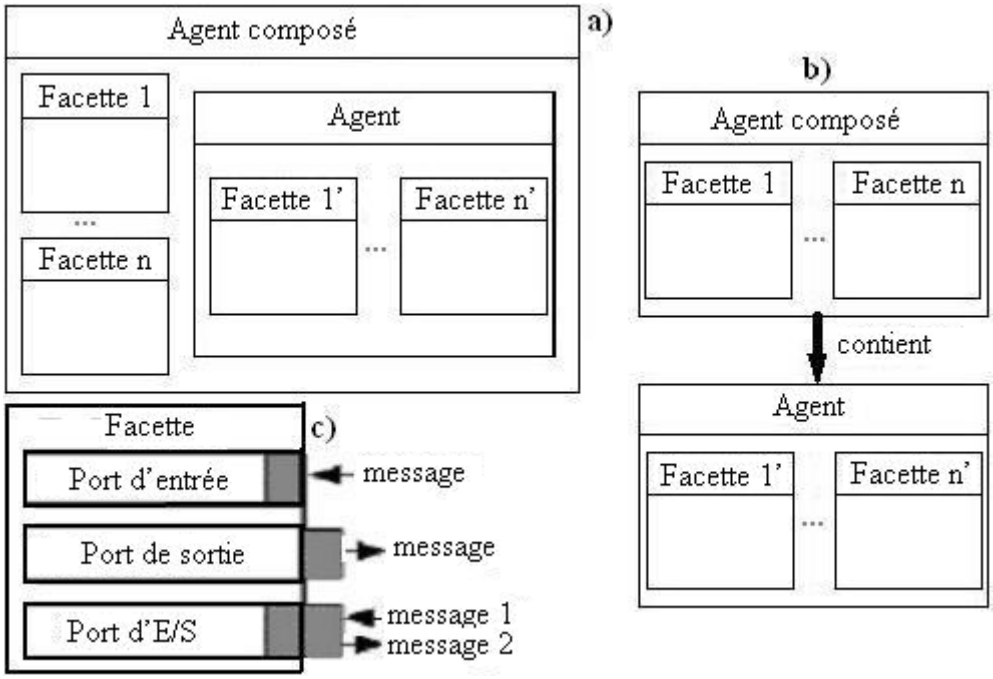

Figure 1.5 : a-b) Structure d'un agent. c) Une facette (avec ses ports) de l'architecture AMF [Tarpin-Bernard et David, 1999]

Au niveau Facette, chaque facette contient les ports de communication associés aux services de la facette. Il y a trois types de ports : d'entrée définissant le service offert, de sortie

définissant le service nécessaire et port d'entrée/sortie combinant les deux. La figure 1.5c illustre une facette avec ses ports.

Chaque facette est associée à une classe applicative dont certaines fonctions appelées "démon" sont associées aux ports de cette facette. La figure 1.6a illustre les ports et les démons associés.

La figure 1.6b matérialise le fait que la facette A peut invoquer le service f de la facette B à travers un administrateur de contrôle. Par exemple, dans une application rudimentaire de gestion d'un agenda qui est modélisée avec l'architecture AMF et qui permet à l'utilisateur de gérer ses rendez-vous (RVs), l'utilisateur peut créer, modifier, déplacer et supprimer les RVs. Lorsque l'utilisateur veut supprimer un RV déjà créé, il peut déclencher une activation du port de sortie intitulé "Supprimer un RV" de la facette *Présentation*, en utilisant la souris ou le clavier, via l'interface graphique de l'application. Ce port de sortie va activer, à travers un administrateur de contrôle, la portie d'entrée de même nom de la facette *Abstraction*. Le démon associé de ce port va supprimer ce RV de la liste des RVs de l'application.

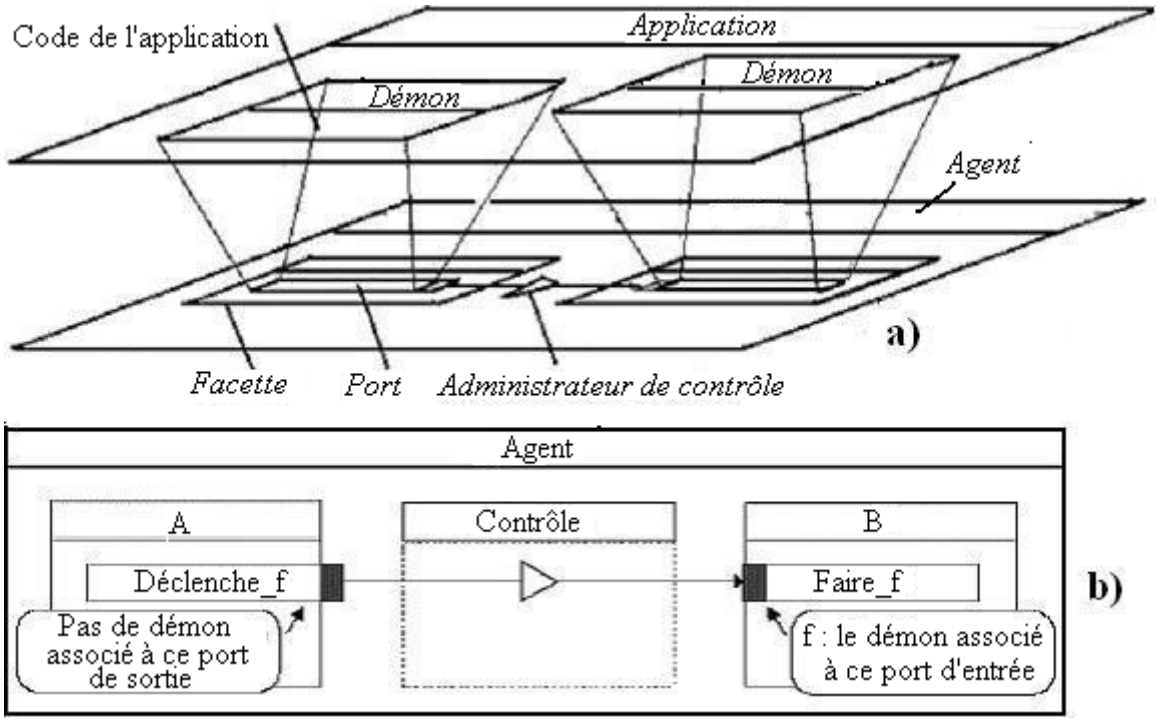

Figure 1.6 : a) Infrastructure de l'architecture AMF et b) Interaction entre les ports des facettes de l'architecture AMF [Tarpin-Bernard et David, 1999]

Au niveau Agent, les administrateurs sont responsables de l'interaction entre les ports de communication des facettes qui appartiennent à un même agent ou aux différents agents. D'après [Tarpin-Bernard et David, 1999], les formalismes de modélisation des composants Contrôle sont rares et généralement limités mais ces composants constituent la clé de voûte des modèles d'architecture et il est indispensable de fournir un outil de modélisation performant. Les administrateurs de contrôle de l'architecture AMF sont standardisés et jouent différents rôles. Un administrateur de contrôle joue trois rôles, selon [Tarpin-Bernard et David, 1999] :

- un **rôle de connexion**. Ce rôle consiste à gérer les relations logiques qui peuvent exister entre les ports qui lui sont attachés (sources => cibles) ;

- un **rôle de traduction**. Ce rôle consiste à transformer les messages émis par les ports sources en messages compréhensibles par les ports cibles. Dans ce rôle, on distingue plusieurs types d'administrateurs :
    - Un administrateur de transfert transporte directement les données issues du port source vers le port cible sans modification.
    - Un administrateur de conversion transforme les données issues du port source en données compréhensibles par le port cible, par exemple, pour la conversion d'un entier en chaîne de caractères.
    - Un administrateur d'assemblage rassemble les données issues des ports sources pour les transporter sous une forme structurée vers le port cible.
    - Un administrateur de calcul arithmétique applique des opérations sur les données des ports sources pour calculer la donnée à envoyer au port cible.
    - Un administrateur de traitement applique un algorithme sur les données des ports sources pour obtenir la donnée à transmettre au port cible.
- un **rôle comportemental**. Ce rôle exprime les règles d'activation de l'administrateur, c'est-à-dire sous quelle condition et à quel moment les messages émis par les ports sources seront transmis aux ports cibles. Dans ce rôle, il y a aussi quelques types d'administrateurs :
    - Un administrateur simple représente une relation orientée entre un port source A et un port cible B. Ce type d'administrateur correspond à cette assertion : si A est activé alors activer B.
    - Un administrateur itératif indique la nécessité d'avoir une répétition de n activations du port source avant d'avertir le port cible.
    - Un administrateur de séquence permet de représenter :
        - Soit une relation d'attente d'activation entre plusieurs ports sources A1, A2…An et un port cible B. Ce type d'administrateur correspond à l'assertion suivante : Si A1 puis A2 puis ... An alors B,
        - Soit une relation inverse. Ce type d'administrateur correspond à l'assertion suivante : Si A1 alors B1 puis B2 puis ... Bn
    - les administrateurs disjonctifs et conjonctifs décrivent respectivement une relation « OR » (si A1 ou A2 ou ... ou An alors B) et une relation « AND » (si A1 et A2 et ... et An alors B)

Un administrateur réel est une combinaison de ces administrateurs élémentaires (comportementaux et traducteurs) qui permet la connexion entre des ports de communication [Tarpin-Bernard et David, 1999].

4. Discussion sur les architectures fonctionnelles et structurelles

Les architectures *fonctionnelles* décomposent un système interactif en plusieurs composants *fonctionnels* indépendants qui permettent une séparation explicite entre la partie *Application* et la partie interface (*Présentation*). L'intérêt de ces modèles est de pouvoir modifier une partie sans en affecter une autre. Les architectures fonctionnelles facilitent aux concepteurs une stratégie d'analyse par décomposition d'un grand système en modules différents mais elles ont aussi quelques inconvénients ; par exemple, la structure interne des composants ainsi que les dialogues entre eux ne sont pas décrits. Ce problème est géré

entièrement par les concepteurs du système. En plus, les composants des architectures fonctionnelles sont trop macroscopiques [Tarpin-Bernard et David, 1999]. D'après [Trabelsi *et al*., 2004], les architectures fonctionnelles sont utiles en tant que cadre de pensée pour une analyse des systèmes interactifs. A cause de ces inconvénients, les architectures fonctionnelles nous semblent insuffisantes pour les systèmes interactifs complexes en général et les systèmes de supervision (cf.ci-dessous) dans le contexte industriel en particulier [Trabelsi, 2006], qui nous intéressent plus particulièrement dans le cadre de cette thèse.

Les architectures structurelles permettent la description de leurs entités ainsi que la communication entre elles. En plus, ces architectures fournissent une décomposition avec un niveau de granularité beaucoup plus fin que les architectures fonctionnelles. Ces architectures nous semblent donc plus adaptées à la conception des systèmes interactifs complexes. En plus, la décomposition d'un système interactif en plusieurs entités autonomes et coopératives peut accélérer l'interaction entre le système interactif et l'utilisateur. Cet avantage est très utile pour les systèmes de supervision parce que les utilisateurs de tels systèmes (opérateurs appelés aussi régulateurs ou superviseurs en salle de contrôle) doivent réaliser des tâches hautement cognitives et exécuter un ensemble d'opérations pour superviser et intervenir sur un processus dynamique dans les cas de dysfonctionnement. Un dialogue intensif et lent entre l'utilisateur et les systèmes hautement interactifs comme les systèmes de contrôle/commande risque de ralentir le système et de diminuer la productivité.[1] A côté de ces avantages, les architectures structurelles possèdent aussi des inconvénients**.** Ainsi, à la différence des architectures fonctionnelles, le rôle et le nombre des agents des architectures structurelles ne sont pas clairement précisés [Coutaz, 1990]. L'interface globale du système interactif peut être difficile à percevoir pour les concepteurs parce que les composants *Présentation* sont distribués sur les différents agents. Ce problème est réglé par les architectures fonctionnelles parce qu'elles fournissent un seul composant, celui de *Présentation.*

Les qualités et les défauts de chaque type d'architecture mènent à la naissance des architectures hybrides qui tentent de tirer profit des caractéristiques de ces deux types d'architectures. Les architectures hybrides proposées visent à décomposer d'une manière structurelle et plus précise, le rôle et le fonctionnement de certains composants des architectures fonctionnelles comme PAC-Amodeus [Nigay, 1994 ; Nigay et Coutaz, 1993] et H^4 [Guittet, 1995].

Le modèle PAC-Amodeus[2], comme l'illustre la figure 1.7, est une hybridation entre le modèle PAC et le modèle ARCH. Ce modèle réutilise les cinq composant du modèle ARCH et affine le composant *Contrôleur de Dialogue* selon une hiérarchie d'agents PAC. Le composant *Adaptateur de domaine* communique directement avec chaque agent PAC du composant *Contrôleur de Dialogue* à travers sa facette *Abstraction* et le composant *Présentation* communique directement avec chaque agent PAC du composant *Contrôleur de Dialogue* à travers sa facette *Présentation*.

Le modèle PAC-Amodeus a pour objectif de combiner les avantages du modèle PAC, qui permet de structurer efficacement le *Contrôleur de dialogue* jusque-là monolithique et les avantages du modèle ARCH, qui intègre des aspects de génie logiciel comme la modifiabilité et la portabilité. Le modèle PAC-Amodeus conserve aussi la décomposition fonctionnelle que le modèle ARCH préconise. Grâce à la décomposition du composant *Contrôleur de dialogue* en agents PAC, il permet aussi de satisfaire les propriétés de qualité du logiciel que sont la

[1] Les utilisateurs intéressés par les systèmes de supervision peuvent consulter une littérature abondante, cf. [Alty, 1995; Alty et Bergan, 1992; Aubry, 1992; Gilmore *et al*., 1989; Gomes, 2003; Johannsen, 1995; Kolski 1997; Le Parc *et al*., 2004; Millot 1996; Moray 1997; Rasmussen 1986; Sheridan 1984, 1985, 1988].

[2] PAC-Amodeus : PAC pour Présentation, Abstraction, Contrôle (Cf. § III.3) et Amodeus pour le projet européen Esprit BRA Amodeus.

modifiabilité et la réutilisabilité, ainsi que les propriétés d'utilisabilité comme le dialogue à fils multiples [Salber, 1995 ; Depaulis, 2002]. Le modèle PAC-Amodeus a été employé dans des application de simulation [Duval et Nigay, 1999] ou pour modéliser des plate-formes multimodales [Nigay and Coutaz, 1995]. Les lecteurs intéressés par ce modèle PAC-Amodeus et les architectures hybrides peuvent consulter [Nigay, 1994 ; Nigay et Coutaz, 1993] pour ses détails.

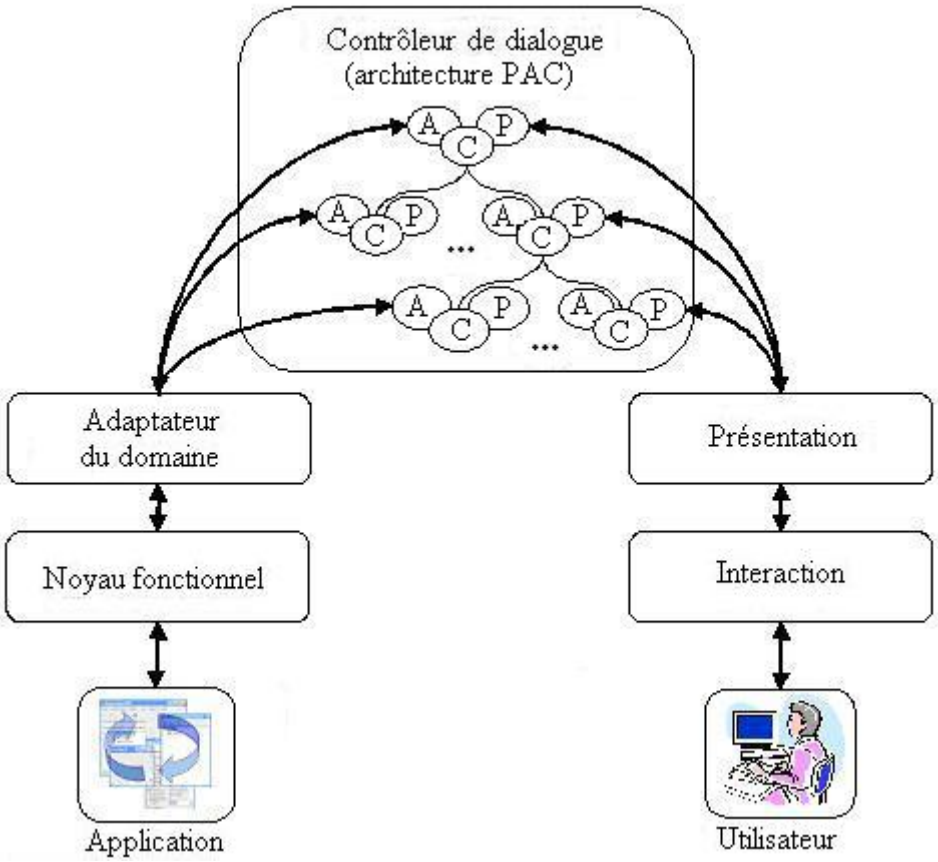

Figure 1.7 : Modèle PAC-Amodeus [Nigay, 1994 ; Nigay et Coutaz, 1993]

Le modèle H^4 [Guittet, 1995] a été développé au laboratoire LISI à Poitiers. Nous utilisons plusieurs travaux de ce laboratoire [Depaulis, 2002 ; Guittet, 1995 ; Patry, 1999 ; Texier, 2000] afin de présenter ce modèle. Les lecteurs intéressés peuvent consulter ces documents pour les détails de ce modèle.

Ce modèle H^4 a pour objectif de prendre en compte les dialogues structurés par lesquels les applications graphiques interactives de conception technique se caractérisent [Guittet, 1995]. Ce modèle correspond au modèle ARCH moyennant quelques modifications comme l'illustre la figure 1.8 :

- La communication directe entre le composant *Adaptateur de Domaine* et le composant *Adaptateur de Présentation* (figure 1.8) est permise. Eventuellement, le composant *Adaptateur de Domaine* est capable de communiquer directement avec le composant *Adaptateur de présentation* pour afficher l'état du composant *Noyau fonctionnel* lorsque le passage par le *Contrôleur de dialogue* n'est pas nécessaire
- Le composant *Contrôleur de dialogue* est structuré et son fonctionnement est décrit d'une manière précise.

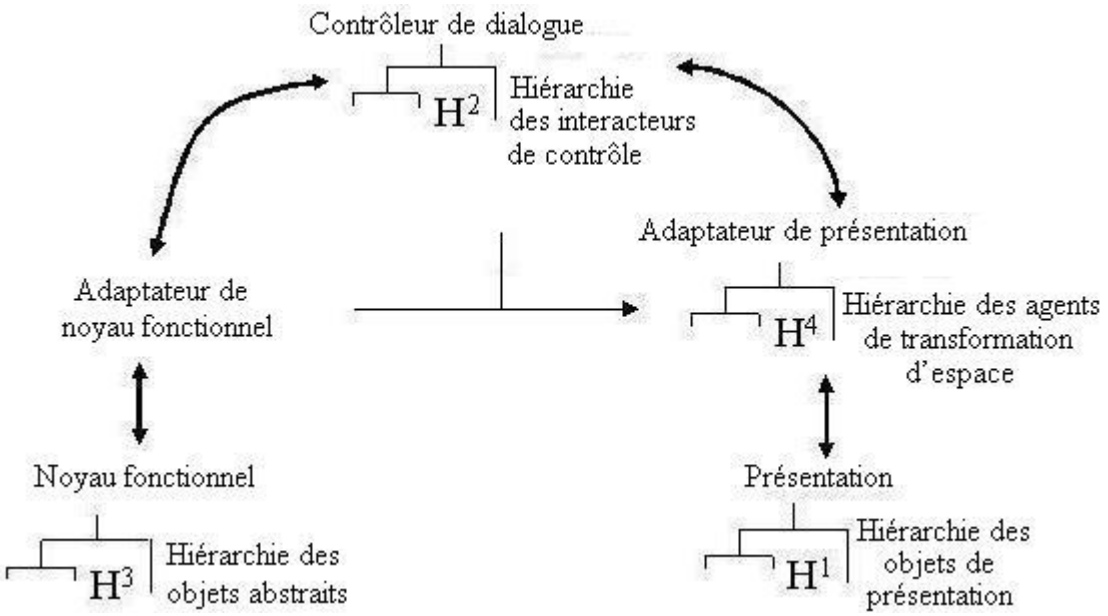

Figure 1.8 : Architecture H4 [Guittet, 1995]

Plus précisément, le composant *Contrôleur de dialogue* est associé à différentes notions (figure 1.9) :

- Les *jetons* qui constituent l'unité d'information. Un *jeton* définit un objet de dialogue et il est une donnée de communication qui circule dans le système. Il y a deux types de *jetons* :
    - Les *jetons paramètres* qui représentent une abstraction des données. Ils peuvent transporter les valeurs (position à l'écran, référence d'un objet du modèle), par exemple, une position graphique donnée par l'utilisateur peut être transportée par un *jeton* « position » stockant son abscisse et son ordonnée. La valeur d'un *jeton* paramètre peut être utilisée par un *interacteur* pour réaliser un appel au composant *Noyau fonctionnel*, via un *questionnaire*. Les éléments *interacteur* et *questionnaire* vont être présentés ci-dessous.
    - Les *jetons commandes*. Un jeton de ce type contient une commande (par exemple, ordre de créer un cercle). Il représente l'intention de l'utilisateur. Un *jeton commande* permet à un *interacteur* de savoir quelle(s) fonction(s) il doit activer, c'est-à-dire qu'il transporte l'identifiant d'une tâche à activer. Un jeton commande n'est donc jamais transmis au composant *Noyau fonctionnel*. Le but d'un *jeton* commande est de placer un *interacteur* dans un état où il attend certains types de jetons paramètres pour pouvoir utiliser le *Questionnaire* associé à la commande. Par exemple, le *jeton* contenant la commande "Centre" sera accepté par *l'interacteur* "Information". Cet *interacteur* attend alors un nouveau *jeton* paramètre référençant un objet du modèle pour pouvoir utiliser le *questionnaire* associé à la commande. A titre d'autre exemple, un *jeton* contenant la commande "Création d'un cercle" sera accepté par *l'interacteur* adéquat, par exemple, *l'interacteur* « formes géométriques ». Ce *jeton commande* place cet *interacteur* dans l'état « Création d'un cercle ». Dans cet état, cet *interacteur* attend seulement un nouveau *jeton paramètre* de type « position, valeur numérique » (qui correspond à la position du centre du cercle futur sur l'écran et à la mesure du rayon de ce cercle) pour pouvoir utiliser le *questionnaire* associé à cette commande « Création d'un cercle ». Si un *jeton paramètre* d'un autre type est transmis à cet *interacteur* qui est dans cet état, alors cet *interacteur* ne va pas

accepter ce jeton. Après que cet *interacteur* reçoive le *jeton paramètre* adéquat, il appelle le *questionnaire* associé en lui transmettant toutes les valeurs de ce *jeton paramètre* pour effectuer cette commande (« Création d'un cercle »). L'état de cet *interacteur* est changé. Il *est replacé* dans l'état précédent où il n'a pas encore reçu le *jeton commande* "Création d'un cercle".

- Les tâches qui sont représentées par des *Questionnaires*. Un *questionnaire* représente une tâche et il est défini par sa signature contenant son nom, ses paramètres d'entrée et, éventuellement, un paramètre de sortie. Les tâches sont implémentées dans le composant *Adaptateur de noyau fonctionnel* et elles constituent le lien entre le composant *Contrôleur de dialogue* et les primitives de l'application (le composant *Noyau fonctionnel*).
- Les *interacteurs*. Les *questionnaires* sont regroupés en *interacteurs*. Un *interacteur* réunit un ensemble de *questionnaires* (qui correspondent à des tâches) et représente ainsi un niveau d'abstraction des tâches. Les *questionnaires* d'un *interacteur* sont considérés à même niveau d'abstraction.
- Le *Moniteur* qui est responsable d'organiser les *interacteurs* de manière hiérarchique (donc on a une organisation hiérarchique des *questionnaires* ou des tâches). En effet, les *interacteurs* de plus bas niveau d'abstraction sont placés sous ceux de niveau d'abstraction plus élevé. Le *Moniteur* organise aussi le passage des *jetons* entre les différents *interacteurs*. Il récupère les *jetons* du composant *Adaptateur de présentation* et les transmet successivement à tous les *interacteurs*, en commençant par celui de plus bas niveau d'abstraction.

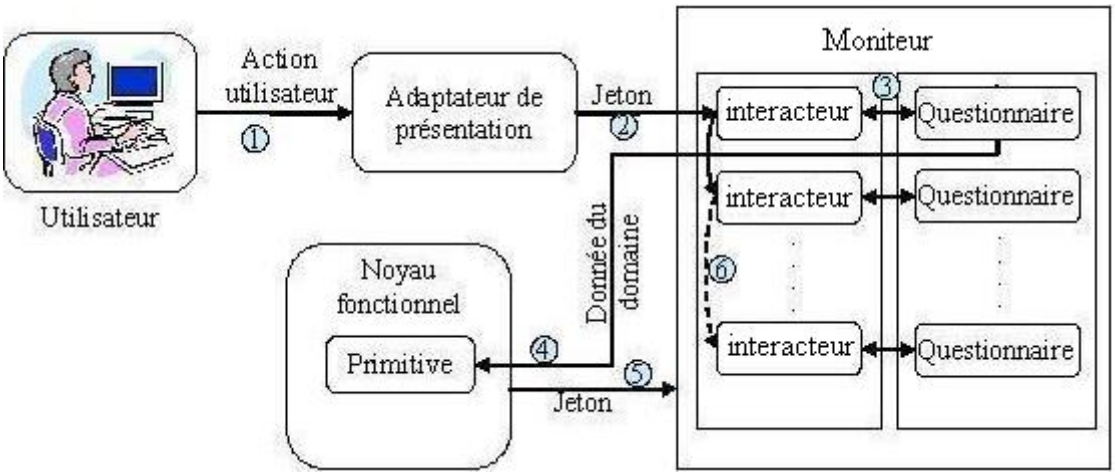

Figure 1.9 : Fonctionnement du modèle H4 [Guittet, 1995]

Le fonctionnement du modèle H^4 est décrit sur la figure 1.9, selon [Depaulis, 2002] :

- Quand l'utilisateur effectue une action (1 sur la figure 1.9), le composant *Adaptateur de présentation* transforme la donnée reçue du composant *Présentation* en un *jeton* contenant une valeur ; par exemple, un clic de souris peut être transformé en un *jeton* de type « position » contenant les coordonnées du point de l'écran sur lequel l'utilisateur a cliqué.
- Ce *jeton* est transmis au *Moniteur* qui le présente au premier *interacteur* de sa hiérarchie (2 sur la figure 1.9). Eventuellement, cet *interacteur* peut accepter ce *jeton* ou non. Si cet interacteur ne se trouve pas dans un état adéquat pour accepter ce *jeton*, alors il rend ce *jeton* au *Moniteur* qui le propage à l'*interacteur* suivant dans la

hiérarchie des *interacteurs*, et ainsi de suite jusqu'à ce qu'il soit accepté par un *interacteur* ou qu'il n'y ait plus d'*interacteur*.

- Quand un *interacteur* se trouve dans un état adéquat et qu'il accepte ce *jeton*, alors cet *interacteur* appelle le *questionnaire* associé en lui transmettant toutes les valeurs stockées par ce *jeton* (3 sur la figure 1.9).
- Ce *questionnaire* associé est ensuite chargé de faire le lien avec le composant *Noyau fonctionnel* en transformant les *jetons* en données du domaine et en appelant la primitive associée (4 sur la figure 1.9).
- Si cette primitive associée produit un résultat, la valeur calculée est à son tour transformée en *jeton* ; puis elle est rendue au *Moniteur* (5 sur la figure 1.9) qui se charge de la transmettre aux *interacteurs* (6 sur la figure 1.9) suivants dans la hiérarchie.

Malgré leurs avantages, les modèles hybrides possèdent les défauts. Le modèle PAC-Amodeus reste imprécis sur l'identification des agents du *Contrôleur de Dialogue* [Depaulis, 2002]. Le modèle H^4 permet la structuration de dialogue avec un *Contrôleur de dialogue* assez détaillé et complexe en introduisant les notions de *jeton*, *moniteur*, *interacteur* et *questionnaire* mais l'implémentation de ce modèle est une tâche fastidieuse et difficile. Son utilisation devient un handicap quand il s'agit d'interface hautement interactive [Depaulis, 2002].

Les lecteurs intéressés par les architectures hybrides peuvent consulter [Fekete, 1996] pour le modèle MultiCouche, [Salber, 1995] pour le modèle CoPAC qui est une extension du modèle PAC-Amodeus pour les systèmes multi-utilisateurs et la communication homme-homme médiatisée par l'ordinateur (computer-mediated commucation), ou [Calvary *et al.*, 1997] pour le modèle PAC* qui est la version collaborative du modèle PAC-Amodeus dédié aux systèmes multi-utilisateurs et plus particulièrement au TCAO (Travail Coopératif Assisté par Ordinateur).

Afin d'exploiter les avantages de ces deux types d'architectures fonctionnelles et structurelles, nous nous basons sur une architecture mixte à base d'agents qui emprunte les principes de ces deux types d'architecture, proposée lors de travaux précédents au laboratoire.

5. Positionnement des travaux par rapport à une architecture à base d'agents logiciels

5.1. Agents et systèmes multi-agents (SMA) : concepts de base

Un système multi-agents (SMA) est un ensemble d'entités qui coordonnent leurs buts, expériences, connaissances et plans pour agir ou résoudre des problèmes, y compris le problème de la coordination inter-agents lui-même [Adam, 2000 ; Ferber, 1995 ; Mandiau et Grislin-Le Strugeon, 2001].

D'après [Bond et Gasser, 1988], un système multi-agent est constitué d'un ensemble d'entités coopérantes capables de communiquer et de coordonner leurs comportements afin d'atteindre un objectif commun.

Une particularité de systèmes multi-agent réside dans les critères retenus pour la répartition des compétences et des connaissances (ou données) entre agents. Les critères de distribution peuvent être multiples [Grislin-Le Strugeon et Péninou, 1998] tels que : le critère de distribution naturelle du problème (spatiale, fonctionnelle, temporelle, etc.), celui de distribution de connaissances (les systèmes multi-experts dont les expertises peuvent

s'imbriquer), les critères à base logicielle afin de construire des modules à couplage faible et à cohérence forte dont le développement et la maintenance sont plus faciles, les critères technologiques (par exemple, concernant l'interconnexion des processeurs), etc.

Dans un système multi-agent, chaque agent a son propre chemin d'exécution, les chemins d'exécution des agents sont différents [Adina *et al*., 2002 ; Ferber, 1991] mais ils doivent avoir un élément commun au moins, par exemple, le but visé ou un environnement commun au minimum. Les interactions entre agents d'un SMA peuvent être de type coopératif (les agents coopèrent pour atteindre un but commun), conflictuel (les agents poursuivent des buts différents et conflictuels) ou d'un type hybride [Chaib-Draa *et al*., 2001 ; Mandiau et Grislin-Le Strugeon, 2001]. L'interaction entre les agents est aussi une problématique de recherche. Les lecteurs intéressés par les types des structures d'organisation et les patterns de coordination entre agents du SMA peuvent consulter [Aknine, 2000 ; Lee et Hwang 2003 ; Wood et Barbacci 1999].

Il existe beaucoup de définitions du terme « agent » [Bradshaw, 1997 ; Ferber, 1995 ; Franklin et Graesser, 1996 ; Gasser, 1989 ; Logan, 1998 ; Luck et d'Inverno, 2001 ; Wooldridge et Jennings, 1995]. Ces définitions varient en fonction des applications visées ou des problématiques de recherche [Mandiau et Grislin-Le Strugeon, 2001]. Mais en général, les chercheurs s'accordent sur les propositions suivantes « un agent est n'importe quoi qui peut percevoir son environnement à travers les capteurs et qui peut agir à travers les effecteurs » [Russell et Norvig, 1995] ou « Un agent agit en fonction d'un ensemble d'informations reçues et perçues de son environnement et en fonction des buts qu'il poursuit » [Ferber, 1995].

D'après ces définitions, l'agent perçoit l'environnement et effectue les actions qui peuvent affecter et changer l'environnement. Par exemple, l'environnement d'un agent humain est le monde réel. Cet agent humain a cinq sens différents (ouïe, vue, odorat, goût, toucher) comme détecteurs et les membres, parole, etc., comme effecteurs. Un agent robotique peut être équipé de détecteurs comme un odomètre mesurant ses changements de position, un sonar (SOund NAvigation and Ranging) fournissant les informations concernant les objets proches, par exemple dans le but de les éviter, etc ; il peut être équipé d'effecteurs pour convertir les commandes numériques (« software commands ») en mouvements physiques comme les « *locomotions* » (changements de position du robot dans l'environnement), les « *manipulations* » (déplacements d'objets du robot dans l'environnement), etc. La figure 1.10 illustre un agent.

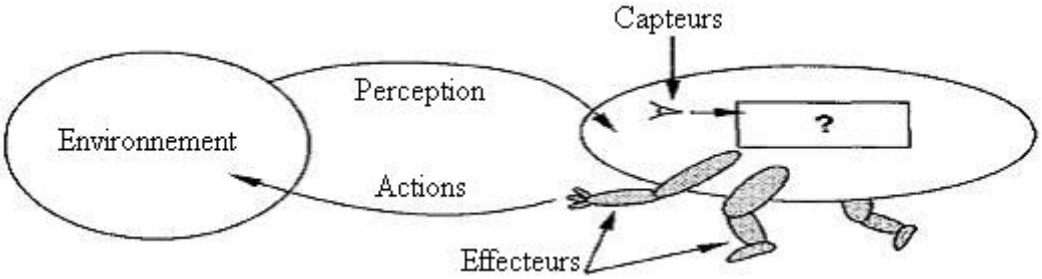

Figure 1.10 : Illustration d'un agent qui perçoit et agit [Russell et Norvig, 1995]

Quand on veut concevoir un agent, il faut avoir une idée de ses actions, de ses perceptions possibles, du but (que l'agent doit poursuivre), de la performance (que l'agent doit satisfaire), et de l'environnement (dans lequel cet agent va opérer) [Russell et Norvig, 1995]. Chaque type d'agent opère dans son propre environnement, par exemple :

- L'environnement d'un agent robotique est constitué des objets du monde réel.

- L'environnement d'un système de diagnostic médical comprend l'hôpital et les patients.
- L'environnement d'un agent logiciel peut être les réseaux d'ordinateurs (Internet, Intranet), les plateformes des machines (logiciels, matériels). Les perceptions et actions de cet agent sont codées sous forme des chaînes des bits.
- L'environnement des agents d'une application interactive utilisateur peut être l'utilisateur qui interagit avec ces agents via une interface graphique, par exemple, l'environnement d'un système « Interactive English tutor » est les étudiants qui doivent saisir les mots, les phrases en anglais afin de faire les exercices fournis par cette application.
- Un agent peut communiquer avec les autres agents, donc ces derniers font partie de l'environnement d'un agent
- L'environnement peut être la combinaison de tous les éléments mentionnés ci-dessus [Genesereth et Ketchpel, 1994].

D'après [Chaib-Draa *et al*., 2001 ; Mandiau 2000], un agent passe par trois étapes pour réaliser une tâche :

- L'agent perçoit tout ou partie de son environnement pour mettre à jour ses propres représentations de celui-ci ;
- L'agent détermine quelles actions doivent être exécutées et comment les exécuter ;
- L'agent exécute les actions choisies qui peuvent affecter et changer l'environnement.

Un agent exécute les actions en se basant sur sa perception de l'environnement ainsi que ses buts [Ferber 1995 ; Russell et Norvig 1995]. L'état d'agent de connaissance doit être toujours mis à jour en se basant sur sa perception de l'environnement. Un agent possède généralement quatre propriétés clés :

- **L'autonomie** : les agents peuvent agir sans intervention directe de l'utilisateur et des autres agents. Ils peuvent contrôler leurs actions et états internes [Castelfranchi, 1995].
- **La sociabilité** qui permet de communiquer avec d'autres agents [Genesereth and Ketchpel, 1994]. Les agents peuvent s'engager dans des interactions complexes variées incluant la coopération, la négociation ou la compétition avec d'autres agents.
- **La réactivité** par rapport aux changements de son environnement [Wooldridge et Jennings, 1995].
- **Proactivité** en ce qui concerne ses buts individuels. Chaque agent agit d'une façon pro-active pour atteindre ses buts [Wooldridge et Jennings, 1995].

On distingue généralement les agents dans le domaine de l'intelligence artificielle (agent IA) de ceux dans le domaine de l'interaction homme-machine (agent IHM). Des agents IA sont dotés de mécanisme de planification et ils poursuivent des buts de manière adaptative [Coutaz et Nigay, 2001]. Ces agents IA peuvent coopérer pour résoudre des problèmes complexes avec un niveau d'intelligence éventuellement élevé. Un agent IHM (dit agent réactif) peut être considéré comme une entité dont les comportements sont les conséquences des observations et interactions de cet agent avec l'utilisateur [Coutaz et Nigay, 2001]. Les comportements de tels agents peuvent être décrits par leurs interactions (directes ou indirectes à travers les autres agents) avec l'utilisateur [Chalon, 2004]. Un agent au sens de l'IHM peut être seulement une petite fenêtre, un bouton simple avec une intelligence très limitée. Evidemment, ces deux types d'agents peuvent être combinés : un agent IHM peut être construit avec l'intelligence de l'agent IA pour constituer des systèmes auto-adaptatifs à

l'utilisateur [Tarpin-Bernard et David, 1999] (on peut par exemple constituer des IHM dites plastiques qui peuvent s'adapter automatiquement aux contextes[3]) ou pour constituer des interfaces intelligentes à base d'agents qui peuvent agir comme des assistants de l'utilisateur.

D'après [Coutaz et Nigay, 2001], un agent est considéré comme un système de traitement de l'information avec un ensemble d'actions qu'il peut exécuter, avec des mécanismes d'entrée/sortie et la capacité à représenter les états (appelé vecteur d'état).

Les mécanismes d'entrée/sortie de l'agent lui permettent de communiquer et d'agir. Le vecteur d'état de l'agent est une mémoire lui permettant de mémoriser son état ainsi que l'état de son environnement.

Un agent est un acteur communicant et il se manifeste par un comportement observable via l'acquisition et la production d'informations (c'est-à-dire qu'il communique). L'acquisition et la production d'information d'un agent sont des actions dont la réalisation se traduit par des événements. Les informations peuvent provenir ou être destinées à d'autres agents. En effet, un agent vit toujours en communauté et il conduit ses activités en parallèle avec celles de ses confrères [Coutaz et Nigay, 2001].

Nous retenons cette définition de l'agent pour notre approche dans le domaine de l'interaction homme-machine.

5.2. Structure de l'architecture à base d'agents considérée dans cette recherche

La figure 1.11 illustre la structure de l'architecture à base d'agents considérée :

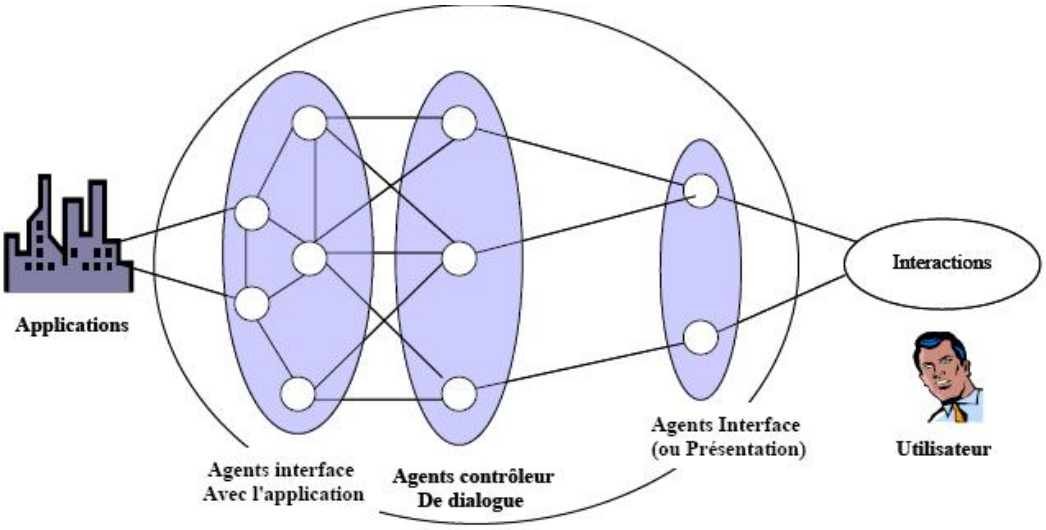

Figure 1.11 : Architecture à base d'agents [Ezzedine, 2002 ; Ezzedine *et al.*, 2005 ; Grislin-Le Strugeon *et al.*, 2001 ; Trabelsi *et al.*, 2004 ; Trabelsi 2006]

En général, les modèles d'architecture proposés visent à augmenter la réutilisabilité, la flexibilité et la maintenabilité des applications, mais ils ne prennent pas en compte les spécificités d'une application interactive de supervision dans le domaine industriel. Afin

[3] Selon [Calvary *et al.*, 2001 ; Thevenin, 2001], un contexte d'interaction pour une interface plastique est défini par le doublet <la plateforme, l'environnement>. Sur le site http://www.plasticite.com, la plasticité des systèmes interactifs concerne leur capacité à s'adapter à la diversité des plates-formes d'interaction (PC, assistant personnel, téléphone mobile) et à l'environnement physique dans lequel s'inscrit l'interaction (chez soi, en voiture, en train, etc.) et l'adaptation logicielle à la diversité des contextes d'interaction doit se faire à moindre coût pour le développeur tout en préservant l'utilisabilité du système

d'exploiter les avantages des deux types d'architecture (fonctionnelle et structurelle), nous nous intéressons aux architectures mixtes à base d'agents empruntant les principes de ces deux types d'architectures. Quelques modèles ont été déjà proposés comme les modèles hybrides (cf. 4) mais notre modèle d'architecture à base d'agents vise principalement la conception des systèmes interactifs complexes dans le domaine industriel. En se basant sur cette architecture ainsi que sa description formelle, nous allons proposer un modèle générique et configurable d'un environnement d'aide à l'évaluation des systèmes interactifs à base d'agents (cf. chapitre 3). Cette architecture a été déjà présentée dans une littérature abondante [Ezzedine, 2002 ; Ezzedine *et al*., 2005 ; Grislin-Le Strugeon *et al*., 2001 ; Trabelsi *et al*., 2004 ; Trabelsi, 2006].

Celle-ci se compose de trois composants fonctionnels (comme le modèle fonctionnel Seeheim) que nous appelons respectivement : le composant *interface avec l'application* (qui se connecte directement à l'application), le composant *Interface* (ou *Présentation* qui est en contact direct avec l'utilisateur) et le composant *contrôleur de dialogue* (qui joue un rôle d'intermédiaire). Chaque composant regroupe comme dans une architecture structurelle, un ensemble d'agents. Ces trois composants peuvent être vus comme trois systèmes multi-agents (SMA) et ils peuvent être prévus pour fonctionner en parallèle.

- Composant *Application* : il contient les agents *interface avec l'application* (dit agents *application*). Ces agents sont en contact direct avec l'application et ils peuvent recevoir l'information de l'application pour la transférer aux autres agents. Ces agents ne peuvent pas être accédés directement par l'utilisateur et ils représentent le noyau fonctionnel de l'application. Ces agents manipulent les concepts du domaine et permettent d'exécuter les fonctions spécifiques au domaine de l'application. Ils assurent le bon fonctionnement de l'application et l'émission en temps réel d'informations nécessaires aux activités des autres agents. Les agents *application* forment donc un modèle de l'application réelle.
- Composant *Interface* : il contient les agents *interface* (ou *présentation*); contrairement aux agents *application*, ils interagissent directement avec l'utilisateur (visibles pour l'utilisateur) à travers les EVIUs (événements interfaces utilisateurs) associés. Ces agents se coordonnent les uns avec les autres pour :
    - intercepter les commandes de l'utilisateur et les envoyer à l'application via les autres agents. Dans les systèmes de supervision, ce rôle permet aux opérateurs en salle de contrôle/commande de réaliser leurs tâches comme la commande du processus réel industriel.
    - ou composer une présentation qui permet à l'utilisateur de comprendre la situation courante de l'application. Ceci correspond à la fonction de supervision pour les opérateurs en salle de contrôle/commande.

  La partie visible des agents *interface* de ce composant peut être simple comme une fenêtre avec une intelligence limitée.
- Composant *Contrôleur de dialogue* (ou *Contrôleur*) : il contient les agents qui correspondent à une représentation intermédiaire entre l'application et l'interface afin d'assurer la cohérence globale du dialogue. En particulier, ces agents ont pour rôle de faire le lien entre les deux autres composants en distribuant les commandes de l'utilisateur vers les agents concernés de l'application, et en distribuant les retours de l'application vers les agents *interface* concernés.

Nous avons décrit le modèle complet de l'architecture par rapport à laquelle nous nous positionnons. En réalité, il n'est pas obligatoire qu'un système interactif possède l'ensemble des trois types d'agents (*application*, *contrôleur* et *interface*). Il est possible qu'un système

interactif soit seulement composé des agents *interface*, ou d'agents *interface* et d'agents *application*. Ces deux cas correspondent à des modèles incomplets de cette architecture. Chaque agent de cette architecture est associé à un ensemble des services qui sont les actions qu'il peut exécuter. Le dialogue entre les agents est réalisé par les invocations entre les services des agents. La description formelle des agents ainsi que ses événements (EVIUs-événements interfaces utilisateurs, les services) seront présentés ultérieurement dans ce mémoire (cf. chapitre 3).

5.3. Discussion sur l'architecture considérée

Nous nous sommes positionnés par rapport à une architecture mixte à base d'agents dédiés aux systèmes interactifs. Avec cette architecture, le concepteur peut ajouter un nombre quelconque d'agents *interface* pour développer un nombre quelconque de nouvelles vues avec l'utilisateur. Cette possibilité est utile pour le développement de l'IHM souvent complexe des systèmes de supervision et pour le développement des interfaces plastiques. En effet, on peut concevoir les différents agents *interface* pour les différents types d'utilisateurs et/ou plate-formes.

Des aspects difficiles à traiter par les concepteurs lorsqu'ils appliquent cette architecture pour construire un système interactif sont l'organisation des agents ainsi que la distribution des travaux et connaissances entre les agents dans chaque composant ou même entre les agents des trois composants, notamment dans les grands systèmes[4]. Cette distribution affecte beaucoup la vitesse de transmission des informations entre les agents [TRAN et *al.*, 2008b]. En effet, dans le contexte des systèmes interactifs complexes de supervision dont les utilisateurs (en tant qu'opérateurs en salle de contrôle) doivent superviser un processus et effectuer sans cesse des manipulations avec exactitude et efficacité, la distribution des travaux et des connaissances entre les agents *interface* de cette architecture peut se baser sur le critère « naturel » qui correspond aux besoins de supervision et de régulation des opérateurs dans leurs problèmes spécifiques. C'est la meilleure solution, à notre avis. Plus précisément, chaque agent *interface* peut servir à une tâche de supervision ou de régulation que les opérateurs doivent réaliser dans la situation réelle de leur travail. Les opérateurs devraient comprendre plus facilement l'interface homme-machine et réaliser les tâches dans de bonnes conditions.

En plus, dans les systèmes interactifs de supervision, les opérateurs doivent connaître le plus rapidement possible l'état courant du processus qu'ils supervisent ; les commandes ou messages des opérateurs doivent être envoyés à ce processus à temps, notamment en cas de dysfonctionnement. L'état du processus supervisé doit être rapidement perçu par les agents *application* concernés, et puis, ces agents *application* doivent transmettre l'information nécessaire sur cet état, en temps réel, aux agents *interface* concernés via les agents *contrôleur* pour que ces agents *interface* puissent présenter l'état courant du processus à l'opérateur. D'autre part, après avoir reçu la commande ou les messages de l'opérateur, les agents *interface* concernés doivent les transmettre rapidement, en temps réel, aux agents *application* concernés via les agents *contrôleur*. Une lenteur ou retard de telles transmissions et des actions de l'opérateur peut être source de conséquences ; c'est pourquoi, à côté de l'organisation, de la distribution du travail et des connaissances entre les agents *application* ainsi qu'entre les agents *contrôleur* qui sont aussi importants, les interactions entre les agents sous forme de leurs services interagissant doivent être aussi pris en compte. Donc, le temps de réponse entre les services interagissant est un paramètre important pour évaluer la

[4] Un système multi-agents (SMA) contenant plus de 10 agents peut être considéré comme un grand système, d'après [Wooldridge et Jennings, 1999].

performance du système interactif. Le plateforme d'évaluation que nous allons proposer et presenter dans le troisième chapitre peut calculer ce paramètre.

6. Conclusion

Dans ce chapitre, nous avons présenté un état de l'art sur les architectures des systèmes interactifs.

D'abord, ce chapitre a été consacré aux deux grands types de modèles d'architectures : les modèles *fonctionnels* et *structurels*. Le principe de base de chaque modèle et quelques modèles parmi les plus connus ont été introduits et discutés (ARCH, Seeheim, MVC, PAC, AMF).

Ensuite, les modèles d'architecture hybrides empruntant les principes de ces deux types d'architectures ont été proposés. Les modèles hybrides H^4 et PAC-Amodeus ont été présentés et discutés.

Après avoir introduit brièvement les concepts de base relatifs aux systèmes multi-agents (SMA) et aux agents, nous avons présenté notre modèle d'architecture mixte à base d'agents dédié principalement à la conception des systèmes interactifs de supervision. Ce modèle mixte permet d'exploiter les avantages des modèles fonctionnels et structurels. Il se compose de trois composants fonctionnels (comme le modèle de *Seeheim*), appelés respectivement : *Interface avec l'application* (en contact lien direct avec l'application), *Contrôleur de dialogue* et *Présentation* (en contact direct avec l'utilisateur). Chaque composant est composé d'un ensemble d'agents de la même façon que les modèles structurels. Les agents de ces composants communiquent entre eux par les invocations entre leurs services. Une discussion sur ce modèle d'architecture a terminé le chapitre 1.

Les modèles d'architecture guident les concepteurs dans leur développement de systèmes interactifs. Ils visent à augmenter la réutilisabilité, la flexibilité des applications et à faciliter la maintenance. Mais ces modèles ne sont pas suffisant pour concevoir des applications de qualité. Afin d'atteindre ce but, l'évaluation est essentielle pour produire un système interactif de qualité (afin de mettre en évidence les défauts de conception) et elle peut aussi aider les concepteurs à améliorer les systèmes interactifs déjà conçus (en détectant leurs problèmes et défauts). Le chapitre suivant est justement consacré aux méthodes et outils d'évaluation qui sont très nombreux et variés.

Chapitre 2 : Méthodes et outils d'évaluation des systèmes interactifs

Sommaire

1. Introduction

L'évaluation des systèmes interactifs est un domaine de recherche très actif depuis plus d'une trentaine d'années. On considère qu'une interface utilisateur doit augmenter le confort, la satisfaction et la productivité des utilisateurs, et aussi diminuer les risques d'erreurs que les utilisateurs peuvent commettre. Afin d'aider les concepteurs à comprendre les difficultés que les utilisateurs rencontrent lorsqu'ils interagissent avec les systèmes interactifs, et à améliorer les IHM en conséquence, des méthodes et outils d'évaluations sont disponibles. Ils permettent d'évaluer les systèmes interactifs en respectant différents critères. Avant de présenter les classifications de méthodes d'évaluation ainsi que plusieurs outils dédiés, le principe général de l'évaluation est défini.

D'après [Senach, 1990], « toute évaluation consiste à comparer un modèle de l'objet évalué à un modèle de référence permettant d'établir des conclusions ». La figure 2.1 illustre le principe de l'évaluation.

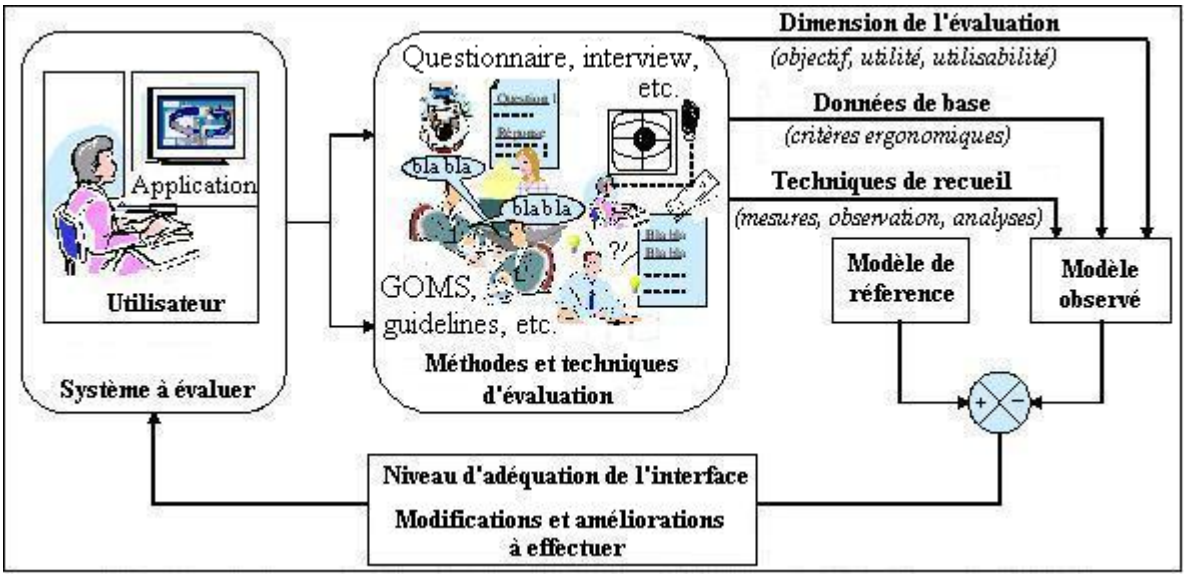

Figure 2.1 : Principe de base de l'évaluation ([Grislin, 1995] inspiré de [Senach, 1990])

D'après le principe illustré en figure 2.1, on applique au système interactif à évaluer des méthodes et techniques d'évaluation exploitant des critères ergonomiques ; il s'agit aussi de faire les analyses nécessaires sur les données recueillies (exécution de statistiques, détermination des violations selon les règles ergonomiques, etc.) pour obtenir un modèle appelé « modèle observé ». Ce modèle va être comparé au modèle de référence qui doit être représentatif de l'interface à évaluer par rapport aux besoins spécifiques déterminés par le concepteur. Le résultat de la comparaison doit permettre au concepteur d'estimer le niveau d'adéquation du système interactif par rapport aux besoins déjà déterminés et aider le concepteur à proposer les modifications pour améliorer celui-ci.

Afin d'éclaircir les relations complexes entre la richesse des fonctionnalités d'un logiciel interactif, leur intérêt pour l'utilisateur et la facilité d'utilisation, deux dimensions principales d'une évaluation sont différenciées [Senach, 1990] :

- *L'utilité* : cette dimension détermine si le système interactif permet à l'utilisateur d'atteindre ses objectifs de travail [Senach, 1990]. Elle porte sur quelques propriétés comme la performance du système, la capacité fonctionnelle, la qualité de l'assistance technique proposée au client [Nielsen, 1993 ; Senach, 1990].

- *L'utilisabilité* : ce terme vient du concept « usability » de [Eason, 1984]. En fait, il n'y a pas de définition homogène pour le terme « utilisabilité » et plusieurs définitions ont été proposées [Abran *et al*., 2003 ; Dix *et al*., 1993 ; ISO/IEC 9126-1, 2001 ; ISO 9241, 1998 ; Nielsen 1993]. En général, cette dimension vise plusieurs facteurs comme le temps d'exécution, la satisfaction de l'utilisateur, la facilité d'apprentissage, la facilité d'appropriation, l'effectivité ou l'efficacité qui peuvent être combinés d'après [Abran et al., 2003]. L'utilisabilité concerne la cohérence, la clarté visuelle, la flexibilité et le contrôle, la compatibilité, les retours d'information, la qualité de la documentation de l'aide en ligne, la gestion des erreurs, etc., d'après [Robert, 2003]. Les lecteurs intéressés peuvent consulter quelques documents comme [Abran *et al*., 2003 ; Dix *et al*., 1993 ; Nielsen, 1993 ; Ravden et Johnson, 1989], etc., pour les différents critères et les modèles d'utilisabilité.

La figure 2.2 illustre ces deux dimensions d'après Senach :

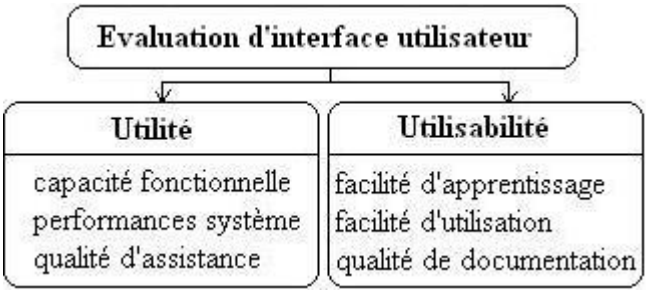

Figure 2.2 : Utilité et Utilisabilité [Senach, 1990]

Lorsque les dimensions sont établies, il faut définir les variables cibles qui vont être enregistrées. La figure 2.3 illustre un ensemble de variables cibles utilisables lors de l'évaluation d'IHM.

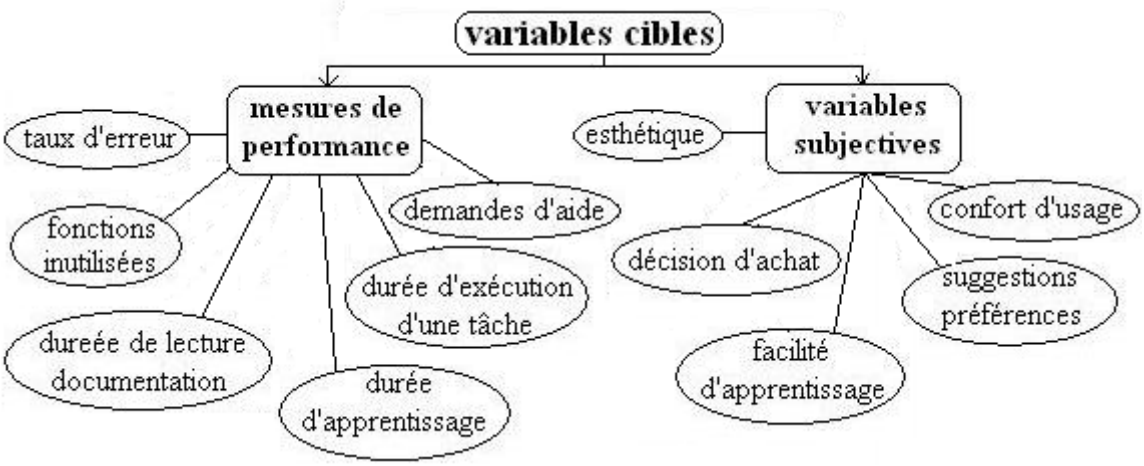

Figure 2.3 : Ensemble de variables cibles utilisables lors de l'évaluation d'IHM [Senach, 1990]

Les méthodes d'évaluation sont très nombreuses. En conséquence, il est nécessaire de les classifier. Plusieurs classifications ont été proposées dans la littérature.

2. Classifications des méthodes d'évaluation

Plusieurs auteurs ont proposé leur propre classification ; nous donnons ci-dessous une vue d'ensemble de quelques classifications :

- Dans [Senach, 1990], les méthodes d'évaluation sont classifiées en deux types : 1) Les méthodes d'évaluation empiriques qui consistent à capturer des données relatives au comportement de l'utilisateur lorsqu'il utilise le système (données comportementales). De telles méthodes exigent un système réel à évaluer (une maquette, un prototype ou le système final) ainsi que la présence d'un ou plusieurs utilisateurs. 2) Les méthodes analytiques visent à évaluer la conception du système et non son utilisation. De telles méthodes sont appliquées sur un modèle dit d'interface ou d'interaction. Le recours à des représentations abstraites permet des prédictions relatives aux performances qui ne peuvent pas être établies dans une méthode purement empirique. La figure 2.4 illustre cette première classification. Le contrôle de qualité d'une IHM est défini comme une évaluation *a priori* des caractéristiques ergonomiques de l'interface, nécessitant l'emploi de méthodes analytiques [Senach, 1990]. D'après l'auteur, les deux types de méthodes ne sont pas opposés mais complémentaires. Par exemple, les études analytiques exploitent les résultats d'études empiriques pour valider leurs prédictions et les méthodes empiriques conduisent à élaborer des hypothèses contribuant au développement des méthodes analytiques.

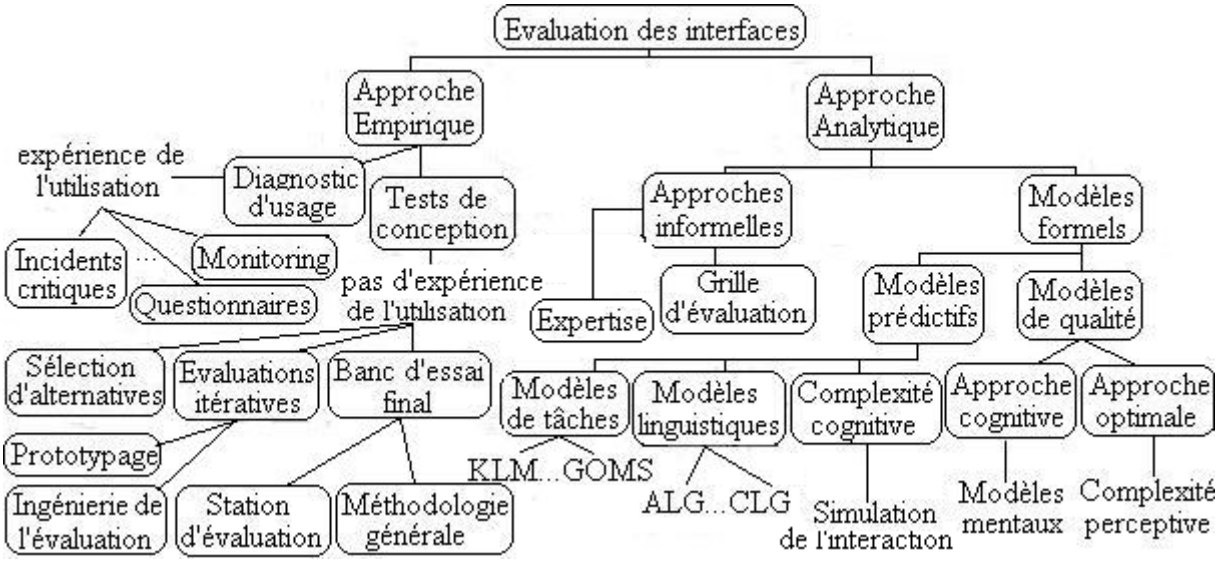

Figure 2.4 : Classification des méthodes d'évaluation proposée par Senach [Senach, 1990]

- [Whitefield *et al*., 1991] se base sur la présence réelle ou représentée de l'utilisateur et du système pour classer les méthodes d'évaluation. Quatre approches d'évaluation possibles sont ainsi distinguées :
    - Les méthodes analytiques (utilisateur et système non réels) : ces méthodes ont recours à des représentations de l'utilisateur et du système pour prédire les performances de l'utilisateur. La représentation peut, par exemple, exploiter des modèles de tâches comme KLM ou GOMS.
    - Les rapports de spécialistes (utilisateur représenté et système réel) : de telles méthodes ne font pas appel aux utilisateurs réels, mais elles nécessitent la présence du système réel. On peut citer, par exemple, les méthodes basées sur l'emploi de règles ergonomiques pour évaluer l'interface.
    - Les rapports d'utilisateurs (utilisateur réel et système représenté) : ces méthodes sont généralement utilisées lorsqu'on ne dispose pas de système réel. Dans ce cas,

il est demandé à l'utilisateur de juger certains aspects conceptuels généraux du système. Par exemple, on peut pour cela utiliser la technique du questionnaire.

- Les méthodes d'observation (utilisateur et système réel) : ces méthodes consistent à observer les utilisateurs lorsqu'ils interagissent avec le système pour déterminer les erreurs, les défaillances du système ou des utilisateurs. Notons que cette classification est similaire à celle de [Bastien et Scapin, 2001] que nous présentons ci-dessous.

- [Bastien et Scapin, 2001] distinguent deux grandes catégories de méthodes d'évaluation :
  - Les méthodes qui exigent la participation directe de l'utilisateur. Dans cette catégorie, les rapports d'utilisateurs et les méthodes d'observation de [Whitefield, 1991] ci-dessus présentés appartiennent à cette catégorie.
  - Les méthodes sans interaction directe entre l'utilisateur et le système. Elles regroupent les méthodes analytiques et les rapports de spécialistes de [Whitefield, 1991] ci-dessus présentés.

- [Grislin et Kolski, 1996] proposent une classification selon trois grandes approches comme l'illustre la figure 2.5.
  - les approches centrées sur les avis et/ou les activités des utilisateurs,
  - les approches centrées sur une expertise. Ces approches se basent sur le jugement d'experts en communication homme-machine ou sur l'utilisation de grilles d'évaluation ou de questionnaires listant les qualités d'une bonne IHM,
  - les approches analytiques centrées sur une modélisation de l'IHM et/ou de l'interaction homme-machine. Ces approches consistent le plus souvent à effectuer l'évaluation à l'aide de métriques objectives à partir d'un modèle descriptif des tâches humaines, ou à partir d'une description des pages-écrans.

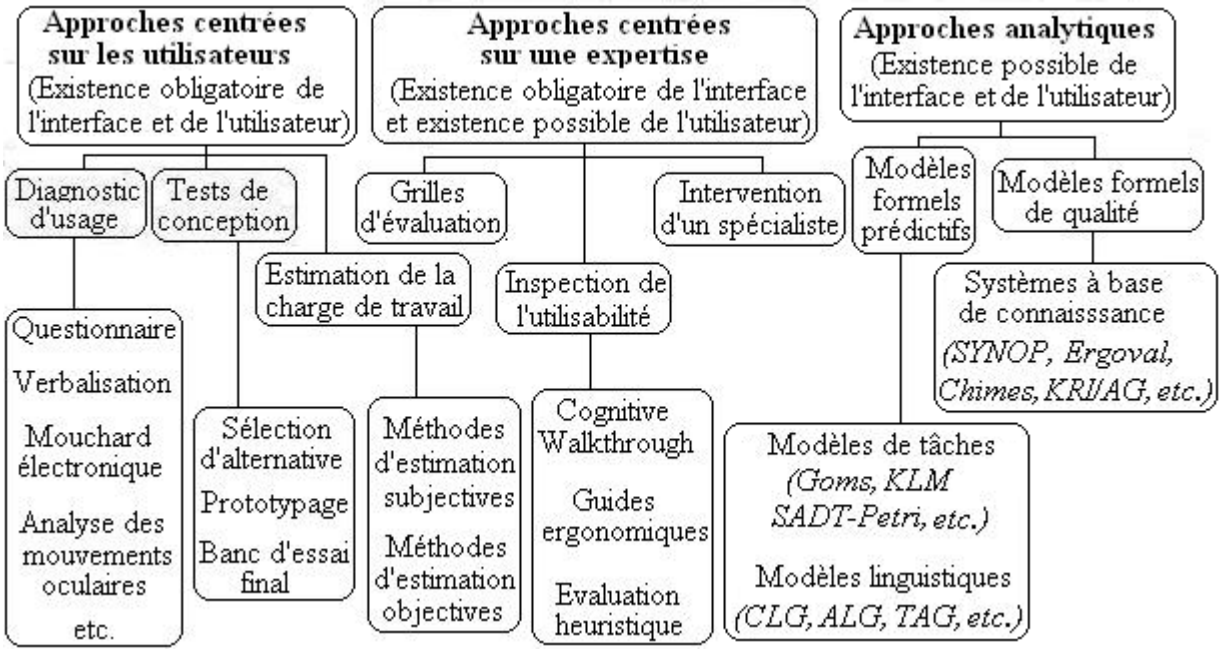

Figure 2.5 : Classification des méthodes et techniques d'évaluation selon [Grislin et Kolski, 1996]

On peut trouver explicitement ou implicitement ces trois approches dans la plupart des classifications existantes. On peut trouver une classification considérant une quarantaine de méthodes dans [Grislin, 1995]

Les lecteurs intéressés peuvent consulter [Balbo, 1994 ; Farenc, 1997 ; Holyer, 1993] pour prendre connaissance d'autres classifications.

L'objectif principal de cette thèse est de proposer et développer un système d'aide à l'évaluation des systèmes interactifs. C'est pourquoi, nous allons présenter d'abord un ensemble d'outils d'aide à l'évaluation.

3. Outils d'aide à l'évaluation

3.1. Introduction

Depuis une trentaine d'années, un ensemble d'outils logiciels d'aide à l'évaluation des systèmes interactifs ont été proposés. A ce sujet, on peut en proposer une première classification ci-dessous en se basant sur [Bastien et Scapin, 2002 ; Ivory et Hearst, 2001] :

- Des outils permettent d'abord d'aider l'évaluateur à prendre des décisions concernant le choix de méthodes ou techniques d'évaluation, par exemple, le système ADHESION [Nendjo Ella, 1999], l'outil d'aide à l'évaluation de [Denley et Long, 1997], etc.
- Des outils d'organisation de l'évaluation, tel FACE (Fast Audit base on Cognitive Ergonomics) [Hulzebosch et Jameson, 1996], visent à permettre à l'évaluateur de cerner la situation de l'évaluation, de choisir la technique d'évaluation à appliquer et de présenter les résultats.
- Certains outils correspondent à des versions logicielles de méthodes exploitant dans leur version des documents papiers comme la version logicielle de la méthode intitulée Cognitive Walkthrough [Rieman *et al*., 1991].
- Des outils utilisent les règles ergonomiques (ergonomic guidelines) pour aider à déterminer si l'interface viole celles-ci.
- Des outils permettent de recueillir des données durant l'interaction entre l'utilisateur et le système interactif pour l'aide à l'évaluation. Ils sont ensuite éventuellement associés à des outils permettant d'en effectuer une analyse.

Ces outils fournissent un niveau varié d'automatisation de l'évaluation. D'après [Ivory et Hearst, 2001 ; Ivory, 2001], l'automatisation de l'évaluation apporte plusieurs avantages :

- Réduire le coût et le temps de l'évaluation parce que les outils automatisent certaines activités d'évaluation comme la capture de données, l'analyse de données capturées, ou les critiques ;
- Faciliter la détection des erreurs. Des activités possibles à l'aide de l'interface peuvent en effet être spécifiées avec des modèles de tâche comme CTT [Paterno *et al*., 1997]. Des outils peuvent détecter automatiquement des erreurs ou des incohérences par rapport à cette spécification
- Réduire le besoin de recours à des experts en évaluation parce que l'automatisation de quelques activités d'évaluation comme la capture, l'analyse et les critiques est bien une première aide aux concepteurs qui le plus souvent n'ont pas ou que peu de connaissance ou expérience en évaluation.
- Augmenter le recouvrement des aspects évalués ; il n'est pas toujours possible d'évaluer indépendamment chaque aspect de l'interface à cause du manque de temps, des ressources

limitées, ou du coût élevé. Grâce aux outils, le nombre d'aspects évalués peut être augmenté. [5]

L'automatisation n'est pas une substitution des techniques d'évaluation manuelles mais vient plutôt en complément. Les différentes techniques visent à découvrir des problèmes complémentaires. Par exemple, les mesures subjectives comme la satisfaction de l'utilisateur est difficilement prédictible par des méthodes automatisées [Ivory, 2001].

Parmi les types d'outils d'aide à l'évaluation cités ci-dessus, nous nous intéressons particulièrement aux deux derniers : aux outils utilisant les règles ergonomiques (ergonomic guidelines) et aux mouchards électroniques capturant les données durant l'interaction entre l'utilisateur et le système interactif en vue d'une aide à l'évaluation. Nous en présentons dans les deux sections suivantes des exemples représentatifs.

3.2. Outils utilisant les règles ergonomiques (ergonomic guidelines)

3.2.1. Introduction

Aujourd'hui, les règles ergonomiques (ergonomic guidelines) sont utilisées comme une source importante pour détecter les problèmes de l'interface utilisateur du système interactif afin de l'améliorer. Le terme « règle » (guideline) englobe les recommendations abstraites ou concrètes possibles qui sont utilisées pour la conception (par les concepteurs) et l'évaluation des systèmes interactifs (par les experts en utilisabilité) afin de concevoir des interfaces utilisateurs plus efficaces et conviviales [Grammenos *et al*., 2000a ; 2000b]. D'après [Vanderdonckt, 1994], une règle ergonomique constitue un principe de conception et/ou d'évaluation à observer en vue d'obtenir et/ou de garantir une interface homme-machine ergonomique. [Vanderdonckt, 1994] a aussi distingué cinq types de recommendations : les standards de conception, les articles de recommendations, les guides de recommendations, les guides de style, les algorithmes de conception ergonomique. Les règles peuvent correspondent à des recommendations générales et indépendantes des domaines comme dans [Scapin, 1986 ; Smith et Mosier, 1986], etc. Elles peuvent aussi être regroupées dans des guides de style (style guides) propres à un système, un environnement ou une organisation particulière comme dans [Apple, 1992 ; HP, 1988 ; IBM, 1993 ; Microsoft, 1995 ; OSF 1993], etc. Actuellement, il existe de nombreuses sources décrivant des règles ergonomiques :

- Les standards internationaux comme ISO 9241, ISO/IEC 9126, etc.
- Les standards nationaux comme HFES (aux Etats Unis), AFNOR (France), CBN (Belgique), BSI (Grande Bretagne), etc.
- Des documents de synthèse passent en revue les critères ergonomiques comme [Scapin, 1986 ; Smith et Mosier, 1986 ;Vanderdonckt, 1994], etc.
- Des standards sont issus aussi d'entreprises comme [Apple, 1992 ; IBM, 1993 ; Microsoft, 1995], etc.

Lors de la conception d'une interface homme-machine, les règles ergonomiques ont pour but de faire respecter un ou plusieurs critères ergonomiques [Vanderdonckt, 1994]. Chaque critère constitue une dimension reconnue sur le chemin qui mène à l'élaboration d'une interface efficace, sophistiquée, plus conviviale et moins encline à l'erreur [Scapin, 1989]. A titre d'exemple, huit critères généraux peuvent être identifiés et retenus [Bastien et Scapin, 1993 ; Vanderdonckt, 1994] : le guidage, le contrôle du dialogue, la gestion des erreurs, la cohérence, la charge de travail, l'adaptabilité, la compatibilité, la signifiance des codes et dénominations. [Vanderdonckt, 1994] a présenté une taxonomie arborescente de règles, qui

[5] Les utilisateurs intéressés peuvent consulter [Ivory et Hearst, 2001] ou [Ivory, 2001] pour plus de détails sur ces avantages.

comprend 12 divisions (la saisie de données, l'affichage de données, le dialogue, le graphisme, les moyens d'interaction, les styles d'interaction, le guidage de l'utilisateur, les messages, l'aide en ligne, la documentation, l'évaluation, et l'implémentation) dans laquelle chaque règle s'inscrit.

Les règles ergonomiques sont souvent documentées dans des manuels de référence. Pour utiliser les règles, on doit consulter ces documents. L'utilisation de cette manière des règles révèle quelques inconvénients. [Grammenos *et al*., 2000b] a présenté quelques limitations à ce sujet, et aussi concernant le contenu de ces règles elles-même (bien que la valeur et l'importance des règles soient indiscutables) :

- Difficulté de gestion des règles ; par exemple, il peut être difficile de sélectionner un sous-ensemble de règles, ou de combiner les règles de différents manuels.
- Difficulté de mise à jour (ajout de nouvelles règles, suppression de règles existantes).
- Difficulté à modifier le contenu des règles existantes.
- Les manuels papier sont difficiles à distribuer.
- Les manuels papier sont difficiles à parcourir. Une recherche peut nécessiter beaucoup de temps.
- En plus, les règles sont souvent indépendantes des contextes ; il faut donc en faire une interprétation éventuellement approfondie pour bien les utiliser. Les concepteurs et développeurs ne sont pas toujours familiers avec les langages et styles utilisés pour exprimer les règles.

Ces inconvénients mènent à la naissance d'un type d'outils considérés commes utilisant les règles ergonomiques. Quelques outils représentatifs parus depuis une dizaine d'années sont examinés successivement.

3.2.2. Système proposé par Parush

[Parush, 2000] a proposé un système de gestion de règles implémenté à l'aide de MS Visual Basic. Il se compose de deux modules principaux : un module de construction et un module de lecture. Ces deux modules travaillent avec une base de données contenant des règles communes. Chaque règle est structurée selon trois niveaux et quatre parties complémentaires. Ces trois niveaux sont :

- **Section :** c'est le niveau le plus élevé. Il peut être général comme *« Controls »* ou *« Displays »* ou plus spécifique comme *« Dialog Boxes Layout »*. Le niveau *« Section »* représente la catégorie de la règle.
- **Sous-Section :** c'est le deuxième niveau. Sa granularité dépend de la section parente. Une sous-section peut avoir plus d'une section parente. Par exemple, *« Buttons »* peut être la sous-section des 2 sections *« Controls »* et *« Displays »*. Le niveau *« Sous-Section »* représente la sous-catégorie de la règle.
- **Titre :** c'est une simple ligne de texte ; sa relation avec la section et sous-section crée une unique règle. Un titre peut être combiné avec plus d'une section ou d'une sous-section. Par exemple *« Font Size »* peut être le titre de la règle de la sous-section *« Buttons »* (de la section *« Controls »*) et aussi de la sous-section *« Buttons »* (de la section *«Displays »*).

Chaque règle peut avoir quelques parties complémentaires :

- **Recommandation :** l'utilisateur détaille la règle dans cette partie avec un texte de longueur non limitée. Par exemple *« Fonts should be MS Sans Serif size 9 normal »* pour

la règle *« Label Font and style», « Field size should fit the amount of text to be entered »* pour la règle *« Field Size »*.

- **Notes additionnelles :** un texte peut être utilisé pour l'administration de la règle. Par exemple, *« This guideline should be ignored in the current 96' version of the system »*.
- **Fichier d'image** pour représenter sur l'interface du système comme un symbole de la règle.
- **Liste des références aux autres règles** : on liste des références des règles concernant celle en question.

Chaque règle est stockée dans la base, en respectant cette structure. L'utilisateur peut entrer les règles en base de données à travers le **module de construction**. Celui-ci fournit à l'utilisateur une interface pour lui permettre d'ajouter les nouvelles règles, ainsi que leurs informations concernées en base des règles et de modifier les règles existantes en respectant la structure d'une règle présentée ci-dessus. Le **module de lecture** de ce système fournit à l'utilisateur une interface pour lui permettre : 1) de parcourir les règles selon la structure hiérarchique du système. 2) d'interroger la base de données pour récupérer un sous-ensemble spécifique de règles selon les besoins de l'évaluateur en se basant sur les informations relatives aux règles (noms de sections, sous-sections, titres, etc.).

D'après l'auteur [Parush, 2000], la structure en 3 niveaux des règles présentée ci-dessus est rigide. Une structure flexible des règles contenant seulement deux entités (« Section » représentant la catégorie du règle et « *Titre* » représentant une règle spécifique) a été aussi proposée. Les sections peuvent être imbriquées entre elles. Le nombre de niveaux des sections n'est pas limité et l'utilisateur peut les créer à volonté. Par exemple, la section « *Dialog* » peut contenir 2 titres « *Dialog Title* » et « *Dialog Termination Buttons* » et une sous-section « *Appearance* » dans laquelle on trouve le titre « *Fonts* ».

3.2.3. Sherlock

Sherlock [Grammenos *et al*., 2000a ; 2000b] est un système de gestion de règles (GMS-Guideline Management System). Il correspond à un environnement intégré permettant d'accéder à des règles, de réutiliser des expériences passées et de faciliter l'inspection automatique sous l'angle de l'utilisabilité de l'interface. La figure 2.6 illustre la structure globale de Sherlock (une architecture client-serveur). Le module client Sherlock et le module serveur Sherlock peuvent résider sur une même machine ou des machines différentes ; ils se connectent à travers l'Internet ou le réseau Intranet de l'entreprise en utilisant le protocole TCP/IP. Comme l'illustre la figure 2.6, le fonctionnement de Sherlock est le suivant :

- Etape 1 : les différentes sources contenant les règles d'utilisabilité sont les entrées du module serveur Sherlock. Les règles et leurs routines d'inspection correspondantes sont exploitées par le module serveur Sherlock pour inspecter l'utilisabilité de l'interface.
- Etape 2 : lorsque le module serveur Sherlock reçoit la description de l'interface utilisateur en provenant du client Sherlock, il procède à l'inspection de l'interface utilisateur en invoquant des routines d'inspection des règles. Enfin, il retourne au client Sherlock le rapport des violations sous l'angle de l'utilisabilité détectées sur l'interface. La communication est réalisée en utilisant TCP/IP comme protocole.

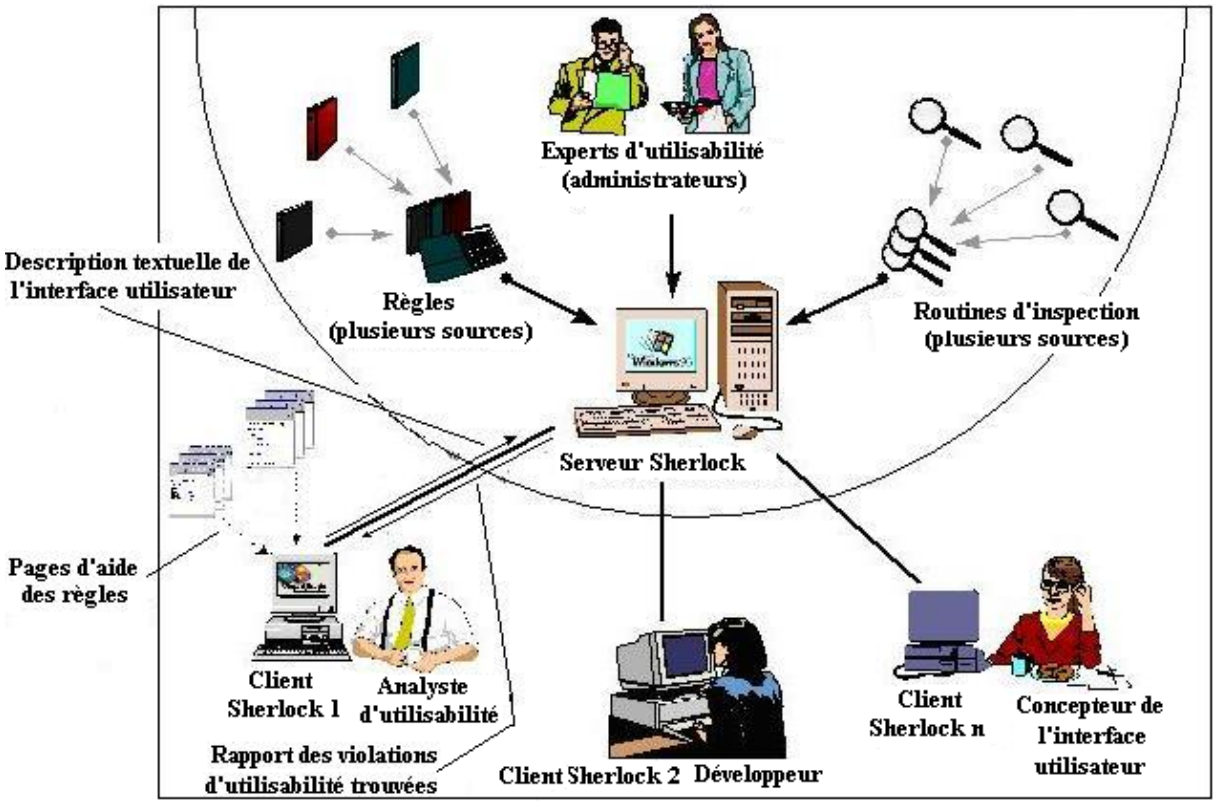

Figure 2.6 : Architecture de Sherlock, traduite de [Grammenos *et al.*, 2000a ; 2000b]

1) Communication entre module serveur Sherlock et fournisseurs des règles

Le module serveur Sherlock et les fournisseurs de règles sont séparés. Les sources des règles correspondent à des modules externes DLLs (Dynamic Link Libraries) qui peuvent résider localement ou être dispersés sur le réseau ; ces modules DLLs peuvent être crées ou étendus par n'importe quel langage de programmation soutenant ActiveX DLLs. Ces caractéristiques assurent une indépendance du langage ainsi qu'un mécanisme flexible pour la mise à jour ; en effet, l'auteur des DLLs peut remplacer les DLLs existant sur son site sans les redistribuer aux consommateurs. Chaque DLL contient une liste de règles et un ensemble de routines d'inspection correspondantes. Le module serveur peut invoquer ces routines pour découvrir les règles contenues par cette DLL, pour lui demander d'évaluer une interface, puis de retourner la description des problèmes d'utilisabilité détectés. Pour être plus précis, nous présentons ici l'interface de programmation composée de trois routines d'inspection que chaque DLL doit soutenir obligatoirement (c'est la seule contrainte imposée pour le développeur des règles) :

- *Public Sub Rules_Declare (Server as clsServer)*. Cette routine est utilisée pour déclarer au module serveur les règles que la DLL contient.
- *Public Sub Evaluate (UITree As clsUITree, ruleKeys As String)*. Cette routine est invoquée par le module serveur durant l'inspection, pour demander à la DLL d'évaluer l'interface décrite dans le paramètre d'entrée *UITree* en se basant sur la liste des règles disponibles dans le paramètre d'entrée *ruleKeys*
- *Public Sub UP_Report (RuleKey, UIComponent, ShortDescription, Solution, Optional LongDescription = "<None>", Optional pPossibleSideEffects = "<None>")* sert à retourner la description des problèmes d'utilisabilité trouvés par le module serveur. Cette description est compilée et renvoyée au client sous forme d'un rapport (figure 2.7).

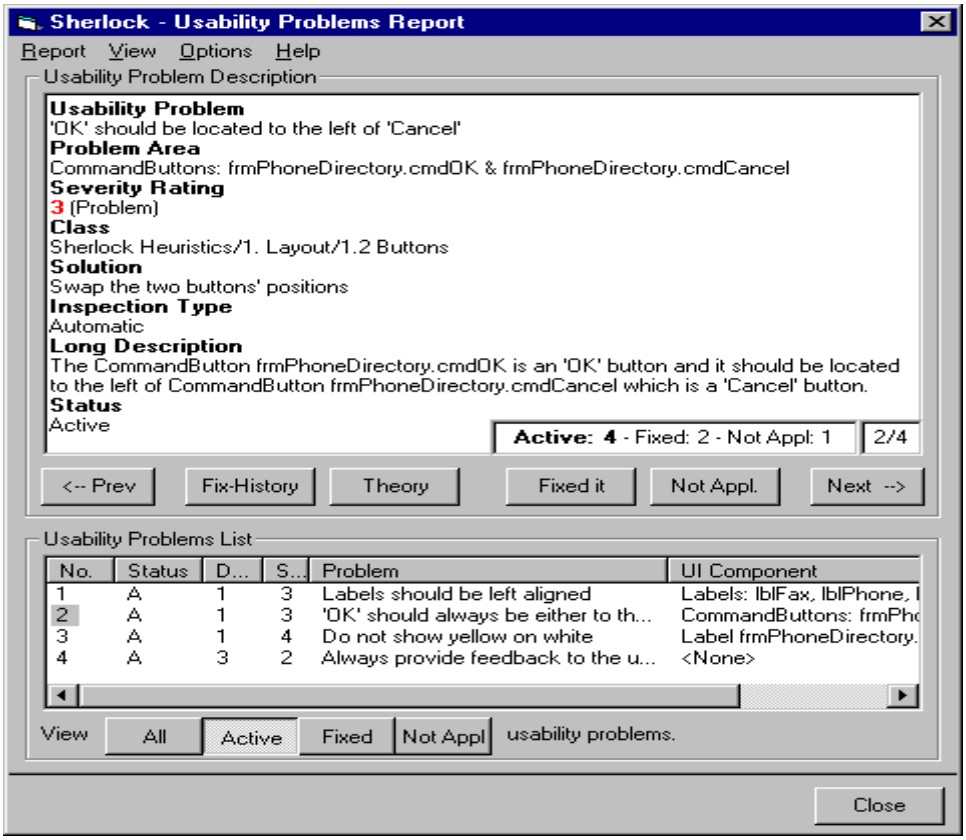

Figure 2.7. Liste des problèmes d'utilisabilité trouvés par Sherlock [Grammenos et al., 2000a ; 2000b]

Il n'y a aucune contrainte imposée sur l'implémentation réelle de ces routines. Cette particularité permet au développeur des DLLs d'optimiser le code des routines. Il n'y a aucune règle ou routine d'inspection embarquée dans le module serveur. Tout ce que le module serveur doit faire est d'invoquer les routines d'inspection des DLLs. Si deux DLLs contiennent un même ensemble des règles, alors l'administrateur va déterminer la priorité (par exemple, si deux DLLs contiennent l'ensemble de règles ISO 9241-Part 17, alors la DLL du serveur ISO est choisie). Des nouvelles sources de règles peuvent être facilement ajoutées : il s'agit de déclarer le nom et la localisation des DLLs correspondantes. Le module serveur permet à l'administrateur de parcourir les règles fournies par une DLL quelconque ; il peut alors choisir un sous-ensemble de ces règles dans le but de les stocker dans la base de règles du serveur sous un format spécifique. Dans Sherlock, chaque règle est structurée selon plusieurs propriétés : *clef*, *titre*, *classe de règle*, description, *niveau de gravité*, *type d'inspection*, *auteur*, *composant contenant le code source*, etc. Ces propriétés sont décrites en détail dans [Grammenos *et al*., 2000a].

Parmi ces propriétés, les deux suivantes sont utilisées par Sherlock pour classifier les règles :

- *La classe de règle* représente l'origine des règles. Par exemple, cette propriété de la règle « *Field labels explain the content of the fields* » peut être affectée par la valeur « *ISO 9241/Part 12/5. Organisation of Information/5.9 Labels/5.9.3 Designation* ».
- *Le type d'inspection* prend une des trois valeurs : « *automatique* », « *semi-automatique* » ou « *par l'utilisateur* ». Ces valeurs expriment si cette règle est évaluée automatiquement, semi-automatiquement ou par l'utilisateur. Par exemple, le serveur peut seulement évaluer la règle « *Information should be located to meet user expectations* » grâce à l'utilisateur, peut évaluer semi-automatiquement la règle « *Field labels explain the content of the*

fields » et peut évaluer automatiquement la règle « *Field labels and labels for other screen elements including text boxes, list boxes, combination boxes and icons, are consistently located adjacent to the displayed field, group or screen element* ».

2) Communication entre modules client et serveur Sherlock :

Dans la version décrite dans [Grammenos et al., 2000a ; 2000b], le module client Sherlock est sous forme d'un add-in de l'environnement de développement intégré MS Visual Basic 5.0. Il compile une description textuelle de l'interface utilisateur (sous forme d'une structure hiérarchique) qui est créée à l'aide de Visual Basic 5.0, puis il l'envoie au serveur pour être inspecté. Cette structure hiérarchique sera traitée par les routines d'inspection pour détecter les violations possibles des règles. Après avoir traité cette description de l'interface utilisateur reçue du client, un rapport des violations d'utilisabilité détectées par le module serveur est renvoyé au module client. Ce module client va informer l'utilisateur de ces résultats d'inspection reçus du serveur sous forme d'une liste des violations. L'utilisateur du client (qui peut être le développeur, le concepteur de l'interface utilisateur ou l'analyste d'utilisabilité) peut parcourir cette liste pour prendre connaissance des détails de chaque violation : raison, actions recommandées, niveau de gravité, fondement théorique et exemples, solutions effectuées dans le passé, etc. (cf. figure 2.7). Il peut envoyer au serveur la liste des règles qu'il ne veut pas prendre en compte lors de l'évaluation (dites règles désactivées). Il peut aussi envoyer au serveur les problèmes d'utilisabilité ainsi que les corrections qu'il a effectuées, pour être réutilisées ultétieurement comme une expérience.

3.2.4. Méthodologie du projet DESTINE

D'après [Beirekdar, 2004 ; Jasselette *et al.*, 2006], un défaut majeur de nombreux outils automatiques existants d'évaluation de l'interface Web utilisant les règles (guidelines) est le « hardcoding » des règles ainsi que leurs logiques (logiques d'évaluation et/ou logiques de correction) à l'intérieur du moteur d'évaluation. Autrement dit, les règles ainsi que leurs logiques sont intégrées à l'intérieur du moteur d'évaluation comme l'indique la figure 2.8 [6].

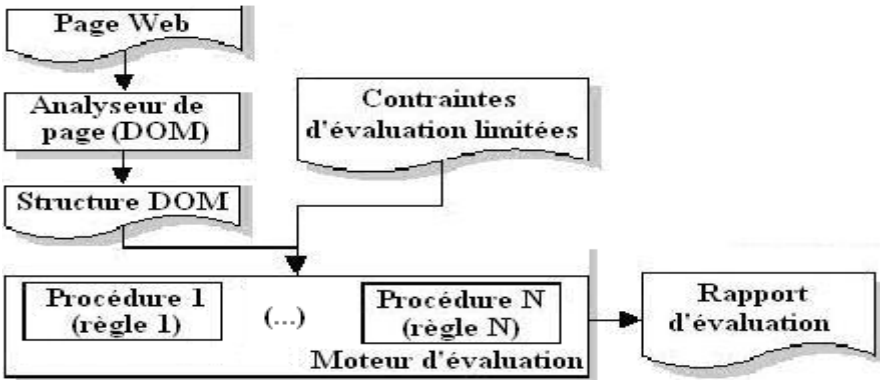

Figure 2.8. Architecture logicielle simplifiée des outils d'évaluation automatique de l'interface Web existants [Beirekdar, 2004]

En conséquence, il est difficile voire impossible d'ajouter, de modifier ou de supprimer les règles existantes sans affecter le code source du moteur d'évaluation. Ce problème de manque

[6] DOM (Document Object Model) est une convention indépendante des plateformes et des langages pour représenter et interagir avec des éléments de documents HTML, XHTML, XML. Les programmeurs peuvent développer les programmes accédant à et manipulant des éléments de tels documents en utilisant DOM API (Application Programming Interface). Selon la figure 2.8, la page Web est analysée pour générer sa structure DOM qui est l'entrée du moteur d'évaluation. La sortie de ce moteur est un rapport contenant les problèmes d'utilisabilité identifiés de cette page Web. Référence : http://www.w3.org/DOM/

de flexibilité des outils ne permet pas de répondre au besoin de changement dans le monde Internet. Pour remédier à ce problème, les auteurs ont proposé une solution d'évaluation automatisée de l'utilisabilité et de l'accessibilité de sites Web basée sur une séparation entre : les règles ergonomiques (ainsi que leurs logiques d'évaluation et/ou logiques de correction) et leur moteur d'évaluation. Cette séparation permet de structurer d'une façon dynamique et flexible les règles et de mettre à jour leurs logiques selon l'évolution rapide des technologies Web et les nouveaux résultats de recherche dans le domaine ergonomique. La version initiale de cette solution (présentée dans [Beirekdar, 2004]) vise à évaluer chaque page Web individuellement (évaluation au niveau page) par analyse statique de son code HTML en utilisant la technique dite de revue de règles (guideline review)). Avec la version étendue de cette solution présentée dans [Jasselette *et al.*, 2006], les auteurs affirment qu'elle permet d'évaluer la qualité d'un site (évaluation au niveau site). Il faut, dans ce cas, collecter et évaluer toutes les pages d'un site ; mais notons que la démarche complète et le résultat n'a pas été clairement présentés et illustrés dans cet article. La méthodologie du projet DESTINE (Design & Evaluation STudio for INtent-based Ergonomic web sites) est illustrée par la figure 2.9 ; elle divise le processus entier d'évaluation en deux étapes : la structuration des règles et l'évaluation.

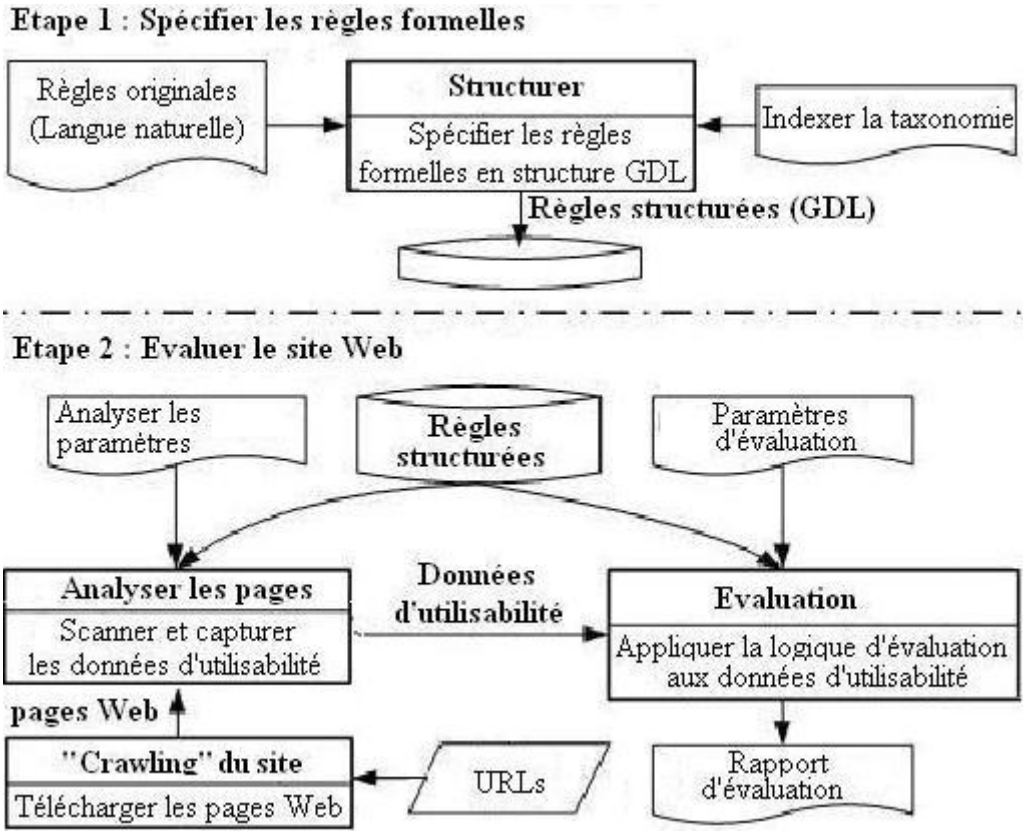

Figure 2.9 : Version initiale (correspondante à la version initiale de GDL) de la méthodologie du projet DESTINE, traduite de [Beirekdar, 2004][7]

Dans l'étape 1 (Spécifier les règles formelles), il faut sélectionner les règles provenant de différentes sources disponibles. Cette sélection est faite manuellement pour les nouvelles règles dans la mesure où l'accès aux règles peut se faire de manières différentes (articles,

[7] To crawl a site : télécharger les pages Web de ce site et les stocker localement.

livres, sites Web, etc.). Pour la méthodologie du projet DESTINE, cette activité peut être répétée plusieurs fois en raison de la séparation entre l'outil d'évaluation et les règles.

Ensuite, il faut transformer les règles (qui sont les plus souvent exprimées initialement en langage naturel) en structures que l'outil d'évaluation peut comprendre et traiter. Ces structures sont stockées sous forme XML respectant la syntaxe d'un langage de définition de règles dit GDL. Le langage GDL permet de structurer les règles d'une façon cohérente et systématique. Il est capable d'exprimer les informations relatives aux règles d'une façon suffisamment riche pour qu'un moteur d'évaluation puisse réaliser les évaluations automatisées selon n'importe quelle règle compatible GDL.

La version initiale de GDL (présentée dans [Beirekdar, 2004]) soutient seulement l'évaluation automatisée pour les éléments HTML d'une page Web ; elle ne soutient pas l'évaluation pour les éléments CSS[8]. Cette évaluation permet de détecter si ces éléments HTML respectent ou violent une règle quelconque structurée en GDL. La version étendue de GDL (présentée par [Jasselette *et al*., 2006]) soutient l'évaluation pour les éléments CSS. A propos de correction d'éléments HTML et/ou CSS violant les règles, la version initiale de GDL n'en soutient aucune. L'évaluateur doit corriger de tels éléments manuellement (en accédant directement aux pages Web concernées et modifiant le code source HTML). La version étendue fournit à l'évaluateur une correction semi-automatisée pour les éléments HTML violant les règles mais il n'y a pas encore de telle correction pour les éléments CSS.

La structure d'une règle sous forme GDL (selon la version étendue présentée dans [Jasselette *et al*., 2006]) fournit les informations principales suivantes (qui seront utilisées par le moteur d'évaluation pour évaluer une page Web quelconque selon cette règle) :

- **Ensemble d'évaluation** : c'est un ensemble d'éléments HTML, CSS que le moteur d'évaluation doit examiner pour évaluer une page Web selon cette règle (par exemple, un élément d'image HTML *img*). Dans GDL, cet ensemble est appelé *ensemble d'évaluation* (evaluation set). Tous les éléments nécessaires pour évaluer un aspect précis d'une règle sont regroupés dans un ensemble d'évaluation. On peut avoir plusieurs ensembles d'évaluation utilisés pour évaluer les différents aspects d'une règle. Par exemple, pour évaluer une page Web selon la règle "*the combination between the background color and the foreground color should belong to the best color combination or should not belong to the worst color combinations*" [Murch, 1987], on doit spécifier plusieurs ensembles d'évaluation en GDL comme :
    - {Body.bgcolor[Page], Body.text[Page]}. Cet ensemble contient les éléments HTML d'une page Web qui doivent être examinés pour évaluer la combinaison entre la couleur de fond (background color) et celle des textes de cette page (1).
    - {Body.bgcolor[Page], Body.link[Page]}. Cet ensemble contient les éléments HTML d'une page Web qui doivent être examinés pour évaluer la combinaison entre la couleur de fond (background color) et celle des liens de cette page (2).
    - {Body.bgcolor[Page], Body.alink[Page]}. Cet ensemble contient les éléments HTML d'une page Web qui doivent être examinés pour évaluer la combinaison entre la couleur de fond (background color) et celle des liens actifs (3).

[8] CSS (Cascading Style Sheets) : langage informatique utilisé pour décrire la présentation des documents HTML. Il est publié par le W3C (World Wide Web Consortium). CSS permet de séparer la structure d'un document HTML de ses styles de présentation (cf. http://www.w3c.org/Style/CSS)

- {Body.bgcolor[Page], Body.vlink[Page]}. Cet ensemble contient les éléments HTML d'une page Web qui doivent être examinés pour évaluer la combinaison entre la couleur de fond (background color) et celle des liens visités (4).
- Etc.

- **Logique d'évaluation** (**condition d'évaluation**) : c'est la logique d'évaluation que le moteur d'évaluation doit appliquer aux éléments HTML, CSS pour déterminer si ces éléments violent ou respectent cette règle. Dans GDL, les logiques d'évaluation sont appelées *conditions d'évaluation*. L'évaluateur peut utiliser des opérations mathématiques, des opérations logiques, des fonctions et/ou des valeurs définies par l'utilisateur (dont les types de données peuvent être numériques, Booléennes, des ensembles des valeurs, etc.) pour spécifier une condition d'évaluation. Par exemple, cette condition d'évaluation suivante est spécifiée pour l'ensemble d'évaluation (2) présenté ci-dessus :

  IF |Luminance of Body.link – Luminance of Body.bgcolor| >0.3 THEN

  The guideline is respected

  ELSE The guideline is violated

  Cette condition d'évaluation demande au moteur d'évaluation de calculer la différence entre la luminance de couleur de fond et la luminance de couleur des liens d'une page pour évaluer leur combinaison de couleur. Une condition d'évaluation peut être appliquée à un ou plusieurs ensembles d'évaluation.

- **Structure de correction** et **logique de correction** : la figure 2.9 illustre la version initiale de GDL dans laquelle la structure GDL d'une règle ne contient que les ensembles d'évaluation et les conditions d'évaluation. La version étendue de GDL permet de spécifier les informations nécessaires à la correction pour les éléments HTML violant cette règle (de telles informations ne sont pas encore spécifiées pour les éléments CSS). Ces informations sont représentées par les deux parties :
  - La **structure de correction**. Cette partie définit l'interface générée pour que l'utilisateur puisse corriger un élément HTML quelconque violant la règle. Le moteur d'évaluation utilise cette structure pour afficher et personnaliser une interface homme-machine (custom interface) qui permet à l'évaluateur de saisir les valeurs nécessaires pour corriger l'élément HTML violant cette règle. Cette correction est semi-automatisée. Les utilisateurs peuvent consulter [Jasselette *et al*., 2006] pour l'illustration d'une interface générée pour corriger un élément HTML *[img]* violant une règle.
  - La **logique de correction**. Cette partie spécifie la manière de traiter les données HTML afin de corriger le contenu d'une page Web. Cette logique vérifie et valide les nouvelles valeurs saisies par l'évaluateur. Si ces valeurs sont validées, alors le code source de l'élément HTML violant cette règle dans la page Web est modifié par ces valeurs saisies.

Comme nous pouvons le constater, la séparation entre les règles (ainsi que leurs logiques) et le moteur d'évaluation est assurée. Les règles ainsi que leurs logiques sont indépendantes du moteur d'évaluation ; en conséquence, elles peuvent être modifiées sans affecter le moteur. Cette activité de structuration exige de l'utilisateur une bonne compréhension de la sémantique originale des règles, une bonne connaissance de HTML (pour identifier les « tags », « attributs », etc.) et bien sûr, une riche expérience en GDL.

Enfin, l'évaluateur peut sélectionner les règles individuelles concernant les aspects ergonomiques (accessibilité, satisfaction de l'utilisateur, utilisabilité, etc.) selon l'objectif de l'évaluation.

Dans l'étape 2 (Evaluer le site web), les pages Web à évaluer sont d'abord téléchargées et analysées pour capturer les données d'utilisabilité concernant les instances des éléments HTML, CSS (l'analyse est conduite individuellement sur chaque page téléchargée). Puis, le moteur d'évaluation applique les logiques d'évaluation (déjà définies dans l'étape 1) aux instances des éléments HTML ou CSS capturés afin de détecter les respects ou les violations selon les règles. Les résultats sont présentés sous forme d'un rapport d'évaluation détaillé. La version étendue GDL soutient une correction semi-automatisée pour les éléments HTML violant les règles (pas pour les éléments CSS pour l'instant) en utilisant la structure et la logique de correction déjà spécifiée en GDL dans l'étape 1. Le processus de correction semi-automatisée fournit à l'évaluateur une interface homme-machine déjà personnalisée (custom interface) permettant à l'évaluateur de saisir les valeurs nécessaires pour corriger les éléments HTML violant la règle, chaque valeur saisie doit être validée et ensuite, le code source HTML (de ces éléments violant la règle dans la page Web) est modifié par ces valeurs saisies.

Cette méthodologie est associée à plusieurs outils :

- Un outil de gestion des connaissances d'utilisabilité et d'accessibilité permet d'organiser les bases de connaissance contenant les règles, de parcourir et de modifier celles-ci.
- Un éditeur GDL permet de formaliser les règles en GDL dans un but d'évaluation automatisée.
- Un évaluateur de qualité ergonomique des sites Web permet d'évaluer la qualité des pages Web et de générer des rapports d'évaluation. L'évaluateur peut aussi consulter le contenu du rapport sous différentes formes, il peut afficher et trier les erreurs, afficher les statistiques d'évaluation, les erreurs et/ou les tests effectués.
- Un outil de réparation ergonomique permet de corriger les pages sur lesquelles des problèmes ergonomiques ont été identifiés.
- Un outil de transformation multimodal permet de transformer les sites Web destinés à être consultés initialement sur l'écran d'ordinateur afin qu'ils puissent être compatibles avec les kiosques interactifs.

Malgré son originalité, cette méthodologie revèle quelques limitations. Ainsi, la qualité de l'évaluation dépend des capacités de l'évaluateur dans la première étape, comme sa compréhension de la sémantique des règles, sa connaissance de HTML, son expérience en GDL et sa qualité de l'interprétation des règles et de la richesse du GDL. Aujourd'hui, quelques règles ne peuvent pas être structurées en GDL, par exemple, *« les messages d'erreur ne doivent pas être moqueurs »*. Cette méthodologie n'évalue que les pages Web selon les règles capables d'être représentées en GDL. Enfin, comme précisé ci-dessus, l'évaluation au niveau site dans la version étendue de cette solution ([Jasselette *et al.*, 2006]) n'a pas été clairement présentée et illustrée.

3.2.5. Méthodologie du projet WebTango

[Ivory, 2001] a proposé une méthodologie d'évaluation automatisée des sites Web ainsi que des outils associés. La proposition de l'auteur se base sur deux critiques :

- Les approches de revue de règles (guideline review) existantes couvrent seulement une petite fraction des aspects d'utilisabilité, d'après [Brajnik, 2000].

- La plupart des règles pour la conception du Web ainsi que les approches de revue des règles existantes ne sont pas validées en se basant sur des calculs empiriques.

En conséquence, l'auteur a proposé une nouvelle méthodologie basée sur des calculs empiriques. En effet, cette méthodologie WebTango concerne des calculs liés à 141 mesures au niveau page et 16 mesures au niveau site. Ces mesures concernent plusieurs aspects des interfaces Web comme la cohérence, l'utilisation des couleurs, des textes sur une page, la vitesse de téléchargement, etc. Ces mesures et les classements (effectué par des experts) d'un grand ensemble de sites Web sont utilisés par cette méthodologie pour déterminer les modèles statistiques des interfaces Web de haut rang dans un classement (« highly-rated ») appelés *profils*. Ensuite, ces modèles sont utilisés pour l'analyse automatisée des pages et des sites Web d'une façon similaire avec les méthodes de revue de règles. Donc, l'auteur considère cette méthodologie comme une synthèse des techniques de mesure dans le domaine de l'évaluation de performance et des techniques de revue de règles dans le domaine de l'évaluation d'utilisabilité. Cette méthodologie est composée de deux phases qui ont les activités communes présentées sur la figure 2.10

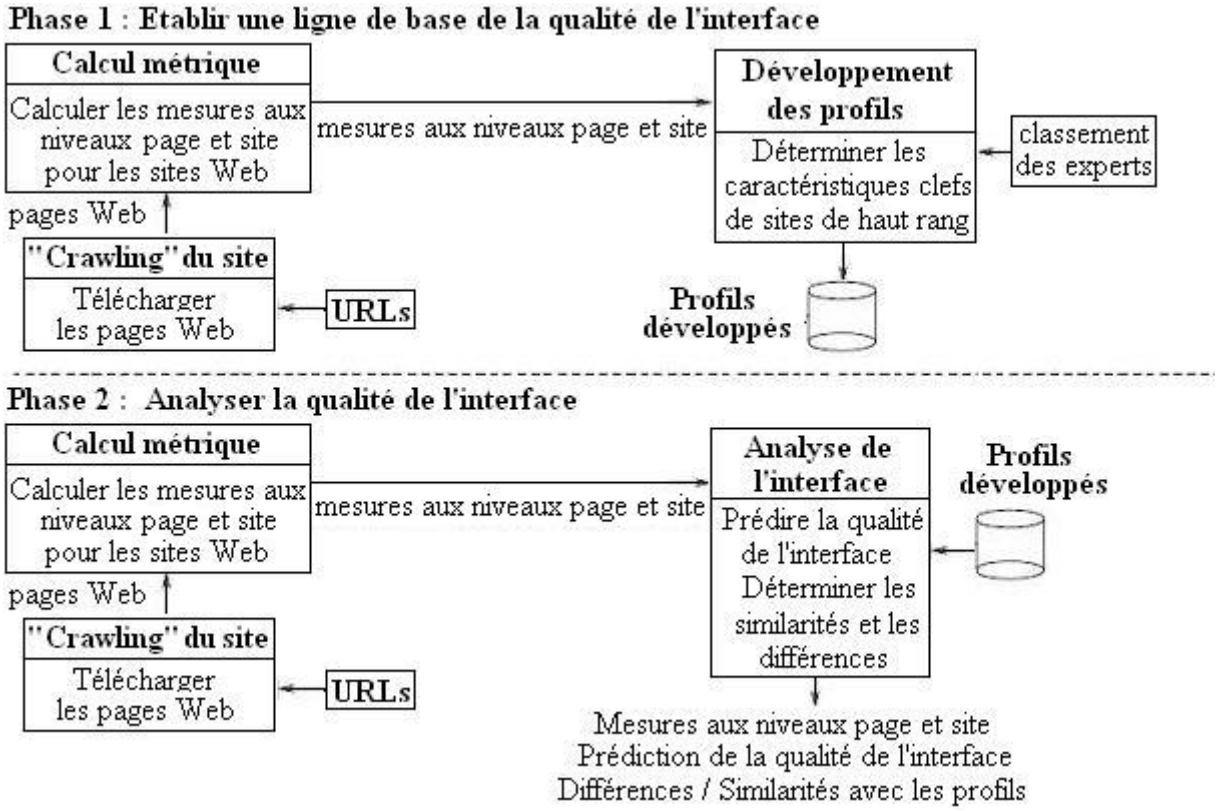

Figure 2.10 : Méthodologie d'analyse du projet WebTango, traduite de [Ivory, 2001]

La première phase vise à construire des profils. Deux activités doivent être effectuées pour atteindre cet objectif : (a) développement des modèles statistiques et (b) utilisation des modèles statistiques développés pour construire des profils.

L'activité 1 de la première phase concerne le développement de quelques modèles statistiques pour classifier les pages et les sites en 3 classes (bon, moyen, et pauvre). On peut citer ici quelques modèles représentatifs aux niveaux page et site. Les lecteurs intéressés peuvent consulter [Ivory, 2001] pour prendre connaissance des autres modèles.

Les modèles au niveau page :

- *Qualité globale de page* : ce modèle classifie une page en trois classes : bonne, moyenne ou pauvre sans considérer le type fonctionnel de la page ou la catégorie du contenu.
- *Qualité de type page* : ce modèle classifie une page en trois classes : bonne, moyenne ou pauvre en considérant le type fonctionnel de la page. Une page peut être d'un de ces types : accueil, lien, contenu, formulaire ou autre. L'outil « *Outil d'analyse* » (présenté ci-dessous) utilisé par cette méthodologie peut déterminer automatiquement le type de page mais il permet aussi à l'utilisateur de le spécifier lui-même.
- *Qualité de catégorie du contenu de la page* : ce modèle classifie une page en 3 classes : bonne, moyenne ou pauvre en considérant la catégorie du contenu du site. L'auteur a choisi six catégories de contenu : communauté, éducation, finance, santé, vie, services.

Chaque modèle détermine les mesures clefs permettant de prédire les pages considérées comme bonnes, moyennes ou pauvres. Par exemple, la taille minimum des polices, le nombre de mots en italiques, le nombre des colonnes de texte, le nombre de liens, le nombre de couleurs du corps (partie « Body » d'une page HTML), etc. Le lecteur intéressé peut consulter dans [Ivory, 2001] les 10 mesures clefs de prédiction au niveau page.

Les modèles au niveau site :

- *Qualité globale de site* : ce modèle classifie un site en trois classes : bon, moyen ou pauvre sans considérer la catégorie du contenu du site.
- *Qualité de catégorie du contenu de site* : ce modèle classifie un site en 3 classes : bon, moyen ou pauvre en considérant la catégorie du contenu du site. Les six catégories de contenu cités plus haut sont étudiées.

Ces deux modèles au niveau site ont une limitation : ils ne prennent pas en compte la qualité au niveau page parce qu'ils se basent sur un ensemble complètement différent de mesures. En conséquence, un site peut être classifié en bon site bien que toutes ses pages soient classifiées en pauvre ou vice versa. Pour remédier à ce problème, quelques modèles ont été proposés. Les lecteurs intéressés peuvent consulter [Ivory, 2001] pour les détails.

Chaque modèle détermine les mesures clefs permettant de prédire les sites qualifiés de bons, moyens ou pauvres. D'après l'auteur, l'exactitude des modèles au niveau page est 93%-96%, alors que l'exactitude des modèles au niveau site est seulement 68%-88%.

L'activité 2 de la première phase concerne le téléchargement d'un grand échantillon de pages des sites Web impliqués dans un classement *a priori*, la réalisation des classements (par les experts) et la détermination des caractéristiques clefs des sites de haut rang en utilisant les modèles statistiques développés.

Les auteurs ont utilisé un grand échantillon de pages et de sites (1002 sites) issus du Webby Awards 2000 dataset [The International Academy of Arts and Sciences, 2000]. Ces sites ont été classifiés selon 27 catégories de contenu et classés par plus de 100 juges, tous professionnels de l'Internet, se basant sur les critères : contenu, structure & navigation, conception visuelle, fonctionnalité, interactivité. L'auteur a sélectionné les sites dans les 6 catégories de contenu (listées plus haut) pour ses études. Trois groupes - bon (premier 33%), moyen (34% suivant) et pauvre (dernier 33%) ont été définis pour l'analyse en utilisant un score global.

Pour effectuer des mesures sur l'échantillon des pages des sites téléchargés, les caractéristiques des pages et des sites des trois classes (bon, moyen, pauvre) ont été identifiées. On peut citer ici quelques caractéristiques à titre d'exemples :

- A travers le *modèle de qualité globale de page*, quelques caractéristiques de pages qualifiées de bonnes pages sont précisées, par exemple : les bonnes pages utilisent des tailles minimums de polices de 9 points, elles contiennent rarement des mots en italiques dans le corps du texte (partie « Body » d'une page HTML), les bonnes pages contiennent environ 27 liens de texte alors que les pages moyennes en contiennent 22 et les pauvres 19, les bonnes pages sont plus interactives - elles contiennent environ 3 objets interactifs (pulldown menu, search button, etc.) alors que les moyennes et les pauvres en contiennent environ 2, etc.
- A travers le *modèle de qualité de type page*, quelques caractéristiques clefs des pages de chaque type sont précisées comme : le nombre de liens sur les bonnes pages d'accueil et sur les bonnes pages de liens est considérablement plus élevé, les bonnes pages d'accueil utilisent les points d'exclamation pour mettre l'accent sur un contenu quelconque tandis que les pages d'accueil pauvres utilisent les textes en italiques, etc.
- A travers le *modèle de qualité de catégorie du contenu*, quelques caractéristiques sont identifiées comme : les bonnes pages de communauté ont plus tendance à utiliser les scripts et un graphique animé et elles visent à utiliser plus de titres distincts entre les pages, les bonnes pages d'éducation ont moins tendance à utiliser les objets de type « applets », etc.
- A travers *le modèle de qualité globale de site*, quelques caractéristiques sont précisées comme : les architectures d'information des sites bons et moyens mettent l'accent sur l'étendue plutôt que la profondeur, la variation concernant les éléments de texte des bons sites est légèrement plus importante que pour les sites moyens et pauvres, etc.

Le résultat de ces activités correspond aux modèles statistiques des interfaces Web de haut rang dans le classement (appelés les profils). Il faut répéter périodiquement (annuellement ou semestriellement) cette activité pour s'assurer que les profils reflètent les pratiques de la conception Web courante.

La deuxième phase exploite les profils développés dans la première phase pour évaluer les sites Web. Comme l'illustre la figure 2.10, il faut d'abord télécharger les pages Web à évaluer. Ensuite, le calcul des mesures sur ces pages est effectué. Ces mesures sont relatives aux tailles de polices des textes, au nombre de mots en italiques, etc., sur des pages Web (déjà illustré dans les exemples ci-dessus). Enfin, ces mesures calculées sur des pages évaluées sont comparées avec les profils développés dans la première phase. Cette comparaison permet de déterminer les différences et les similarités entre les interfaces des pages évaluées et les profils, donc l'évaluateur peut prédire la qualité des interfaces (pages bonnes, moyennes ou pauvres).

Cette méthodologie est associée à quelques outils. Les principaux sont les suivants :

- Un *« Outil de crawling de sites* » permet de télécharger les pages Web dans un but de traitement. Cet outil ne télécharge pas les pages Flash, chatrooms, advertisements, etc.
- Un « *Outil de calcul métrique* » permet de traiter les pages téléchargées. Il permet d'effectuer 141 mesures au niveau page et 16 mesures au niveau site. Ces 157 mesures visent à quantifier plusieurs aspects des interfaces Web comme la cohérence entre les pages du site, l'utilisation des couleurs et des polices, le type et la quantité de texte sur une page, le graphique des pages, la vitesse de téléchargement des pages, etc. Ceci est effectué dans les deux phases (présenté ci-dessus). Après une enquête, l'auteur a identifié 62 caractéristiques (« features ») qui ont des impacts sur la qualité et l'utilisabilité des

interfaces Web. Ces 157 mesures soutiennent l'évaluation de 56 des 62 caractéristiques (90%).

- Un « *Outil d'analyse* » englobe quelques modèles statistiques (arbres de décision, fonctions de classification des discriminants, etc.) pour évaluer la qualité aux niveaux page ou site Web en utilisant les profils déjà développés et en invoquant l'« *Outil de calcul métrique* » pour générer les mesures. Le résultat de cette analyse concerne la détermination des différences et des similarités des interfaces évaluées avec les profils et la prédiction des classements des interfaces évaluées (bon, moyen ou pauvre).

Les différences entre cette méthodologie et les autres méthodes de revue de règles proviennent de :

- l'utilisation des mesures quantitatives pour quantifier plusieurs aspects des interfaces Web
- l'utilisation des calculs empiriques et classements des experts pour développer les règles. Dans cette méthodologie, on utilise le mot *profil*. Ces profils sont utilisés comme une base de comparaison entre les pages Web évaluées et les pages Web de haut rang dans le classement déjà effectué par les experts.

Grâce à ces différences, l'auteur affirme qu'il est possible d'utiliser les modèles statistiques pour valider ou invalider plusieurs règles proposées dans la littérature concernant la conception Web. Bien que cette méthodologie soit originale, elle dispose aussi de quelques limitations :

- La détermination ou suggestion des améliorations appropriées et l'implémentation de ces améliorations sont encore manuelles. L'évaluateur doit proposer au concepteur des améliorations par lui-même.
- L' *« Outil de calcul métrique »* de cette méthodologie traite seulement les pages Web en anglais. Le temps nécessaire pour réaliser les calculs est long, c'est aussi un problème de cet outil. D'après l'expérimentation des auteurs dans une entreprise ayant 450 serveurs, cet outil prend 4 minutes en moyenne de traitement de chaque page Web pour effectuer les mesures au niveau page, les mesures entre les pages et une minute supplémentaire pour calculer les mesures au niveau site. Une heure peut être prise pour un site. Le temps pris dépend aussi de la complexité des pages (tailles de pages, nombre des éléments associés, etc.)
- Cette approche est principalement développée pour les sites Web dont les tâches principales sont des localisations des informations spécifiques et où l'utilisateur peut accomplir les tâches de localisation des informations souhaitées par des chemins différents. En effet, l'utilisateur peut commencer à n'importe quel lien d'une page du site ou même à partir de liens extérieurs comme ceux proposés par le moteur de recherche, pour localiser les informations souhaitées. L'application de cette approche est limitée pour les sites orientés fonctions (où l'utilisateur doit suivre des étapes explicites pour réaliser une tâche quelconque).
- Les prédictions des différents modèles peuvent être différentes comme présenté ci-dessus. En effet, les auteurs ont utilisé ces modèles pour évaluer un site de 9 pages classifiées selon deux catégories de contenu : santé et éducation. La prédiction pour les sites de santé trouve que c'est un site pauvre dans ce domaine tandis que celle pour les sites d'éducation est contradictoire, trouvant que c'est un bon site d'éducation. A travers le *modèle de qualité de page global* et le *modèle de qualité de catégorie du contenu de santé*, toutes les 9 pages sont classées en bonnes pages mais à travers le *modèle de qualité de catégorie du contenu d'éducation*, 5 des 9 pages sont classées en pages moyennes, les 4 pages restantes

sont classées en pages pauvres. Les différences entre les prédictions des modèles peuvent causer des difficultés pour la détermination ou la suggestion des changements nécessaires. Il est difficile de satisfaire tous les modèles simultanément.

- Les auteurs ont fait une expérimentation pour déterminer le lien entre le classement des juges et celui des participants non experts. Bien qu'il y ait quelques cohérences, les auteurs ne sont pas sûr d'affimer que les classements des juges reflètent exactement l'utilisabilité.

3.2.6. Synthèse sur les méthodes étudiées

Nous avons présenté les méthodes provenant de la communauté IHM pour évaluer les systèmes interactifs en utilisant les règles ergonomiques. Il faut donc préciser que dans la communauté génie logiciel, plusieurs méthodes ont été proposées pour évaluer l'utilisabilité logicielle. Les lecteurs intéressés peuvent consulter [Folmer et Bosch, 2003 ; Golden *et al*., 2005 ; Rafla *et al*., 2007 ; Seffah *et al*., 2008].

Les tableaux 2.1 et 2.2 proposent la synthèse des méthodes examinées ci-dessus en se basant sur différentes dimensions :

- **Règles avec lesquelles la méthode peut travailler** : on y précise les règles qu'une méthode peut utiliser pour évaluer l'interface.
- **Structure et stockage des règles** : on s'intéresse à la manière utilisée par une méthode pour structurer les règles et les stocker dans la base.
- **Principe adopté pour l'accès, l'ajout, la modication et suppression des règles** : on précise les manières d'accès (à la base de règles), d'ajout (de nouvelles règles), de modification et de suppression (des règles existantes).
- **Exploitation des règles pour détecter les violations des interfaces et représentation des résultats** : on s'intéresse à la manière d'exploiter des règles d'une méthode pour déterminer si l'interface viole la règle ou non, et à la manière de représenter des règles.
- **Application des résultats d'évaluation pour améliorer l'interface** : on précise la démarche d'utilisation des résultats d'évaluation pour améliorer l'interface.
- **Quelques propriétés non fonctionnelles** : on précise les avantages, les défauts d'une méthode et/ou l'applicabilité, la performance d'une méthode, etc.

Tasbleau 2.1 : Synthèse sur les outils utilisant les règles 1

Outils	Règles avec lesquelles la méthode peut travailler	Structure et stockage des règles	Principe adopté pour l'accès, l'ajout, la modification et la suppression des règles
Système proposé par Parush [Parush, 2000]	Différents types de règles (règles générales ou provenant de guides de style spécifiques à des plateformes ou des corporations)	Chaque règle est structurée selon trois niveaux (section, sous-section, titre) et quatre parties complémentaires (recommandation, notes additionnelles, fichier d'image, liste des références aux autres règles).	On peut ajouter facilement de nouvelles règles et modifier des règles existantes. On peut aussi parcourir les règles suivant ses structures hiérarchiques, interroger la base de données pour récupérer un sous-ensemble de règles selon les besoins en se basant sur les informations liées aux règles (sections, sous-sections, titres).
Sherlock [Grammenos *et al*., 2000]	N'importe quel type de règles relatives aux interfaces WIMP (Windows, Icons, Menus, Pointing device). Les règles sont classifiées selon deux critères : *classe de règle* représentant l'origine des règles et type d'inspection pouvant prendre l(une des trois valeurs : *automatique*, *manuelle* ou *semi-automatique*. C'est-à-dire que l'évaluation de Sherlock peut être automatique, manuelle ou semi-automatique.	Chaque règle est structurée par propriétés : *clef*, *titre*, *classe de règle*, etc. Les règles sont stockées dans la base des règles de Sherlock sous un format spécifique. Cette base est indépendante de l'implémentation physique des règles. Les sources de règles et de leurs routines d'inspection sont contenues dans des modules externes DLLs.	Les modules externes DLLs peuvent résider localement ou être distribués sur le réseau. Elles peuvent être créées ou étendues par n'importe langage soutenant ActiveX. Il faut seulement déclarer le nom et la localisation des DLLs pour ajouter des nouvelles sources de règles à Sherlock. Le module serveur permet de parcourir les règles des DLLs et de choisir un sous-ensemble de règles pertinentes à utiliser. Le module client peut sélectionner quelques règles qu'il ne veut pas prendre en compte lors de l'évaluation. La seule contrainte imposée aux développeurs des règles est une interface de programmation spécifique mettant à disposition l'ensemble des routines que chaque DLL doit soutenir.
Destine [Beirekdar 2004 ; Jasselette *et al*., 2006]	Cette méthodologie travaille avec les règles relatives aux interfaces Web. Le niveau d'automatisation de l'évaluation est utilisé pour classifier les règles. Cette méthodologie ne travaille qu'avec les règles capables d'être exprimées en GDL – Guideline Definition Language (*pratiquement vérifiables*).	Les règles sont structurées sous forme XML respectant la syntaxe de GDL. La structure GDL d'une règle est composée des parties suivantes : les *ensembles d'évaluation* ; et les *logiques d'évaluation* et *de correction* associées (pour la version étendue de GDL).	La démarche pour structurer une nouvelle règle sous forme GDL est la suivante : interpréter les règles en fonction du contexte d'évaluation ; spécifier les *ensembles d'évaluation* et les *logiques d'évaluation* et de *correction* associées (pour la version étendue de GDL). Les évaluateurs peuvent structurer une règle différemmment selon leur compréhension de la sémantique originale des règles, de leur interprétation des règles, de leur connaissance de HTML, de leur expérience en GDL.
Web Tango [Ivory, 2001]	Cette méthodologie construit automatiquement des règles (dites *profils*) pour quantifier plusieurs aspects des interfaces Web. A la différence des autres méthodes de revue de règles, cette méthodologie utilise des mesures quantitatives, de même qu'un classement de sites par des experts pour développer les profils comme servant de base de comparaison	La manière de structurer et stocker les profils n'est pas présentée dans [Ivory, 2001]. Nous pensons qu'un format spécifique est utilisé pour stocker les profils.	Afin de construire des profils, un grand échantillon d'interfaces Web est téléchargé et classé par les experts. Des mesures sont effectuées sur ces interfaces pour déterminer les caractéristiques des interfaces Web de haut rang (*profils*). Il faut répéter cette activité périodiquement (annuellement ou semestriellement) pour assurer que les profils reflètent les pratiques de la conception Web courante.

Tableau 2.2 : Synthèse sur les outils utilisant les règles 2

Outils	Exploitation des règles pour détecter les violations des interfaces et représentation des résultats	Application des résultats d'évaluation pour améliorer l'interface	Quelques propriétés non fonctionnelles
Système proposé par Parush [Parush, 2000]	C'est seulement un système de collection de règles. Les évaluateurs consultent ce système pour connaître et étudier les règles pour effectuer leurs tâches d'évaluation.	Il ne fournit aucune suggestion relative à l'amélioration du système.	Le système est facile à comprendre et à utiliser. La création et la modification des règles n'exigent pas de connaissance spécifique, Ce système est accessible sans besoin de lancer un environnement particulier.
Sherlock [Grammenos *et al*., 2000]	Le module serveur invoque les routines d'inspection dans les DLLs pour évaluer la description sous forme de structure d'arbre de l'interface, provenant du client Sherlock. Un rapport contenant la liste des violations d'utilisabilité détectées (avec suggestion de correction pour chaque violation) est renvoyé au client par le serveur. L'utilisateur du client peut parcourir cette liste pour prendre connaissance des détails de chaque violation. Sherlock fournit 3 types de détection des violations : automatique, semi-automatique, manuelle.	L'utilisateur du client doit corriger l'interface de lui-même (correction manuelle). Il peut donc suivre les suggestions fournies par Shelock ou non. Il peut aussi consulter l'historique des solutions précédentes pour le même problème pour l'aider à prendre la décision. Les corrections qu'il a effectuées peuvent être stockées dans l'historique des solutions à ce problème.	1) La séparation entre le serveur Sherlock et les fournisseurs des règles assure une indépendance de langage ainsi qu'un mécanisme flexible et efficace pour la mise à jour. 2) Les fournisseurs des règles doivent avoir une connaissance de haut niveau sur la programmation des ActiveX DLLs. 3) L'interface du client Sherlock est facile à utiliser. L'utilisateur appuie sur le bouton *Evaluate* et attend le rapport d'évaluation. 4) L'applicabilité de Sherlock est limitée parce que le client Sherlock peut seulement traiter des interfaces utilisateur créées à l'aide de Visual Basic 5.0.
Destine [Beirekdar 2004], [Jasselette *et al*., 2006]	1) La version initiale de GDL permet seulement d'évaluer automatiquement des éléments HTML (pas CSS). La version étendue de GDL permet d'évaluer automatiquement tous les deux. 2) Après avoir structuré et chargé les règles, la page Web à évaluer est analysée pour capturer les éléments HTML, CSS à évaluer. 3) le moteur d'évaluation applique les *conditions d'évaluation* des règles en GDL aux éléments capturés pour déterminer s'ils violent ou respectent les règles. 4) Ce moteur génère un rapport d'évaluation.	Un rapport d'évaluation contenant les résultats (respects et violations détectées) est généré. Ensuite, une correction semi-automatisée est fournie pour les éléments HTML (pas pour CSS) dans la version étendue de GDL. Cette correction vise à afficher une interface personnalisée permettant à l'utilisateur de saisir les valeurs nécessaires pour corriger l'élément HTML violant une règle quelconque.	1) Cette méthodologie évalue chaque page Web individuellement. D'après les auteurs, il faut seulement collecter et évaluer toutes les pages d'un site pour évaluer la qualité de ce site mais la démarche n'a pas été clairement présentée. 2) La séparation entre les règles et le moteur d'évaluation apporte de la flexibilité. 3) La structuration des règles exige de l'utilisateur une bonne compréhension de la sémantique originale des règles, une bonne connaissance de HTML, une riche expérience en GDL, une qualité élevé de l'interprétation des règles. La qualité d'évaluation dépend de ces facteurs et de la richesse de GDL.
WebTango [Ivory, 2001]	Cette méthodologie permet des mesures nécessaires et relatives à plusieurs aspects des interfaces Web (taille de police des textes, nombre de mots en italiques, etc.) sur les pages Web à évaluer. Puis, celles-ci sont comparées avec les profils déjà construits pour déterminer les différences et similarités entre elles. Donc, l'évaluateur peut prédire la qualité de l'interface des pages (bonnes, moyennes ou pauvres)	La suggestion des améliorations et leur implémentation sont manuelles.	1) Cette approche est principalement développée pour les sites Web orientés informations. 2) Elle traite seulement les pages Web en anglais. 3) Le temps nécessaire pour calculer les mesures concernant une page Web est long. 4) l'exactitude des modèles, au niveau site, n'est pas élevée. 5) Les différences entre les prédictions des modèles peuvent causer des difficultés pour la suggestion d'améliorations à apporter.

3.3. Outils de recueil des données d'interaction entre l'utilisateur et le système interactif pour l'aide à l'évaluation (mouchards électroniques)

3.3.1. Introduction

Un mouchard électronique est un outil de capture automatique de données objectives durant l'interaction entre l'utilisateur et le système interactif en situation de travail réelle. Les données capturées sont analysées après d'une manière manuelle, ou automatique [Bastien et Scapin, 2002 ; Ezzedine et Abed, 1997 ; Ezzedine et Trabelsi, 2005 ; Mariage, 2005]. Il faut préciser quelques différences entre les mouchards électroniques et quelques outils similaires :

- Le mouchard électronique est différent d'un outil dit de monitoring. Ce dernier surveille ou capture seulement des données sans proposer une analyse quelconque ultérieure. Le mouchard électronique permet de capturer, d'enregistrer des données et il peut proposer aussi une analyse plus au moins approfondie des données capturées [Mariage, 2005].
- Le mouchard électronique est aussi différent des QFA (Quality Feedback Agents) qui sont des logiciels utilisés pour capturer des données relatives à l'historique d'une application dans le cas où celle-ci rencontre des problèmes de fonctionnement. Le QFA se contente de capturer des données techniques (par exemple, la version du système d'exploitation, le type du processeur, le type d'écran, les registres, les fonctions appelées juste avant l'échec de l'application, etc.) au moment où l'application a eu un problème. Ces données sont envoyées à l'équipe de développement pour l'aider à comprendre le problème et la raison de l'échec de l'application et à améliorer la version future de l'application. Le QFA peut aussi permettre à l'utilisateur d'envoyer le rapport sur ce qu'il faisait avec l'application lors de l'échec, à l'équipe de développement de celle-ci. Le QFA est complétement différent des mouchards électroniques qui capturent les interactions entre l'utilisateur et l'application en situation de travail réelle, dans un but d'analyse ultérieure. En conséquence, pour l'évaluation des systèmes interactifs, le QFA est limité par rapport aux mouchards électroniques [TRAN *et al*., 2008a ; b].

Le principe du fonctionnement d'un mouchard électronique est illustré en figure 2.11. Il se compose de trois étapes :

- D'abord, le mouchard capture les données provenant de l'interaction entre l'utilisateur et le système en situation de travail (par exemple, les actions des utilisateurs et leurs répercutions sur le système comme l'appui sur une touche du clavier, sur un bouton de la souris, la sélection d'un objet dans un menu, un clic sur un bouton de la fenêtre d'interface, l'apparition d'un message d'avertissement, l'ouverture ou la fermeture d'une fenêtre, etc.). Cette capture de données est faite de façon discrète, transparente pour que l'utilisateur de l'application ne se sente pas gêné par le fonctionnement du mouchard.
- Les données capturées sont stockées dans une base de données et ensuite, analysées par le mouchard afin d'aider l'évaluateur dans ses activités. Les analyses peuvent être le résultat de calculs divers (ex. statistiques) et elles peuvent être fournies à l'évaluateur sous des formes différentes (diagrammes, textes, tableaux, etc.)
- Grâce aux enregistrements obtenus, l'évaluateur peut reconstituer des modèles de l'activité de l'utilisateur et les réactions de l'IHM. Ces modèles sont appelés modèles observés et ils peuvent être comparés à des modèles des tâches à effectuer qui sont déjà spécifiées par le concepteur [Ezzedine et Abed, 1997]. Le résultat de ces comparaisons peut être utile au concepteur pour améliorer le système interactif. Les techniques d'interview et de questionnaire peuvent être utiliséees pour mieux comprendre les activités ainsi que les motivations des utilisateurs quand ils ont utilisé l'application.

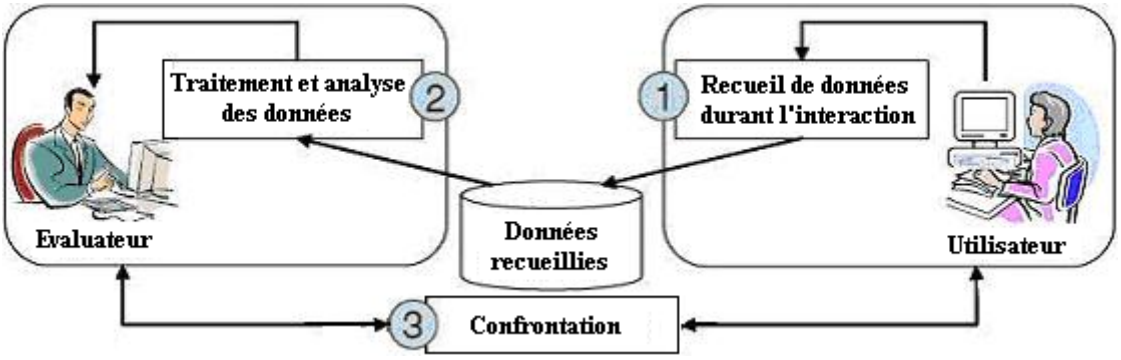

Figure 2.11 : Principe de fonctionnement du mouchard [Ezzedine et Abed, 1997]

Le point fort du mouchard électronique est sa transparence vis-à-vis de l'utilisateur ; le travail de ce dernier n'est pas perturbé lors de ses activités. Les points faibles du mouchard électronique résident dans le traitement des données capturées et stockées, et aussi dans le niveau de granularité de l'analyse [Bastien et Scapin, 2002]. En effet, la quantité de ces données peut être énorme et l'analyse peut prendre des heures, voire des journées. En conséquence, le mouchard doit fournir des analyses des données pour faciliter le travail de l'évaluateur. Sinon, les données capturées et stockées peuvent demander plus de temps pour tirer des conclusions utiles au concepteur pour l'amélioration.

Il est important de souligner que, cette méthode capture directement seulement les données objectives. Elle ne peut pas capturer directement des informations subjectives comme les préférences, l'opinion de l'utilisateur, etc. [Ivory, 2001]. En plus, il faut impliquer les utilisateurs réels pour appliquer cette méthode d'évaluation ; en conséquence, elle est coûteuse mais fiable si on se réfère au point de vue de l'utilisateur [Lecerof et Paterno, 1998].

Nous examinons ci-dessous quelques mouchards électroniques représentatifs parus depuis une dizaine d'années, sans souci d'exhaustivité. Les lecteurs intéressés peuvent consulter [Hilbert et Redmiles, 2000] pour prendre connaissance d'outils plus anciens.

3.3.2. USINE (USer INterface Evaluator)

Dans [Lecerof et Paterno, 1998], une méthode a été proposée pour évaluer des interfaces de type WIMP (Window, Icons, Menus, Pointing device). Elle analyse les actions physiques de l'utilisateur sur l'interface à partir d'informations stockées dans des fichiers textuels « log » en utilisant le modèle de tâches CTT (ConcurTaskTrees) [Paterno *et al*., 1997]. La figure 2.12 illustre cette méthode.

a) Modèle de tâches, table de préconditions et tâches disponibles

D'abord, dans un but d'évaluation, le concepteur doit spécifier le *modèle de tâches* de l'interface utilisateur en utilisant la notation CTT (ConcurTaskTrees) avec un éditeur graphique associé. Ce modèle de tâches correspond à une structure hiérarchique partant d'un haut niveau d'abstraction jusqu'à un niveau bas ; il permet au concepteur de spécifier différentes relations temporelles entre les tâches : activer, désactiver, synchronization, choix, etc. Les lecteurs intéressés peuvent consulter [Paterno *et al*., 1997] pour les détails de ce modèle.

Parmi les relations entre les tâches spécifiées dans ce modèle, on trouve les relations : « activer » ou « désactiver », c'est-à-dire que quelques tâches peuvent activer ou désactiver les autres tâches. Autrement dit, une tâche quelconque T peut avoir une ou plusieurs autres tâches (comme ses *tâches de précondition*) qui doivent être réalisées avant que la tâche T

puisse être réalisée. En conséquence, une *table de préconditions* est automatiquement créée à partir de ce modèle de tâches. Cette table contient la liste des tâches avec leurs *tâches de précondition* associées ainsi que l'information concernant les satisfactions de ces *tâches de précondition*. Au début de la session utilisateur, toutes ces *tâches de précondition* ne sont pas bien sûr satisfaites. Durant la session, chaque fois que l'utilisateur réussit à réaliser une *tâche de précondition*, alors cette table est mise à jour. A partir de cette liste, la méthode d'évaluation peut récupérer la *liste des tâches disponibles* que l'utilisateur peut réaliser à l'état courant de la session utilisateur. Une tâche est disponible si toutes ses *tâches de précondition* sont satisfaites. Cette *liste des tâches disponibles* est mise à jour durant la session utilisateur.

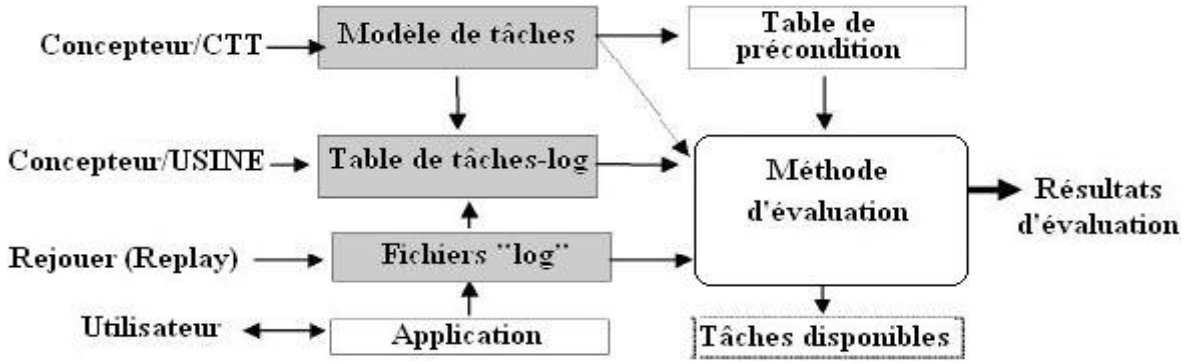

Figure 2.12 : Méthode d'évaluation USINE en utilisant le modèle de tâches CTT, traduite de [Lecerof et Paterno, 1998]

b) Fichiers « log » et Table de tâches-log

Les fichiers « log » sont des simples fichiers textuels stockant les actions physiques de l'utilisateur sur l'interface. Dans cette méthode, l'auteur doit installer les logiciels supplémentaires (JDK, QC/Replay) sur la machine client pour enregistrer ces actions de l'utilisateur durant une session. QC/Replay capture et enregistre les données correspondant aux événements liés aux dispositifs d'interaction (appuis ou relâchements des boutons de la souris, appuis sur les touches du clavier) et aux événements qui se sont produits sur les « widgets » de l'interface (clic sur un bouton, saisie de texte dans une boîte de texte, etc.). Ces informations ainsi que les informations concernant (coordonnées de la souris, temps, nom et contenu des « widget » affectés, etc.), sont enregistrées dans les fichiers utilisés en tant qu'entrée de l'outil USINE.

Quand le concepteur spécifie le modèle de tâches, il doit décomposer celui-ci jusqu'au niveau le plus bas possible pour que chaque tâche élémentaire de ce modèle corresponde à une seule action physique dans le fichier « log ». L'outil USINE fournit au concepteur et/ou à l'évaluateur une interface leur permettant d'associer les tâches élémentaires dans le modèle de tâches aux actions physiques correspondantes dans le fichier « log » pour créer la *table de tâches-log*. Si une action n'est pas associée à une tâche élémentaire, alors cette action est considérée comme une *erreur de l'utilisateu*r. Il y a deux types d'erreur de l'utilisateur qui seront présentés ci-dessous. Si quelques tâches ne sont pas réalisées dans la session considérée pour l'association mais le sont dans une autre session, alors l'outil USINE permet de mettre à jour la *table de tâches-Log*.

c) Méthode d'évaluation de l'outil USINE

Après avoir permis au concepteur et/ou à l'évaluateur la création de la *table de tâches-log*, l'outil USINE continue à traiter le fichier « log ». Afin de traiter une action quelconque stockée dans le fichier « log », l'outil USINE détermine si cette action considérée existe dans

la *table de tâches-log* (c'est-à-dire que USINE détermine si cette action est associée à une tâche élémentaire) :

- Si c'est le cas, USINE consulte sa tâche associée. Si cette tâche est associée à des *tâches de préconditions*, alors il faut les vérifier. Si toutes ces préconditions ne sont pas encore satisfaites, alors une *erreur de précondition* survient. Sinon, cette tâche est considérée comme une tâche accomplie. Si cette tâche accomplie est une précondition d'une autre tâche, alors, la *table de préconditions* est mise à jour, puis la *liste des tâches disponibles* à ce moment-là est récupérée.
- Sinon, alors cette action est marquée comme une *erreur de l'utilisateur*.

Ensuite, l'action suivante du fichier « log » est traitée, etc., jusqu'à la dernière action.

d) Résultats d'évaluation

Les résultats concernent :

- Les tâches accomplies, les tâches échouées (parce que leurs préconditions ne sont pas satisfaites) et les tâches qui ne sont jamais réalisées.
- Deux types d'erreurs de l'utilisateur :
    - *les erreurs de précondition* (l'utilisateur a réalisé des séquences de tâches violant les contraintes temporelles spécifiées dans le modèle de tâches. Une tâche est réalisée lorsque ses préconditions ne sont pas satisfaites). Ce type d'erreur est une raison d'échec des tâches.
    - *les erreurs de l'utilisateur* concernent les actions de l'utilisateur qui n'existent pas dans les tâches élémentaires associées. Ce sont des actions inutiles.
- Les informations numériques et temporelles relatives aux tâches et aux erreurs ; par exemple, le nombre de fois que les tâches sont réalisées, le nombre d'erreurs commises, l'ordre temporel entre les tâches et les erreurs.
- Le temps pris par chaque tâche: on peut le calculer en utilisant le temps de chaque action élémentaire. Le temps des tâches abstraites est calculé en effectuant le somme des temps des tâches filles.
- Les instants où les erreurs sont commises.
- Les patterns de tâches : ce sont les séquences de tâches accomplies qui apparaissent plusieurs fois.
- La liste des tâches disponibles à l'état courant de la session (cf. ci-dessus).
- Les autres informations utiles comme le nombre d'apparitions de « scroll bars »[9] et de redimensionnements de fenêtre.

e) Utilisation des résultats pour l'évaluation de l'interface

Les résultats provenant de USINE peuvent être exploités différemment :

- Les informations concernant l'accomplissement des tâches sont utiles pour déterminer si les objectifs de l'utilisateur sont atteints. L'évaluateur peut le constater sous forme de l'ordre temporel des tâches dans l'outil USINE.
- Les fréquences des tâches sont utiles pour déterminer si l'apparence (« layout ») courante de l'interface est efficace. Pour les tâches souvent réalisées, il faut que les « widgets »

[9] « Scoll bars » : un événement apparaît dès que l'utilisateur monte ou descend l'ascenceur de la fenêtre.

correspondants puissent être regroupés pour réduire les mouvements de la souris, donc l'interface est, dans ce cas, plus efficace selon [Sears, 1995] ; par contre, dans le cas de tâches rarement ou jamais réalisées, il faudrait que les « widgets » correspondants puissent être cachés, voire éliminées.

- Les informations concernant les erreurs aident l'évaluateur à découvrir si l'utilisateur réalise facilement les tâches et quelles tâches lui causent des problèmes. Avec les *erreurs de précondition*, l'évaluateur peut interpréter que l'interface impose des contraintes illogiques ; par exemple, supposons que l'ordre séquentiel soit trop rigide entre deux tâches bien qu'il n'y ait pas de dépendance logique entre elles, en conséquence, l'évaluateur peut suggérer un changement visant un modèle de tâches plus flexible à ce sujet. Avec les erreurs liées aux actions inutiles de l'utilisateur, l'évaluateur peut interpréter que l'interface n'est pas efficace et qu'elle cause des problèmes d'interprétation à l'utilisateur. L'évaluateur peut conseiller au concepteur d'y ajouter des labels plus explicatifs.
- Les instants où les erreurs sont commises sont utiles ; par exemple, les erreurs sont seulement commises au début de l'interaction, l'évaluateur peut interpréter qu'il faut fournir plus d'aide à l'utilisateur au début de l'interaction.
- Les informations concernant les patterns des tâches sont utiles pour identifier les « macros » qu'il faudrait implémenter ou les automatisations qu'on pourrait fournir.
- Les informations concernant le nombre d'événements (de type « scroll bars ») et le nombre de fois que la fenêtre est redimensionnée sont utiles pour examiner les problèmes de lisibilité.

Il s'agit de préciser que les résultats sont affichés sous forme textuelle ou graphique. L'evaluateur les utilise pour interpréter les problèmes et suggérer des améliorations.

3.3.3. RemUSINE (Remote USer INterface Evaluator)

Les outils USINE (déjà décrit ci-dessus), RemUSINE, WebRemUSINE, Multimodal WebRemUSINE, Multidevice WebRemUSINE (que nous allons présenter ci-dessous) sont développés par la même équipe. Tous utilisent le modèle de tâches CTT et exploitent des données empiriques concernant les actions physiques de l'utilisateur sur l'interface pour l'évaluation.

[Paterno et Ballardin, 1999 ; 2000] ont proposé la méthode RemUSINE pour évaluer des interfaces de type WIMP. Son fonctionnement est similaire à USINE, tout en soutenant la possibilité d'évaluation à distance. Elle vise à évaluer les applications interactives qui peuvent être utilisées par plusieurs utilisateurs et localisées dans les endroits différents. La figure 2.13 illustre cette méthode.

Il y a une différence notable entre la RemUSINE et USINE : bien que les deux méthodes permettent d'associer les tâches élémentaires dans le modèle de tâches aux actions physiques correspondantes dans le fichier « log » pour créer la *table de tâches-log*, dans le cas de RemUSINE :

- Plusieurs fichiers « log » peuvent être utilisés pour créer cette *table de tâches-log*,
- une tâche élémentaire peut être associée à une ou plusieurs actions physiques mais une action physique donnée est toujours associée à une seule tâche élémentaire.

Il faut seulement réaliser l'association une seule fois pour évaluer une application. Cette association est un point clef de la méthode RemUSINE.

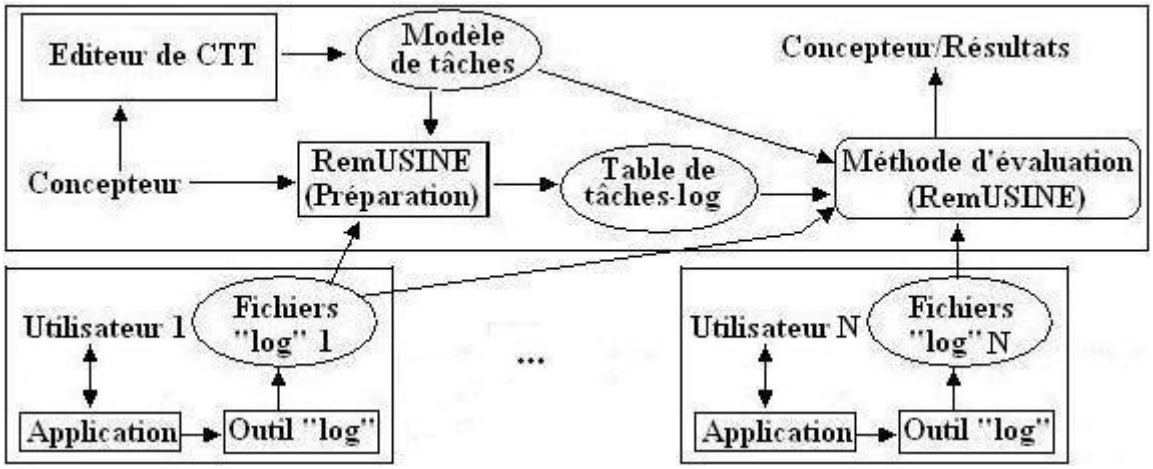

Figure 2.13 : Méthode d'évaluation RemUSINE [Paterno *et al*., 1999 ; Paterno et Ballardin, 2000]

L'analyse des actions dans des fichiers « log » peut être réalisée sur une session individuelle ou un groupe de sessions pour déterminer si un problème quelconque est souvent rencontré par plusieurs utilisateurs ou s'il est limité à des utilisateurs spécifiques dans une circonstance particulière.

3.3.4. WebRemUSINE (Web REMote USer INterface Evaluator)

Exploitant des principes similaires à USINE et RemUSINE, la méthode WebRemUSINE [Paganelli et Paternò, 2002 ; 2003] vise à évaluer des interfaces Web. Elle utilise aussi le modèle de tâches CTT et les fichiers « log » (stockant les événements qui se sont produits sur le navigateur Web) pour son évaluation. La figure 2.14 illustre cette méthode.

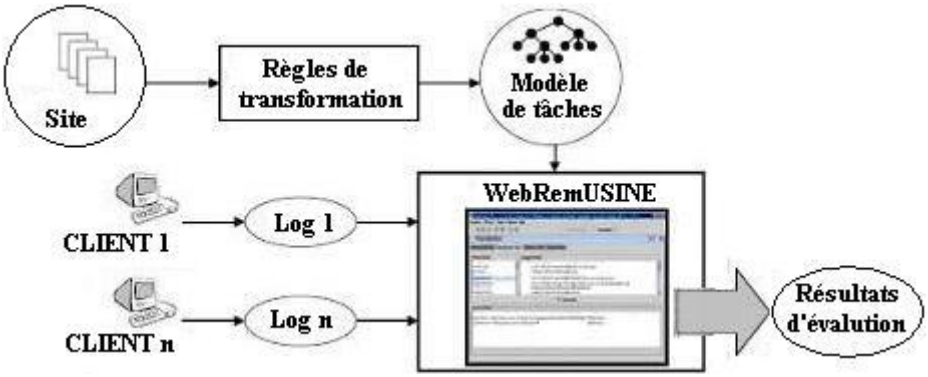

Figure 2.14. Méthode d'évaluation WebRemUSINE, traduit de [Paganelli et Paternò, 2002 ; 2003]

Les méthodes RemUSINE et USINE, pour évaluer des interfaces de type WIMP, utilisent les relations « activer », « désactiver » entre les tâches spécifiées dans le modèle de tâches CTT pour détecter les *erreurs de précondition* (c'est-à-dire les actions de l'utilisateur violant ces relations). D'après les auteurs, dans le cas de pages Web, il est difficile d'identifier automatiquement l'intention de l'utilisateur afin de détecter automatiquement ces erreurs. C'est pourquoi les auteurs proposent une solution qui oblige l'utilisateur à sélectionner exactement la tâche cible qu'il veut réaliser à partir d'une liste de tâches de haut niveau. Ces tâches de haut niveau sont soutenues par le site Web évalué et elles sont toujours affichées à l'utilisateur dans sa session (pour cela, on sépare une page Web en deux « frames » : un pour l'affichage de la liste des tâches de haut niveau et un autre pour l'affichage du contenu de page). L'utilisateur peut changer la tâche cible durant sa session. La fin d'une session est explicitement indiquée par la sélection de l'utilisateur du bouton « Stop » dans le frame des

tâches de haut niveau. Dans cette méthode, trois types d'événement sont enregistrés dans les fichiers « log » :

- Les événements provenant des interactions de l'utilisateur sur le navigateur Web. Ces événements correspondent aux tâches d'interaction du modèle de tâches CTT.
- Les événements intérieurs du navigateur Web (chargement d'une page, envoi d'un formulaire, etc.). Ces événements correspondent aux tâches système (dites aussi tâches automatiques) du modèle de tâches CTT.
- Les événements concernant le changement de tâche cible de l'utilisateur.

Comme avec RemUSINE, une tâche élémentaire peut être associée à une ou plusieurs actions physiques mais une action physique donnée peut correspondre à une seule tâche élémentaire.

Notons que dans cette méthode WebRemUSINE, on doit aussi installer JDK au minimum sur la machine client pour que Java Applet puisse fonctionner. En effet, un outil dit de « logging » utilise Javascript et Applet pour enregistrer les événements. Le code Javascript est ajouté aux pages HTML et ensuite exécuté par le navigateur. Ce code Javascript capture tous les événements et les communique à Java Applet. En fin de session (clic de l'utilisateur sur le bouton « Stop »), ces informations sont enregistrées dans des fichiers « log » qui sont transmis au serveur pour des traitements ultérieurs. Quelques informations supplémentaires sont insérées dans chaque fichier « log » comme : la date, l'heure de réception du fichier, l'adresse IP et le nom de la machine du client, le nom de l'utilisateur, etc. Cet outil fonctionne sur deux navigateurs (Microsoft IE et Nescape Communicator).

Cette méthode peut détecter des problèmes similaires à ceux détectés avec RemUSINE et USINE, comme les résultats concernant les réalisations des tâches (réussite, échec), les erreurs commises (deux types d'erreurs) ainsi que les informations statistiques les concernant ; mais elle peut aussi détecter d'autres problèmes concernant les pages Web ainsi que proposer les informations statistiques associées :

- les pages visitées et le nombre d'accès, de même que le nombre moyen d'accès à chaque page ;
- les patterns de visite des pages (les séquences de pages visitées qui apparaissent plusieurs fois) durant la navigation et leur fréquence ;
- le temps pris pour visiter ou télécharger chaque page, de même que le temps moyen pris pour visiter ou télécharger une page.

Les analyses concernant les pages Web sont utiles. Par exemple, si le temps pris pour télécharger une page est très long, alors l'évaluateur peut interpréter que la taille de ses éléments constitutifs est à vérifier. Avec une page rarement visitée, l'évaluateur peut interpréter que cette page est ennuyeuse ou difficile à atteindre. Si le temps pris pour visiter une page est trop long, l'évaluateur peut interpréter que cette page est très intéressante ou qu'elle contient trop d'information ou que son contenu est difficile à comprendre, etc.

Nous pensons que cette méthode peut être utilement appliquée aux applications Web dont l'objectif vise à fournir des fonctionnalités métier spécifiques dans un domaine quelconque. Dans de telles applications, pour réaliser une tâche métier spécifique, il faut que l'utilisateur suive rigoureusement les étapes déjà explicitement prédéfinies. Dans ce cas, on peut spécifier rigoureusement un modèle de tâches pour l'application à évaluer. Mais pour les sites Web dans lesquels il s'agit seulement de chercher des informations et d'accéder aux pages souhaitées, il n'est pas sûr que cette méthode soit applicable. En effet, dans de tels sites, pour

l'accès à une page d'information cible donnée, l'utilisateur peut commencer n'importe où, à n'importe quelle page dans le site ou même à l'extérieur du site (par exemple, à partir d'une page extérieure, à l'aide d'un moteur de recherche) et ensuite choisir un des différents chemins possibles à suivre afin d'arriver à la page d'information cible [Ivory, 2001]. En conséquence, si le concepteur veut spécifier le modèle de tâches pour une tâche quelconque concernant l'accès à une page d'information cible donnée, il doit spécifier tous les chemins possibles pour arriver à cette page dans ce modèle de tâche. Ce travail peut surcharger le concepteur, il peut être difficile, voire impossible à réaliser, le nombre de chemins possibles pouvant être très élevé.

3.3.5. Multimodal WebRemUSINE (MM WebRemUSINE)

Dans le prolongement de WebRemUSINE, la méthode Multimodal WebRemUSINE [Paterno *et al*., 2006] vise à évaluer des interfaces Web en utilisant le modèle de tâches CTT et en intégrant les données de plusieurs sources comme les fichiers « log » stockant les événements qui se sont produits sur le navigateur Web, la vidéo (par Web cam) et un oculomètre (eye-tracker) pour que l'évaluateur puisse disposer d'informations les plus complètes possible pour l'interprétation, l'analyse et l'évaluation de l'interface Web concernée. La Web cam et l'oculomètre sont supposés disponibles dans l'environnement client pour capturer d'une part les données vidéo, et d'autre part celles concernant les mouvements oculaires de l'utilisateur. La figure 2.15 illustre cette méthode :

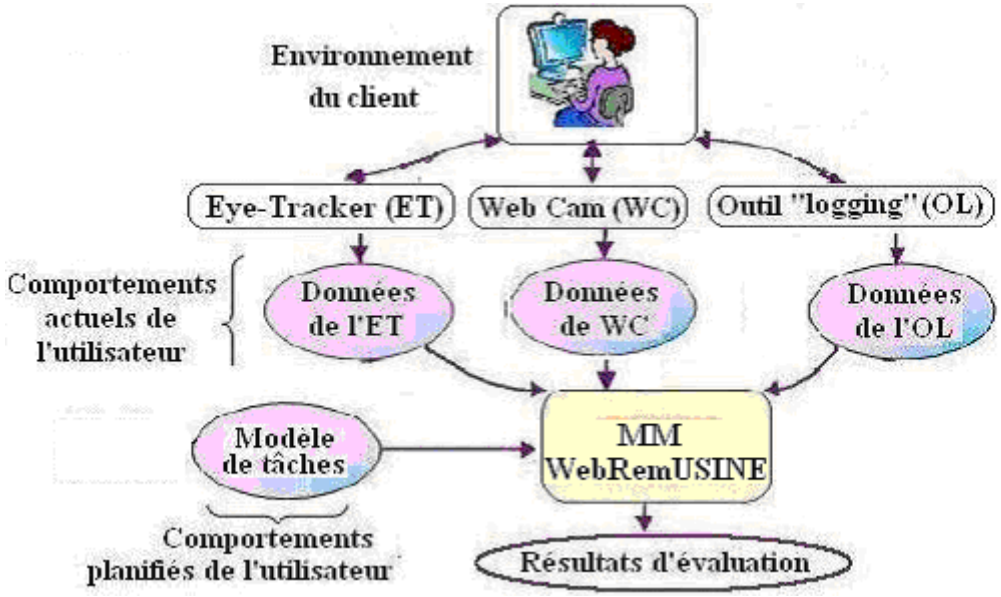

Figure 2.15 : Méthode d'évaluation MultiModal WebRemUSINE, traduite de [Paterno *et al*., 2006]

a) L'oculomètre fournit à l'évaluateur les données quantitatives concernant la direction du regard de l'utilisateur durant la session d'évaluation et ses fixations oculaires. Les fixations consistent en une série de points de consultation. L'oculomètre fournit le nombre de fixations et les durées des fixations. Le chemin oculaire est la trace du regard de l'utilisateur. Les données de l'oculomètre peuvent aider l'évaluateur à comprendre la stratégie de navigation de l'utilisateur durant sa visite de chaque page Web ; elles peuvent permettre à l'évaluateur d'identifier les zones des pages qui attirent ou pas son attention. Les fixations longues peuvent être interprétées de différentes manières : l'utilisateur est attiré par les informations correspondantes, ou il a des difficultés pour comprendre l'information, etc. Un grand nombre de fixations sur un espace quelconque peut être interprété aussi différemment : l'utilisateur est

désorienté, il ne peut pas trouver l'information souhaitée, etc. Un long chemin oculaire peut permettre de mettre en évidence que la structure d'une page est complexe.

b) L'évaluation est réalisée à distance, l'évaluateur n'étant pas sur le même lieu et poste de travail que l'utilisateur. En conséquence, l'évaluateur ne peut pas connaître les conditions de travail de l'utilisateur. Pendant que l'utilisateur utilise l'application, un incident peut se produire ; par exemple : l'utilisateur reçoit un appel téléphonique pendant qu'il utilise l'application, en conséquence, cet incident peut le perturber dans son cheminement. L'évaluateur doit en prendre connaissance pour effectuer des évaluations efficaces. La vidéo permet à l'évaluateur de connaître les changements de condition de travail. et aussi d'enregistrer les commentaires oraux de l'utilisateur sur l'interface ou une fonctionnalité quelconque de l'application, etc. Ces types d'information sont appelées les informations contextuelles de l'utilisateur.

c) L'évaluateur utilise les données de ces sources pour interpréter, analyser et évaluer l'interface. La corrélation entre les données des fichiers « log » et les données de la vidéo est assurée par les périodes de temps où l'utilisateur réalise les tâches. Une association entre les tâches réalisées et la vidéo est automatiquement réalisée par l'outil en utilisant ces périodes de temps. L'évaluateur peut étudier aussi les trajectoires et les fixations du regard de l'utilisateur sur l'interface pendant qu'il réalise ses tâches en utilisant les données oculométriques. Toutes ces données sont automatiquement intégrées dans l'outil ; pour chaque tâche réalisée, l'outil peut en effet fournir les données vidéo et oculomètriques associées.

Cette méthode MultiModalUSINE est plus coûteuse que les autres de la famille USINE (USINE, RemUSINE, WebRemUSINE) parce qu'on doit intégrer et utiliser des dispositifs supplémentaires (WebCam, oculomètre) dans l'environnement des utilisateurs.

3.3.6. Multi Device RemUSINE

Dans le prolongement des autres outils de la famille USINE, la méthode MultiDevice RemUSINE [Paterno *et al*., 2007] vise à évaluer à distance les applications sur dispositifs mobiles. Elle utilise aussi le modèle de tâches CTT et les fichiers « log » stockant les événements qui se sont produits sur l'interface pour son évaluation. La figure 2.16 illustre cette méthode.

D'après les auteurs, pour disposer de la description complète de ce que l'utilisateur a effectué durant la session, il faut non seulement capturer les interactions entre l'utilisateur et le système, mais aussi capturer les informations contextuelles de l'utilisateur. Ces informations concernent le contexte de travail de l'utilisateur ; elles influencent beaucoup la performance et les activités de celui-ci. Si ces informations contextuelles ne sont pas capturées, il est difficile à l'évaluateur de tirer les conclusions exactes concernant les problèmes que l'utilisateur a rencontrés. Dans le cas des applications mobiles, la capture des informations contextuelles est bien plus difficile que dans celui des applications de type desktop. En effet, dans le cas des applications de type desktop, on peut utiliser certains dispositifs, comme la webcam, dans l'environnement pour capturer de telles informations contextuelles. Donc, les auteurs proposent une autre solution qui utilise l'outil « Mobile Logger » pour détecter les interactions entre l'utilisateur et les applications mobiles en prenant connaissance des conditions environnementales capables d'affecter la performance et les activités de l'utilisateur.

Les informations contextuelles pouvant être capturées par l'outil Mobile Logger sont : la localisation de l'utilisateur, le bruit et la lumière de l'environnement, la capacité de la batterie du dispositif mobile sur lequel l'application fonctionne, la puissance du signal réseau. Quand cet outil est activé, il commence par demander à l'utilisateur de s'identifier, puis de choisir les

informations contextuelles considérées, et enfin de spécifier la tâche cible qu'il vise à atteindre par sélection dans une liste de tâches de haut niveau. Les évenements capturés par l'outil « Mobile Logger » sont catégorisés en différentes classes : événements concernant les interactions de l'utilisateur durant la session, événements système générés pour répondre aux interactions de l'utilisateur, événements d'intention concernant les changements de tâches cibles de l'utilisateur. Toutes ces informations sont enregistrées dans des fichiers XML. Cet outil « Mobile Logger » communique avec MultiDevice RemUSINE à travers ces fichiers.

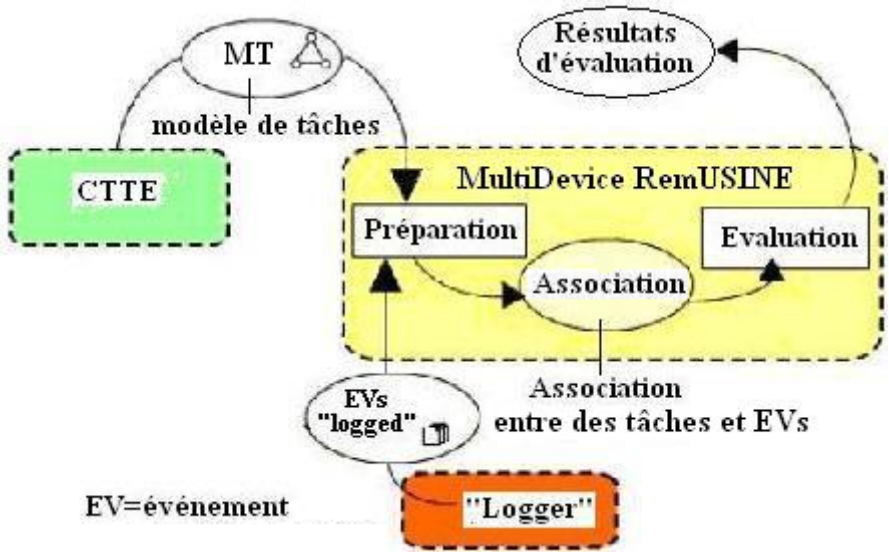

Figure 2.16 : Méthode d'évaluation MultiDevice RemUSINE, traduite de [Paterno *et al*., 2007]

D'après les auteurs, la capture des informations contextuelles dans lesquelles l'utilisateur évolue aide l'évaluateur à comprendre les conditions affectant ses activités.

3.3.7. WET

[Etgen et Cantor, 1999] ont proposé l'outil WET (Web Event-logging Tool) qui capture les événements se produisant sur une interface Web à évaluer.

WET fonctionne sur deux navigateurs Internet Explorer (IE) et Nescape Communicator. Il peut capturer les événements d'interaction de l'utilisateur sur l'interface Web comme le clic, le double clic sur la souris, les boutons, etc., les événements du navigateur comme le chargement d'une page, etc. Les informations associées à ces événements comme les coordonnées sur l'écran et sur la fenêtre du navigateur (pour les événements concernant la souris), le temps d'apparition ou la source d'événement sont aussi capturées.

WET utilise la capacité de « event handling »[10] globale du navigateur pour capturer les événements parce que les auteurs veulent éviter l'installation d'un nouveau logiciel commercial sur chaque machine client pour capturer des données, tout en cherchant une indépendance des plateformes.

Il existe plusieurs méthodes pour stocker des données relatives à des événements capturés (cookies, éléments cachés du formulaire, etc.) ; le choix des auteurs s'est porté sur les

[10] Chaque navigateur, IE ou Nescape, possède son propre modèle d'événements mais ils traitent, de manière similaire, un ensemble commun d'événements relatif aux interactions apparues sur une page Web (voir [Etgen et Cantor, 1999]. Lors de l'apparition d'un événement, le navigateur (IE ou Nescape) génère un objet d'événement contenant les informations relatives à ces événements. Les auteurs utilisent les functions globales de traitement d'événements (global event handling functions) au niveau fenêtre et document pour capturer les objets d'événéments générés. Les lecteurs intéressés par le traitement de ces événements globaux peuvent visiter http://javascript.about.com/library/bltut33.htm

cookies[11]. Précisons que cette méthode a un défaut lié à la taille limite de stockage d'une cookie (4K) ; 20 cookies étant un maximum pour un site. Donc, avec cette méthode, il est impossible de capturer tous les événements se produisant ; on doit choisir un nombre limité d'événements intéressants à capturer.

Le début et la fin de chaque session de test correspondent à la période entre les appuis sur les deux boutons *Start* et *Stop*. Quand le bouton *Stop* est cliqué, les données capturées sont récupérées pour être stockées en base de données et les cookies sont supprimées.

A partir des événements capturés, quelques mesures peuvent être effectuées comme le nombre de tâches accomplies, le temps pris pour réaliser les tâches, etc. A partir de ces mesures, l'évaluateur peut déterminer quelles tâches ont causé des problèmes à l'utilisateur et ainsi comparer les activités de différents utilisateurs.

WET est simplement une méthode de collecte de données liées aux événements d'interaction. L'analyse des fichiers « log » est encore manuelle. Ensuite, l'évaluateur doit interpréter les résultats d'analyse lui-même et proposer les améliorations possibles.

3.3.8. IBOT

[Zettlemoyer *et al*.,1999] ont proposé un agent logiciel utilisé dans l'environnement d'interfaces utilisateur de type WIMP. Le but final des auteurs est un agent idéal qui peut agir de façon autonome et rationnelle, par exemple, interagir avec n'importe quelle application interactive comme un utilisateur humain. L'architecture de ce système est illustrée sur la figure 2.17. Ce système est composé de trois modules : gestion des événements, traitement des images et représentation des états internes.

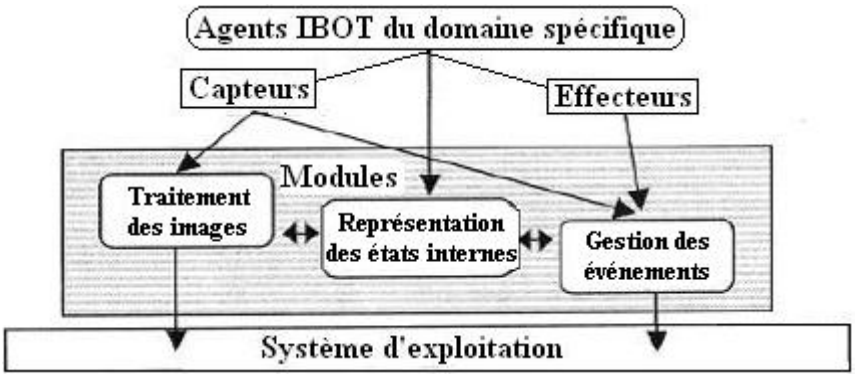

Figure 2.17 : Architecture de IBOT, traduite de [Zettlemoyer *et al*., 1999]

a) *Le module de traitement des images* est responsable de « voir » ce que l'utilisateur « a vu » sur l'écran au moment où il a interagi avec l'application interactive. Ce module interagit avec le système d'exploitation pour capturer l'information de tampon d'écran (« buffer »). Par le traitement automatique de l'image de l'écran, ce module capture l'information sur le contenu de l'écran pour identifier les « widgets » de l'interface utilisateur.

b) *Le module de gestion des événements* joue deux rôles :

- Le rôle de capteur : dans celui-ci, ce module accède directement à la queue d'événements du système d'exploitation pour capturer les événements de bas niveau comme les actions sur la souris ou le clavier suite aux interactions avec l'interface de l'application. Notons

[11] Un Cookie correspond à des données textuelles stockés par le navigateur Web sur la machine de l'utilisateur, utilisées par le serveur Web (préférence de l'utilisateur, identificateur de session, etc.). Le serveur Web envoie le cookie au client Web (souvent un navigateur). Le client Web le renvoie au serveur chaque fois qu'il accède à celui-ci.

que ce module considère seulement les événements de la souris et du clavier et il peut choisir les événements intéressants à capturer, l'évaluateur pouvant spécifier par avance ces événements.

- Le rôle d'effecteur : dans celui-ci, ce module insère les événements capturés dans la queue d'événements du système d'exploitation. En conséquence, ce module peut rejouer (playback) les actions déjà réalisées par l'utilisateur sur l'interface.

c) *Le module de représentation des états internes* : il peut combiner l'information sur le contenu de l'écran (capturée par *le module de traitement des images)* avec les événements clavier ou souris (capturés par *le module de gestion des événements)* pour déterminer les événements correspondant au niveau le plus haut sur l'interface, c'est-à-dire produits sur les widgets de l'interface.

Par exemple, l'utilisateur a cliqué sur le bouton gauche de la souris au moment T sur le point de coordonnées X,Y de l'écran. Cet événement a été capturé par *le module de gestion des événements*. L'information correspondante au contenu de l'écran à cet instant T a été aussi capturée par *le module de traitement des images*. Le *module de représentation des états internes* peut combiner cet événement de la souris avec cette information sur le contenu de l'écran à un instant T pour déterminer que l'utilisateur a cliqué sur le bouton OK de l'interface (événement *boutonOK_Click*).
Ces informations sont enregistrées et peuvent être récupérées n'importe quand.

d) L'outil IBOT permet à l'évaluateur de « voir » vraiment toutes les actions que l'utilisateur a effectué sur l'interface parce qu'il peut rejouer (playback) les actions déjà réalisées par celui-ci. Donc, l'évaluateur peut identifier les comportements inutiles ou incorrects de l'utilisateur, comparer les comportements de différents utilisateurs ; par exemple, il peut comparer ceux d'utilisateurs expérimenté et novices. L'évaluateur peut aussi identifier les parties de l'interface avec lesquelles les utilisateurs ont interagi rarement. A partir de ces informations, l'évaluateur peut suggérer des améliorations nécessaires.

3.3.9. WebQuilt

[Hong *et al*., 2001 ; Waterson, 2002] ont présenté l'outil WebQuilt permettant d'évaluer des interfaces Web. Les auteurs ont voulu remédier aux inconvénients des outils utilisant des « logs » du client comme WET ou WebRemUSINE (présentés ci-dessus) ou de ceux utilisant des « logs » du serveur. Les outils utilisant des « logs » du client peuvent nécessiter l'installation d'un logiciel sur la machine client pour capturer les données, alors que les outils utilisant des « logs » du serveur ont plusieurs inconvénients, le principal étant leur imposibilité de capturer les événements relatifs aux interactions locales avec les éléments de l'interface (boîte de texte, bouton, etc.) sur la machine client, comme mentionné dans [Etgen et Cantor, 1999 ; Hong *et al*., 2001 ; Paganelli et Paternò, 2003]. En conséquence, les auteurs proposent l'architecture de proxy pour capturer les données comme sur la figure 2.18a.

Avec cet outil, on ne doit pas installer de logiciels supplémentaires sur la machine client pour capturer les données. Cet outil est compatible avec les différents systèmes d'exploitation et navigateurs. Pour l'utiliser, il faut d'abord prédéfinir les différentes tâches à réaliser, et puis recruter des utilisateurs pour les réaliser en accédant à l'URL du serveur Web via le proxy qui va automatiquement collecter les données. Ensuite, l'évaluateur peut utiliser les fonctionnalités de WebQuilt pour analyser, visualiser et interagir avec les données collectées pour détecter les problèmes d'utilisabilité. La figure 2.18b illustre la structure des composants de WebQuilt.

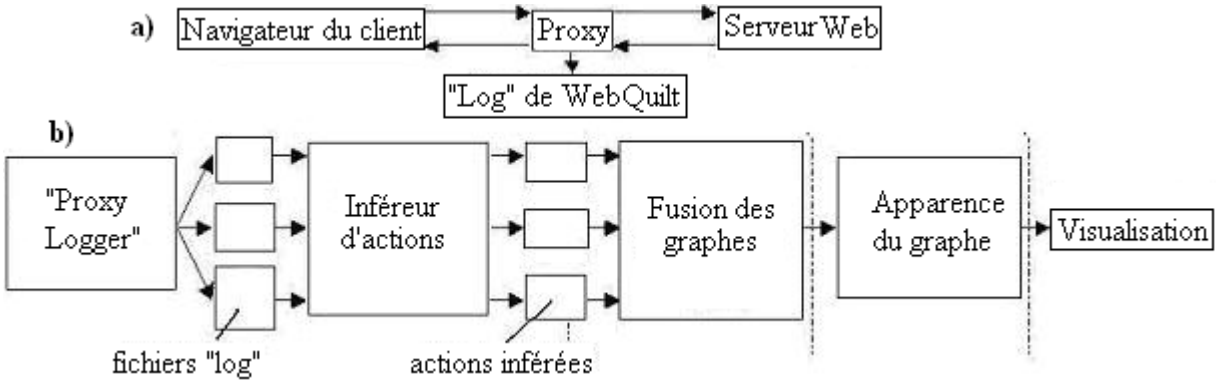

Figure 2.18 : a) Architecture pour la collecte des données dans WebQuilt. b) Structure des composants de WebQuilt, traduits de [Hong *et al*., 2001 ; Waterson, 2002]

La structure de l'outil est organisée avec des composants indépendants. Les communications minimales entre eux facilitent la modification d'un composant sans affecter les autres. Chaque composant est maintenant présenté.

a) « Proxy Logger » et « Inféreur des actions »

D'abord, le composant « proxy logger » traite la requête du client pour extraire et sauvegarder les données nécessaires en fichiers « log » pour des traitements différés (ces données seront présentées ci-dessous). Puis, il transfère la requête au serveur pour récupérer le document demandé. Si ce document est existant, alors le proxy télécharge celui-ci, ajoute quelques données au fichier « log » et effectue les traitements nécessaires sur le document téléchargé avant de le retourner au client. Enfin, le proxy stocke toutes les pages HTML demandées dans une mémoire « cache »[12] du proxy pour que l'évaluateur puisse les ouvrir localement et regarder leur contenu comme les utilisateurs. Un fichier de session est créé pour chaque session de test.

Les informations que le proxy sauvegarde dans le fichier « log » concerne les pages Web visitées par les utilisateurs. Les informations sauvegardées sont : les paires de pages sous forme (page de départ, page cible), les liens cliqués sur les pages, etc.

Le composant « Inféreur d'actions » exploite le fichier « log » de chaque session : à travers la séquence des pages visitées ainsi que des liens cliqués, ce composant peut déterminer les actions que l'utilisateur a effectuées. Ces actions peuvent être, par exemple, des requêtes à des pages, ou l'appui sur les boutons « précédent » ou « suivant » du navigateur Web.

Bien que cet outil de collecte de données puisse remédier à quelques inconvénients des outils utilisant des « logs » du client, il ne peut pas capturer les interactions locales avec les éléments de l'interface (boîte de texte, bouton, etc.) sur la machine client. Les outils utilisant des « logs » du client peuvent le faire.

b) « Fusion des graphes »

Ce composant fusionne plusieurs fichiers « log » contenant toutes les actions effectuées par les utilisateurs afin de générer un graphe interactif orienté dont les nœuds sont les images

[12] Elle correspond à une mémoire de stockage temporaire des données fréquemment accédées dans un but d'accélération de la vitesse d'accès. Quand les données sont stockées dans la mémoire « cache », elles sont accédées ultérieurement au lieu des données originales. Plusieurs types de « cache » existent : CPU cache, disk cache, etc. La mémoire cache Web est utilisée par les navigateurs et/ou les proxy des serveurs Web pour stocker les réponses précédentes du serveur au client (par exemple, les pages Web) ; elle permet de réduire la quantité des informations transmises à travers le réseau.

des pages visitées et dont les flèches représentent les transitions entre les pages à travers des actions des utilisateurs. La figure 2.19 illustre une capture d'écran de ce graphe.

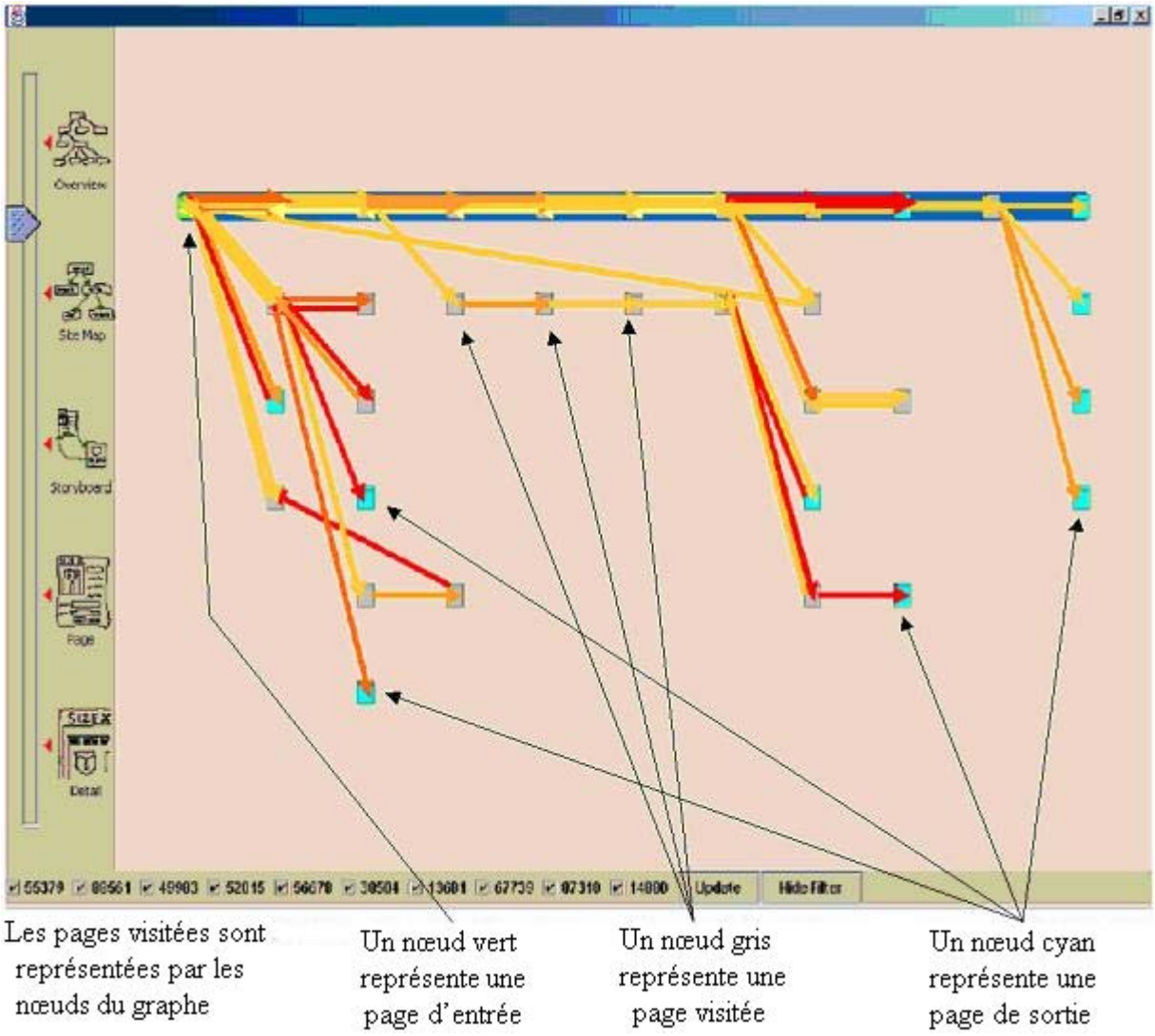

Figure 2.19 : Visualisation de WebQuilt. Les nœuds du graphe représentent les pages visitées, les flèches entre les nœuds représentent les transitions entre les pages [Hong *et al*., 2001 ; Waterson, 2002].

c) « Apparence du graphe » (Graph Layout) et « Visualisation »

Comme abordé ci-dessus, les données capturées dans les fichiers « log » sont visualisées sous forme d'un graphe interactif orienté. La figure 2.19 illustre une capture d'écran de ce graphe :

- Les pages visitées sont représentées par les nœuds du graphe. Les couleurs des nœuds sont différentes comme précisé sur la figure.
- Les flèches entre les nœuds du graphe représentent les transitions entre les pages. La couleur d'une flèche représente la durée de visite de la page avant la transition associée. Plus la flèche est rouge, plus la durée est longue. Plus la flèche est jaune, plus la durée est courte. L'épaisseur d'une flèche représente la fréquence d'apparition de la transition associée. Plus la flèche est épaisse, plus la fréquence de suivi de cette transition est élevée. Un chemin de navigation suivi pour compléter une tâche est composé d'une ou de

plusieurs transitions entre les pages (à partir de la page d'entrée jusqu'à la page de sortie). Le chemin optimal attendu par le concepteur est coloré en bleu.

- L'évaluateur peut utiliser les fonctionnalités *Zoom avant* et *Zoom arrière* pour afficher les pages et les chemins avec différents niveaux de détails.

d) Les caractéristiques et les problèmes découverts :

Avec cet outil, l'évaluateur peut détecter certains problèmes et caractéristiques comme :

- les pages que l'utilisateur a visitées le plus longtemps ; les patterns de navigation (les séquences des pages visitées qui apparaissent plusieurs fois durant la navigation) ; les pages d'entrée et de sortie, les chemins de navigation suivis par l'utilisateur, la fréquence des chemins suivis (fréquemment ou rarement), etc. L'évaluateur peut aussi détecter les problèmes concernant le contenu, le chemin, la structure etc. du site Web en utilisant ces informations.
- L'évaluateur peut déterminer la discordance entre un chemin de navigation suivi par l'utilisateur et un chemin considéré comme optimal par le concepteur. L'évaluateur peut aussi ouvrir localement et visualiser les contenus des pages (qui ont été stockées dans la mémoire « cache » du proxy) comme les utilisateurs.

L' évaluateur doit interpréter les résultats d'analyse et proposer les améliorations au concepteur du site Web.

3.3.10. MESIA proposé dans le cadre de travaux précédents au LAMIH

Ce mouchard, baptisé MESIA (Mouchard électronique dédié à l'Evaluation des Systèmes Interactifs orientés Agents) est spécifique au système interactif à base d'agents appelé SAI (Système d'Aide à l'Information) . Il a été développé au sein du laboratoire LAMIH et présenté dans [Ezzedine et al., 2003 ; Trabelsi, 2006]. Ce système SAI a été conçu pour superviser le transport public terrestre Valenciennois (Tramway, bus). Il permet aux régulateurs dans la salle de contrôle de superviser l'état courant de chaque ligne, des stations des lignes et des véhicules (position, heure, perturbations, etc.). Selon les situations, les régulateurs peuvent aussi envoyer des messages aux conducteurs des véhicules, aux voyageurs (à l'intérieur des véhicules ou en attente dans les stations). Le SAI sera présenté plus en détails dans le chapitre concernant notre expérimentation (cf. chapitre 4).

Le système interactif SAI est composé de six agents d'interface (*Etat_Trafic*, *Etat_Ligne*, *Station*, *Véhicule*, *Message*, *Vue_Globale*) qui sont associés aux agents mouchard du MESIA présenté maintenant. La vue d'ensemble de ce mouchard est introduite ci-dessous, les lecteurs intéressés peuvent consulter [Ezzedine *et al*., 2003 ; Trabelsi *et al*., 2004 ; Trabelsi, 2006] pour les détails. Il a été spécifiquement développé et utilisé pour aider à évaluer le SAI. Ce mouchard est composé de différents agents mouchards. Le principe de base du couplage entre le SAI et ce mouchard est d'associer chaque agent mouchard à chaque agent *interface* du SAI. C'est pourquoi ce mouchard doit disposer de cinq agents mouchards correspondant à cinq agents *interface* principaux du SAI comme l'illustre la figure 2.20 (notons que le mouchard électronique a trois autres agents mouchards en dehors de ceux cités : agent mouchard souris, agent mouchard clavier, agent mouchard superviseur).

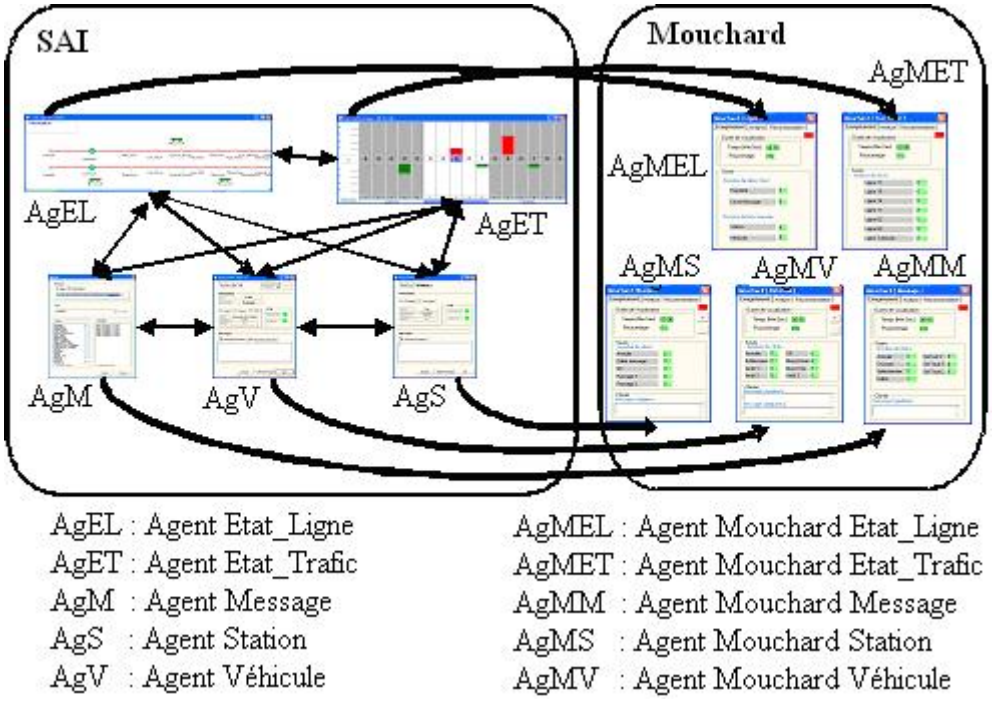

Figure 2.20 : Principe de base du couplage entre le mouchard MESIA et le SAI - Association de chaque agent mouchard à chaque agent interface du SAI [Trabelsi, 2006]

On peut voir l'architecture du MESIA sur la figure 2.21. Les agents constitutifs ont été développés pour capturer le nombre de clics de souris sur les éléments interactifs de l'agent interface correspondant du SAI :

- *Agent Mouchard Etat_Ligne* : il capture les événements qui se sont produits sur l'agent interface *Etat_Ligne* et ses éléments interactifs (*Station*, *Véhicule*, *Propriétés*, etc.).
- Agent Mouchard *Etat_Trafic* : il capture les événements qui se sont produits sur l'agent interface *Etat_Trafic*, lignes de bus et de tramway.
- Agent Mouchard *Station* : il capture les événements qui se sont produits sur l'agent interface *Station* et ses éléments constitutifs (boutons *OK*, *Annuler*, etc., onglets *Passage 1*, *2*, etc.).
- Agent Mouchard *Véhicule* : il capture les événements qui se sont produits sur l'agent interface *Véhicule* et ses éléments interactifs (boutons *OK*, *Annuler*, etc., onglets *Arrêt 1*, *2*, *3*, etc.).
- Agent Mouchard *Message* : il capture les événements qui se sont produits sur l'agent interface *Message* et ses éléments interactifs (boutons *Envoyer*, *Annuler*, etc., autres checkboxs, etc.).
- Agent Mouchard *Souris* et agent Mouchard *Clavier* : ils capturent les données concernant les événements provenant des actions de l'utilisateur sur la souris et le clavier (déplacement de la souris, clic sur les boutons droit et gauche de la souris, messages saisis au clavier, touches appuyées comme *Entrée*, *Echappement*, etc.).

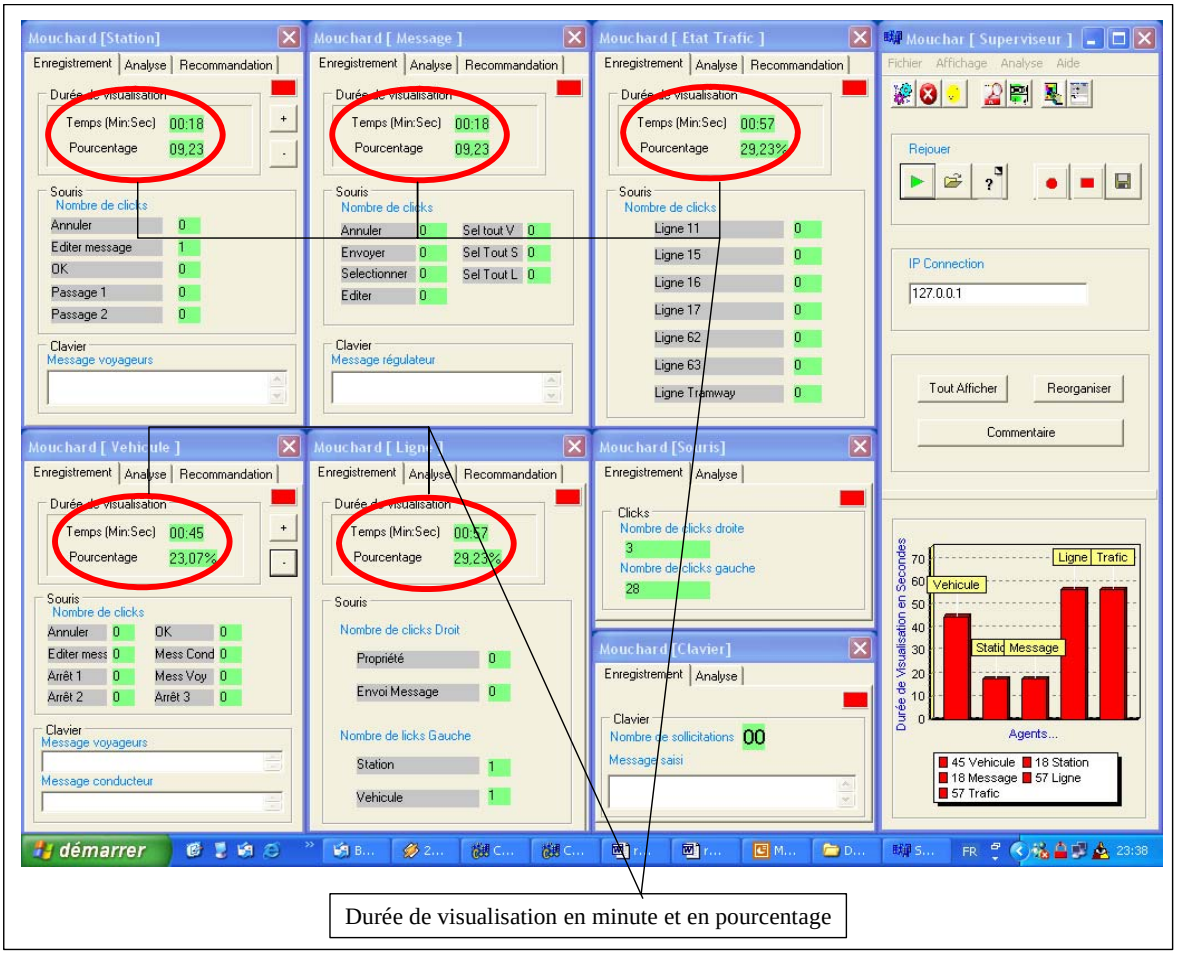

Figure 2.21 : Architecture de MESIA [Trabelsi, 2006]

3.3.11. Synthèse sur les mouchards électroniques étudiés

Nous avons présenté quelques mouchards représentatifs proposés récemment. Notons que quelques autres outils existent ; les lecteurs intéressés peuvent consulter à ce sujet :

- [Al-Qaimari et McRostie, 1999] relativement à KALDI (Keyboard/mouse Action Logger and Display Instrument) ;
- [Cugini et Scholtz, 1999 ; Scholtz et Laskowski 1998] pour prendre connaissance de l'ensemble des outils de NIST (National Institute of Standards and Technology) comme WebVIP (Web Variable Instrumenter Program), WebSAT (Web Static Analyzer Tool), WebCat (Web Category Analysis Tool), FLUD (Framework for Logging Usability Data), etc.
- [Hochheiser et Shneiderman, 2001] pour Starfield visualisation et [Chi *et al.*, 2001] pour DOME Tree visualisation.

Les deux tableaux 2.3 et 2.4 synthétisent et confrontent les outils présentés selon différentes dimensions[13] :

- **Données capturées** : on y précise les événements capturés par une méthode.

[13] Le mouchard spécifique MESIA n'est pas dans ces tableaux. Il sera critiqué dans le chapitre suivant.

- **Principes de capture, stockage et récupération des données à traiter** : on s'intéresse à la manière utilisée par une méthode pour capturer, stocker et récupérer les événement afin de les analyser.
- **Analyse des données capturées** : on précise les analyses qu'une méthode peut effectuer sur les données capturées
- **Affichage des résultats d'analyse** : on s'intéresse à la manière d'affichage des résultats d'analyse d'une méthode.
- **Problèmes détectés, proposition d'améliorations** : on s'intéresse aux problèmes capables d'être détectés par une méthode, à la démarche d'utilisation les résultats d'évaluation pour suggérer ses améliorations.
- **Quelques propriétés non fonctionnelles** : on précise les avantages, les défauts d'une méthode et/ou l'applicabilité, la performance d'une méthode, etc.

Tableau 2.3 : Synthèse sur les mouchards électroniques 1

Outils	Données capturées (EVs = Evénements)	Principes de capture, stockage et récupération des données à traiter	Analyse des données capturées	Affichage des résultats d'analyse	Problèmes détectés, proposition d'améliorations	Quelques propriétés non fonctionnelles
USINE [Lecerof et Paterno 1998]	Les actions physiques de l'utilisateur sur l'interface de type WIMP (EVs des dispositifs d'interaction, EVs se produisant sur les «widgets ») et les informations associées (coordonnés de la souris, temps, nom et contenu des «widget» affectés, etc.)	L'outil QC/Replay capture et enregistre les événements dans des fichiers textuels. L'évaluateur doit spécifier le modèle de tâches de l'interface jusqu'au niveau le plus bas en utilisant l'outil CTTE. Enfin, l'évaluateur doit associer les tâches aux événements capturés pour les traitements ultérieurs.	Réalisation des tâches (tâches accomplies ou échouées), types d'erreurs commises par l'utilisateur (erreurs de préconditions ou actions inutiles), calculs statistiques (nombre de tâches réalisées, temps pris pour chaque tâche, nombre d'erreurs commises, etc.)	L'analyse est représentée sous forme textuelle ou graphique.	L'évaluateur peut déterminer : si les objectifs de l'utilisateur sont atteints, si l'apparence courante de l'interface est efficace, si l'interface est lisible, si l'utilisateur réalise les tâches facilement, les tâches qui causent des problèmes à l'utilisateur, etc. L'évaluateur doit interpréter les résultats d'analyse lui-même pour détecter les problèmes et pour proposer des améliorations.	1) L'évaluateur doit comprendre le modèle de tâches CTT pour utiliser ces méthodes de la famille USINE. 2) L'outil de « logging » de WRU, Multimodal WebRemUsine fonctionne seulement sur les deux navigateurs (IE et Nescape Communicator). 3) Nous pensons que WRU et Multimodal WebRemUsine peuvent être utilement appliquées aux sites Web dont l'objectif vise à fournir des fonctionnalités métier spécifiques dans un domaine quelconque mais il n'est pas sûr qu'elles soient applicable aux sites orientés informations (cf. 3..3..4) Pour les méthodes USINE et RemUSINE, on doit installer des logiciels supplémentaires sur la machine client pour capturer les données. Pour WebRemUsine, on doit aussi installer JDK sur la machine client au minimum pour que Java Applet puisse fonctionner afin de capturer les données. L'installation supplémentaire sur la machine client cause des difficultés pour le déploiement. 5) Multimodal WebRemUsine est plus coûteux parce qu'on doit équiper les machines des clients de dispositifs (WebCam, oculomètre).
RemUSINE [Paterno *et al.*, 1999], [Paterno et Ballardin 2000]						
Web RemUSINE (WRU) [Paganelli et Paternò 2002], [Paganelli et Paternò 2003]	Trois types d'EVs concernant l'interface Web (EVs provenant des interactions de l'utilisateur sur le navigateur Web, EVs internes du navigateur Web, EVs concernant le changement de la tâche cible de l'utilisateur). Quelques informations supplémentaires sont aussi capturées (date, adresse IP et nom du système client, etc.)	Javascript et les Applets sont utilisés pour capturer les événements. En fin de session, ces informations sont enregistrées dans des fichiers « log » qui sont transmis au serveur. L'évaluateur doit spécifier le modèle de tâches de l'interface jusqu'au niveau le plus bas en utilisant l'outil CTTE. Enfin, l'évaluateur doit associer les tâches aux événements capturés pour les traitements ultérieurs.	Certaines analyses sont similaires à celles de USINE. D'autres analyses statistiques concernent les pages Web (nombre d'accès aux pages, patterns de visite des pages, temps pris pour visiter ou télécharger chaque page, etc.). L'analyse peut être réalisée pour une session individuelle ou un groupe de sessions.		On peut détecter les problèmes similaires à ceux détectés avec USINE et aussi d'autres problèmes concernant le contenu, la taille, etc., des pages Web. L'analyse est réalisée pour une session ou un groupe des sessions pour déterminer si un problème est souvent rencontré par plusieurs utilisateurs ou est limité à des utilisateurs spécifiques. L'évaluateur doit interpréter lui-même les résultats d'analyse pour détecter les problèmes et pour proposer des améliorations.	
Multimodal Web RemUSINE [Paterno *et al.*, 2006]	Trois types d'EVs similaires à ceux de WRU et des données concernant la direction du regard de l'utilisateur et de son contexte de travail	L'outil de « logging » (comme pour WRU), une Webcam, et un oculomètre sont installés sur la machine client. L'évaluateur spécifie le modèle de tâches et fait les associations comme la méthode WRU	Les analyses sont similaires à celles de WRU, tout en permettant la combinaison entre différents types de données (vidéo, oculomètre).	L'analyse est représentée sous forme textuelle ou graphique, et sous forme de séquence de vidéo.	La combinaison entre plusieurs types de données permet de mieux comprendre (par rapport à WRU) l'activité et le contexte de travail de l'utilisateur. L'évaluateur doit interpréter les résultats d'analyse lui-même.	
Multidevice RemUSINE [Paterno *et al.*, 2007]	Trois types d'EVs similaires à ceux de WRU sur les applications mobiles et autres informations contextuelles (localisation, puissance du signal réseau, etc.)	L'outil « Mobile Logger » capture et stocke les données dans des fichiers XML. L'évaluateur spécifie le modèle de tâches et fait les associations comme la méthode WRU	Les analyses sont similaires à celles des autres méthodes, tout en les combinant avec des informations contextuelles.	L'analyse est représentée sous forme textuelle ou graphique.	La combinaison entre des EVs d'interaction capturés et des informations contextuelles par rapport auxquelles l'utilisateur évolue permet de prendre connaissance des conditions affectant les activités de l'utilisateur. L'évaluateur doit interpréter les résultats d'analyse lui-même.	

Tableau 2.4 : Synthèse sur les mouchards électroniques 2

Outils	Données capturées (Evs=Evénements)	Principes de capture, stockage et récupération des données à traiter	Analyse des données capturées	Affichage des résultats d'analyse	Problèmes détectés, proposition d'améliorations	Quelques propriétés non fonctionnelles
WET [Etgen et Cantor, 1999]	Les EVs d'interaction entre l'utilisateur et l'interface Web (clic souris, bouton, etc.), les EVs de navigateur (chargement d'une page, etc.) et les informations associées (coordonnées sur l'écran des EVs souris, temps d'apparition, etc.).	Wet utilise la capacité de « event handling » globale du navigateur pour capturer les EVs. Les cookies sont utilisés pour stocker ces données. A la fin de la session, les données capturées sont envoyées à la base de données et les cookies sont supprimées.	Quelques mesures peuvent être effectuées (nombre de tâches accomplies, temps pris pour accomplir les tâches, etc.). WET est simplement une méthode de collecte des EVs. L'analyse des fichiers « logs » est manuelle	Pas d'affichage. WET est seulement une méthode de collecte des EVs.	L'évaluateur peut déterminer quelles tâches ont causé des problèmes à l'utilisateur et il peut aussi comparer les performances de différents utilisateurs. L'évaluateur doit interpréter les résultats d'analyse et proposer des améliorations lui-même.	1) WET est seulement applicable sur l'interface Web des 2 navigateurs IE et Nescape. 2) on ne doit pas installer un logiciel supplémentaire sur l'ordinateur client pour capturer les données, donc le déploiement est plus facile. 3) L'utilisation des cookies pour stocker les EVs capturés est la cause de la limitation de la quantité des informations. Donc, il est impossible de capturer tous les EVs se produisant, on doit choisir quelques EVs s intéressants à capturer.
IBOT [Zettlemoyer *et al.*, 1999]	Deux types EVs : 1) l'information sur le contenu de l'écran lorsque l'utilisateur a interagi avec l'application. 2) les EVs de bas niveau (souris et clavier).	L'agent IBOT accède directement à la queue d'événements du système d'exploitation pour capturer les événements de bas niveau provenant de la souris ou du clavier et capture l'image de l'écran dans la mémoire tampon .	IBOT combine l'information sur l'écran avec les EVs de clavier ou souris pour déterminer l'EV correspondant au niveau le plus haut. IBOT insère les EVs dans la queue d'EVs du système d'exploitation pour rejouer les actions faites par l'utilisateur sur l'interface.	Pas d'affichage.	L'évaluateur peut étudier ce que l'utilisateur a effectué pour identifier les actions inutiles ou incorrectes, comparer les performances des utilisateurs, identifier les parties de l'interface ayant des problèmes. L'évaluateur doit interpréter les résultats d'analyse et proposer des améliorations lui-même.	IBOT peut être utilisé pour capturer localement les données des interactions de l'interface WIMP ; la capture à distance n'est pas possible. IBOT considère seulement les EVs de la souris et du clavier.
WebQuilt [Hong *et al.*, 2001; [Waterson ,2002]	Les pages Web visitées sous forme (page de départ, page cible), les liens cliqués sur les pages. Cet outil ne peut pas capturer les interactions locales avec les éléments de l'interface (bouton, etc.) sur la machine client. L'inférence d'actions est effectuée pour détecter l'action de l'utilisateur (appuis sur les boutons « précédente » ou « suivante » du navigateur Web, etc.)	Les données sont capturées par le proxy. Elles sont sauvegardées dans des fichiers « log » (un fichier pour chaque session de test). Les actions des utilisateurs peuvent être inférées dans ces fichiers « log ».	L'outil peut détecter les pages visitées, la fréquence des chemins suivis, les patterns de navigation, la discordance entre le chemin réellement suivi et le chemin optimal attendu par le concepteur, etc. L'évaluateur peut ouvrir et regarder localement les contenus des pages (stockées dans la mémoire « cache » du proxy).	un graphe interactif orienté dont les nœuds représentent les pages visitées et les flèches entre les noeuds représent les transitions entre les pages.	L'évaluateur peut détecter les problèmes concernant le contenu, le chemin, la structure, etc., du site Web. L'évaluateur doit interpréter les résultats d'analyse et proposer des améliorations lui-même.	On ne doit pas installer de logiciels supplémentaires sur la machine client pour capturer les données. La structure de l'outil est organisée autour de composants indépendants avec un minimum de communications entre eux. Donc, la modification d'un composant n'affecte pas les autres.

4. Conclusion

Dans ce chapitre, après avoir introduit les principes de base de l'évaluation des systèmes interactifs et quelques classifications de méthodes d'évaluation, nous avons présenté un état de l'art des outils d'aide à l'évaluation.

Parmi les différents types d'outils, nous nous sommes intéressés essentiellement aux deux types suivants : les outils utilisant les règles ergonomiques (ergonomic guidelines) ; les mouchards électroniques capturant les données durant l'interaction entre l'utilisateur et le système interactif, avec une visée d'aide à l'évaluation. Nous avons examiné, sans souci d'exhaustivité, quelques méthodologies et outils concernés de ces deux types. Ces méthodologies et outils ont été proposés dans la littérature depuis une dizaine d'années comme WebTango, DESTINE, Sherlock, WebRemUSINE, MultiDevice RemUSINE, IBOT, WET, WebQuilt, MESIA, etc. Enfin, une synthèse sur ces méthodologies et outils a été proposée en se basant sur différentes dimensions.

Dans la lignée des outils d'aide examinés dans ce chapitre, nous avons proposé et développé, dans le cadre de cette thèse, un environnement générique et configurable pour l'aide à l'évaluation des systèmes interactif à architecture à base d'agents. Cet environnement est décrit dans le chapitre suivant.

Chapitre 3 : Proposition de l'environnement générique et configurable EISEval pour l'aide à l'évaluation des systèmes interactifs à base d'agents

Sommaire

1. Introduction

Dans ce chapitre, nous présentons l'environnement générique et configurable dit EISEval (Environment for Interactive System Evaluation) qui est proposé et développé dans le cadre de cette thèse en vue d'aider à l'évaluation des systèmes interactifs à base d'agents. Cet environnement est structuré d'une manière modulaire pour que le développeur puisse modifier un module sans affecter les autres.

Avant d'aborder notre proposition, il est important de critiquer le mouchard MESIA (cf. chapitre 2) et les autres mouchards traditionnels. Ce travail découle des limitations de ces mouchards. Ensuite, les principes de l'environnement d'évaluation EISEval, les niveaux abstraits des événements considérés dans celui-ci et la description formelle de l'architecture à base d'agents proposée seront présentés. Enfin, chaque module du EISEval sera décrit et illustré d'exemples.

2. Etude critique du mouchard MESIA à l'origine de cette thèse

Le mouchard MESIA (Mouchard électronique dédié à l'Evaluation des Systèmes Interactifs orientés Agents (présenté au chapitre précédent et proposé dans le cadre d'une thèse [Trabelsi, 2006]) présente quelques défauts et inconvénients qui sont les suivants :

- Premièrement, il ne constitue pas un modèle générique et configurable : il est **spécifique** et vise à aider à évaluer un système interactif à base d'agents appelé SAI (Système d'Aide à l'Information). On ne peut pas l'utiliser pour évaluer d'autres systèmes interactifs à base d'agents parce que son contenu dépend du système à évaluer. En effet, le contenu des agents mouchards (de MESIA) est lié au contenu des agents *interface* correspondants (du SAI) à évaluer comme l'illustre la figure 3.1 :

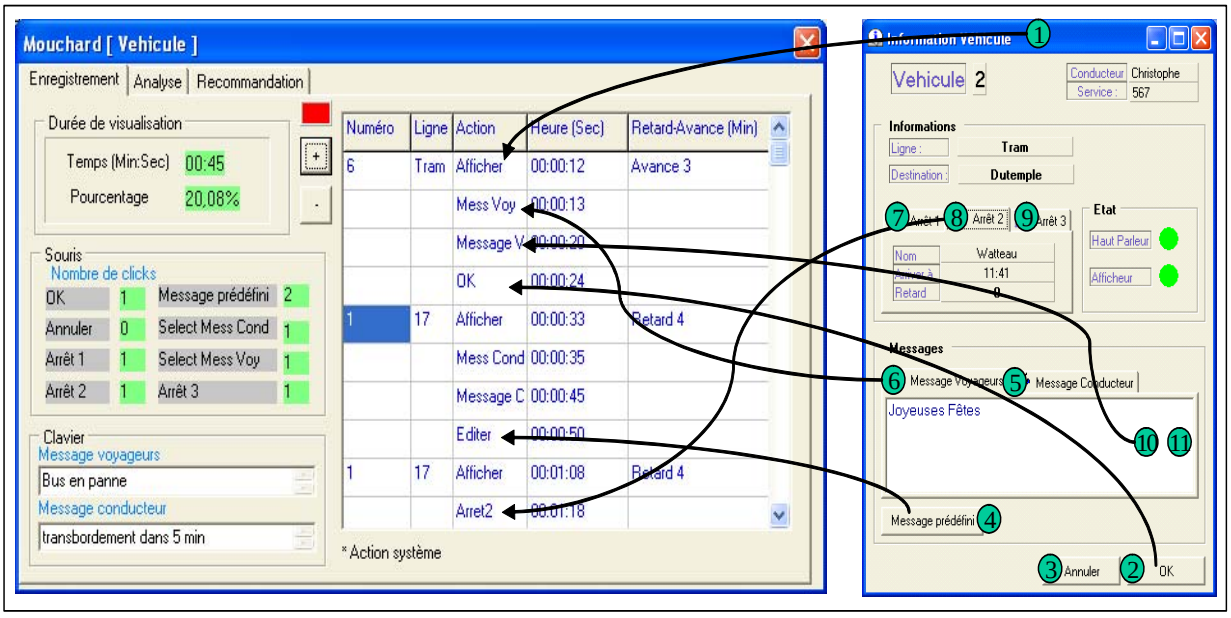

Figure 3.1 : Correspondance entre Agent *interface Véhicule* du SAI et Agent Mouchard *Véhicule* [Trabelsi, 2006]

Sur la figure 3.1, on voit clairement que le contenu de l'agent mouchard *Véhicule* (de MESIA) est fortement lié au contenu de l'agent *interface Véhicule* correspondant (du SAI) à évaluer. Le contenu des autres agents mouchards sont aussi fortement liés au contenu

des agents *interface* correspondant du SAI de la même façon. En conséquence, ces agents mouchards ne peuvent pas être utilisés pour évaluer un autre système interactif à base d'agents, étant spécifiques au SAI.

- Il présente un autre inconvénient. Le contenu des agents mouchards se base essentiellement sur les éléments interactifs prédéfinis des agents *interface* correspondants du SAI. Dans le cas où le contenu des agents *interface* du SAI est modifié, suite à l'insertion de nouveaux éléments interactifs dans un agent *interface*, ou bien suite à la suppression de quelques éléments interactifs existants d'un agent *interface* par exemple, le contenu des agents mouchards correspondants devra être aussi modifié. Ceci nécessitera l'intervention du développeur pour mettre à jour le code source de ces agents mouchards.
- Son principe de base de couplage avec le SAI est d'associer chaque agent mouchard à chaque agent *interface* du SAI. Ce couplage présente un inconvénient. Un outil générique ne peut pas appliquer ce principe de couplage parce que le nombre d'agents mouchard va dépendre du nombre d'agents du système interactif à évaluer ; la structure du mouchard va dépendre aussi de la structure des systèmes interactifs à évaluer. En effet, d'après ce principe, si le système interactif à évaluer est un système complexe dont le nombre d'agents *interface* est élevé (un système multi-agents contenant plus de 10 agents est considéré comme un grand système [Wooldridge et Jennings, 1999]), alors le mouchard devra avoir le même nombre d'agents mouchards correspondants. En conséquence, le mouchard peut devenir complexe et lourd à cause de cette dépendance du nombre d'agents. D'autre part, si la structure du système interactif à évaluer doit changer (insertion, suppression d'agents *interface*), alors la structure du mouchard doit être changée ; c'est-à-dire que le développeur doit changer le code source du mouchard.
- Il ne prend pas en compte les niveaux abstraits des événements capturés. Il s'intéresse seulement aux événements de bas niveau. Ces événements peuvent être très nombreux et peuvent surcharger l'évaluateur. En effet, les événements des dispositifs d'interaction ou des éléments interactifs graphiques peuvent être très diversifiés selon les intentions et il n'est pas alors facile pour l'évaluateur de comprendre, d'analyser les intentions, les motivations, les processus d'activité de l'utilisateur ainsi que ceux du système interactif à évaluer à travers des données comme le nombre de clics droit ou gauche de la souris, le nombre de clics sur les boutons *OK* ou *Annuler* de l'*interface*, etc. L'évaluateur peut être surchargé de données. En outre, ces données peuvent être capturées sur des périodes longues (plusieurs heures, 1 jour, 2 jours ou une semaine, par exemple). Les propriétés non-fonctionnelles du système interactif comme les temps de réponse, la fiabilité, etc., ne sont pas faciles non plus à évaluer avec de tels types de données.
- Il permet la capture et l'analyse essentiellement des interactions entre les utilisateurs et les agents *interface* du système (les actions des utilisateurs : appui sur une touche, mouvement de la souris, sélection d'un objet, etc. ; de même que les réactions du système : apparition d'un message d'avertissement, changement de vue, apparition d'une nouvelle fenêtre, changement du contenu, fermeture d'une fenêtre, etc.). Pour contribuer à l'évaluation complète du système, le mouchard devrait capturer puis analyser toutes les interactions entre les utilisateurs et les agents de l'*interface* et entre les agents eux-mêmes. Notons qu'une telle perpective a été envisagée dans [Trabelsi, 2006].
- Ses fonctionnalités sont limitées ; il mesure le temps d'affichage de chaque agent *interface* spécifique du SAI et le nombre des événements de bas niveau concernant le clavier, la souris ou concernant les éléments interactifs spécifiques (qui sont déjà prédéfinis) des agents *interface* correspondants du SAI. Enfin, il affiche les résultats sur une fenêtre. Il

s'agit maintenant de fournir de nouvelles fonctionnalités pour aider l'évaluateur dans ses activités.

- Le mouchard MESIA et le système interactif à évaluer doivent fonctionner en même temps afin de capturer en temps réel les données des interactions. En conséquence, en cas de problème quelconque (par exemple, si la machine tombe en panne ou dans le cas d'apparition d'erreurs du système d'exploitation, etc.), le mouchard ne peut pas fonctionner en temps réel et les données des interactions ne peuvent pas être capturées.

Dans la section suivante, avant de présenter l'environnement d'évaluation EISEval, nous discutons de la motivation de la proposition et du développement de cet environnement.

3. Proposition d'un environement générique et configurable pour l'aide à l'évaluation des systèmes interactifs à base d'agents

Dans cette section, les limitations des mouchards traditionnels sont d'abord présentées. Ensuite, nous allons décrire successivement les principes que l'environnement EISEval doit respecter, les niveaux abstraits des événements considérés dans celui-ci et la description formelle de l'architecture à base d'agents des systèmes interactifs. Enfin, la structure modulaire et chaque module de l'EISEval seront présentés et illustrés d'exemples.

3.1. Motivation

Nous avons proposé et développé un environnement d'évaluation étendant les possibilités offertes par les mouchards traditionnels, de même que par MESIA. En effet, la motivation de la proposition de l'environnement d'évaluation EISEval prend sa source des limitations du mouchard MESIA (présentées ci-dessus) mais aussi de celle des mouchards traditionnels :

- Plusieurs mouchards ont été proposés mais ils ne prennent pas en compte les spécificités liées aux architectures à base d'agent des systèmes interactifs pour l'évaluation. MESIA en a tenu compte mais il présente les défauts et inconvénients présentés précédemment. L'environnement d'évaluation proposé doit prendre en compte de telles spécificités tout en remédiant aux défauts et inconvénients de MESIA.

- La majorité de mouchards traditionnels sont focalisées sur l'interface utilisateur du système interactif. Ils ne facilitent pas beaucoup l'évaluation des propriétés non-fonctionnelles du système comme le temps de réponse, la fiabilité, etc. Pourtant ces propriétés sont très importantes pour évaluer le bon fonctionnement d'un système. Il s'agit donc d'aider l'évaluateur à évaluer plusieurs aspects du système interactif : l'interface, les propriétés non fonctionnelles du système et aussi certaines propriétés de l'utilisateur (facilité à utiliser le système, habitude des manipulations, progrès dans l'utilisation du système, comparaison entre utilisateurs, etc.).

- Les mouchards traditionnels capturent souvent les événements provenant de l'interface utilisateur ou les événements des dispositifs d'interaction et permettant des analyses de bas niveau sur ces données, par exemple : recherche d'une séquence d'interaction quelconque, calculs statistiques, visualisation de ces résultats. Les mouchards traditionnels se limitent souvent à cette étape. L'évaluateur doit lui-même interpréter ces résultats d'analyse pour en tirer des conclusions utiles et proposer au concepteur des améliorations nécessaires. Ces conclusions, lorsqu'elles sont tirées par des évaluateurs différents, peuvent être différentes et dépendre de leur expérience. Il n'y a aucune aide à proprement parler pour l'évaluateur. Nous avons donc l'intention d'aller plus loin que ces mouchards traditionnels en proposant et développant un nouvel environnement visant à associer les

résultats d'analyse à une liste ouverte de critères d'évaluation afin d'aider l'évaluateur à interpréter les résultats d'analyse issus des données capturées.

En conclusion, l'environnement d'évaluation EISEval proposé doit respecter un ensemble de principes pour remédier à ces défauts et inconvénients ; ces principes sont décrits ci-dessous.

3.2. Principes de l'environnement d'évaluation EISEval proposé

Six principes sont particulièrement visés :

- L'environnement proposé doit ***prendre en compte, pour l'évaluation, les spécificités de l'architecture à base d'agent des systèmes interactifs***. L'exploitation de ces spécificités implique en principe plusieurs avantages. En effet, l'évaluateur doit mieux comprendre les problèmes du système interactif à évaluer ; il doit pouvoir répondre plus rapidement à des questions comme : quels sont les services (des agents) qui ont à plusieurs reprises mal fonctionné (ils ont échoué, ou ont pris beaucoup de temps, etc.) ? Quelles sont les interactions (entre des services interagissant) qui ont pris beaucoup de temps ? Quels sont les agents *interface* qui ont souvent ou rarement interagi avec l'utilisateur ? Quels sont les agents *application* qui ont souvent ou rarement été mis à contribution ? Quels sont les agents qui ont connu des problèmes (concernant leurs services ou interfaces) ? etc. À partir des réponses à de telles questions, il devient plus facile à l'évaluateur de déterminer les améliorations nécessaires à apporter, puis de les proposer au concepteur.

  On doit pouvoir aussi utiliser cet environnement pour évaluer un système interactif qui n'est pas à base d'agents. En effet, on peut considérer que tel système interactif à évaluer est composé d'un seul agent *interface* qui constitue le système complet ; l'environnement doit pouvoir capturer les événements pour les analyser mais dans ce cas, il ne peut pas exploiter les spécificités de l'architecture à base d'agents du système interactif à évaluer bien qu'il possède des fonctionnalités originales par rapport aux mouchards traditionnels.

- L'environnement proposé doit ***capturer toutes les interactions du système*** : entre l'utilisateur et les agents *interface* (sous forme d'EVIUs, **év**énements **i**nterfaces **u**tilisateurs ; et d'EVDIs, **év**énements des **d**ispositifs d'**i**nteraction) et entre les agents eux-mêmes (sous forme des services), dans un but d'analyse. Précisons que les mouchards traditionnels ne capturent souvent que les interactions entre l'utilisateur et la partie interactive du système.

- L'environnement proposé doit aller plus loin que les mouchards traditionnels. Il doit ***aider l'évaluateur à interpréter les résultats d'analyse des données capturées.*** Comme précisé plus haut, cette interprétation doit permettre d'***évaluer des aspects différents*** d'un système interactif à base d'agents :
  - l'évaluation de l'interface utilisateur du système ;
  - l'évaluation de quelques propriétés non fonctionnelles du système : temps de réponse, fiabilité, etc.,
  - l'évaluation de certaine propriétés de l'utilisateur (capacité à utiliser le système, habitudes, préférence, etc.) et la comparaison des performances de différents utilisateurs.

- L'environnement proposé proposé doit être ***générique***, c'est-à-dire qu'il doit être indépendant des systèmes interactifs. Pour atteindre cet objectif, il est nécessaire de disposer d'une description formelle de l'architecture à base d'agents des systèmes interactifs. L'ancienne description (présentée dans [Trabelsi, 2006]) est inexploitable

directement vis-à-vis de cet objectif ; en conséquence, notre première tâche dans cette thèse est de proposer une autre description formelle de l'architecture à base d'agents des systèmes interactifs qui permettra d'atteindre cet objectif. Cette description formelle sera présentée plus loin dans ce chapitre.

- L'environnement proposé doit être ***configurable***, c'est-à-dire qu'il doit pouvoir être configuré pour aider à évaluer différents systèmes interactifs.
- L'environnement proposé doit ***distinguer les niveaux abstraits*** des événements. Nous allons présenter ceux-ci dans la section suivante.

3.3. Niveaux abstraits des événements

L'ancien mouchard MESIA ne prend pas en compte des niveaux abstraits des événements capturés. Il s'intéresse seulement aux événements de bas niveau. Ces événements capturés peuvent être en quantité important et surcharger l'évaluateur. En effet, les événements issus des dispositifs d'interaction ou des éléments interactifs peuvent être très nombreux et difficiles à interpréter. Une répartition en plusieurs niveaux abstraits des événements du système interactif peut faciliter le travail de l'évaluateur. A ce sujet, [Hilbert et Redmiles, 2000] ont présenté un modèle ayant six niveaux abstraits (ce qui nous semble beaucoup). Dans notre cas, l'environnement d'évaluation proposé travaille avec trois niveaux qui sont successivement décrits.

3.3.1. Evénements considérés

1) Les *EVDIs* (événements de dispositif d'interactions) sont des événements placés au niveau le plus bas, celui où on retrouve ceux générés par les dispositifs d'interaction (clic du bouton droit ou gauche de la souris, appui sur une touche quelconque du clavier, etc.).

2) Le niveau intermédiaire contient deux types d'événements :

2.1) Les *EVIU*s (événements interfaces utilisateurs) : il s'agit des événements concernant les objets d'interface sur l'écran de l'application (ouverture ou fermeture d'une fenêtre, clic sur un bouton, sélection d'un item dans un menu, et*c.)*. De tels événements peuvent se produire seulement sur les agents *interface* de l'architecture proposée ; ils ne peuvent pas se produire sur les deux autres types d'agents : *contrôleur* et *application*. L'activité d'un EVIU se déroule de la façon suivante :

- Un EVIU se produit sur son agent *interface* associé s'il est déclenché par un des trois objets suivants (dit événement déclencheur) :
    - Un *utilisateur*. Un EVIU peut être déclenché par l'utilisateur à travers un dispositif d'interaction quelconque, par exemple : un item dans un menu est sélectionné ou un bouton sur l'écran est cliqué lorsque l'utilisateur clique sur le bouton gauche de la souris, etc.
    - Un *service*. Un EVIU peut être déclenché par un service (décrit ci-dessous) de son agent *interface* associé, c'est-à-dire que cet EVIU est considéré comme une *action interne* (*aci*) de ce service. Un service peut déclencher une ou plusieurs *actions internes*. Par exemple, dans le cas d'un système interactif de supervision du transport public terrestre, lorsqu'une perturbation se passe sur le trafic, le service d'avertissement de perturbation d'un agent de ce système va déclencher une *action interne* concernant l'affichage d'une fenêtre d'avertissement destinée à avertir le régulateur du trafic et lui demander de traiter cette perturbation.

  - Un *autre EVIU*. Un EVIU peut être déclenché par un autre EVIU de son agent *interface* associé ; par exemple, une fermeture d'une fenêtre peut être déclenchée par un clic sur le bouton *Annuler* de cette fenêtre.

- Un EVIU peut aussi jouer le rôle d'un événement déclencheur. En effet, il peut déclencher un ou plusieurs services de son agent *interface* associé ; par exemple, un clic sur un bouton de la fenêtre peut déclencher un service qui exécute une fonction métier du système interactif. De plus, un EVIU peut aussi déclencher un ou plusieurs EVIUs de son agent *interface* associée ; par exemple, un clic sur un bouton *Annuler* d'une fenêtre peut déclencher une fermeture de cette fenêtre comme présenté ci-dessus

2.2) Les *services*. En général, dans notre cas, un service[14] d'un agent est une action que celui-ci peut exécuter. Les services peuvent s'exécuter par les trois types d'agents : agents *interface*, *application* et *contrôleur*. Le dialogue entre les agents est réalisé par les invocations entre les services des agents. L'activité d'un service se déroule de la façon suivante :

- Similaire avec un EVIU, un service peut être déclenché par un des trois objets suivants (dit événement déclencheur) :
  - Un *autre service*. Un service peut être déclenché par un autre service du même agent ou d'un autre agent ; par exemple, dans un système interactif de supervision du transport public terrestre, un service de détection de perturbation d'un agent peut déclencher un service d'avertissement de perturbation qui va déclencher l'affichage d'une fenêtre d'avertissement destinée à avertir le régulateur du trafic et lui demander de traiter celle-ci. Le dialogue entre les agents est réalisé sous forme de telles interactions entre les services.
  - Le *système*. Un service d'un agent *interface* ou d'un agent *application* peut être déclenché par le système (c'est-à-dire que ce service est automatiquement exécuté). Les services des agents *contrôleur* ne peuvent pas avoir cette capacité. Par exemple, un service de détection de perturbation d'un agent du système interactif de supervision peut être automatiquement exécuté lorsqu'une perturbation se produit dans l'application supervisée.
  - Un *EVIU*. Un service d'un agent *interface* peut être déclenché par un EVIU sur le même agent *interface*, par exemple, un clic sur un bouton de la fenêtre peut déclencher un service qui exécute une fonction métier quelconque du système interactif comme présenté ci-dessus.

- Un service peut aussi jouer le rôle d'un événement déclencheur. En effet, un service peut déclencher un ou plusieurs services du même agent ou d'un autre agent ; le service déclenché est dit une *action externe* (*ace*) du service déclencheur. De plus, un service d'un agent *interface* peut aussi déclencher un ou plusieurs EVIUs de cet agent *interface*. L'EVIU déclenché est dit une *action interne* (*aci*) du service déclencheur comme présenté ci-dessus.

Nous pouvons aussi utiliser les Réseaux de Pétri (RdP) pour décrire l'activité d'un service d'un agent comme l'illustre la figure 3.2. Sur cette figure, la place *Etat actuel* contient toujours la vue actuelle de l'agent (résultat d'un service). Elle définit l'état actuel de l'agent. Un service peut avoir besoin des ressources éventuelles nécessaires ou des conditions à vérifier pour exécuter. Pour qu'un service soit déclenché, trois conditions doivent être vérifiées : l'apparition de l'événement déclencheur, la validation de la condition du

[14] Un service est une action effectuée par une entité (personne physique ou morale, entreprise, machine, programme) pour le bien d'une autre, avec ou sans contrepartie » (selon le dictionnaire « Petit Larousse).

déclenchement du service, la disponibilité de certaines ressources éventuellement requises pour le déclenchement du service. Si un événement déclencheur apparaît à sa place, et que les conditions sont vérifiées et les resources nécessaires disponibles, alors la transition est validée et le service est déclenché. Ce service exécute les *actions internes* (déclenchement des EVIUs) et/ou les *actions externes* (déclenchement des autres services du même agent ou d'un autre agent).

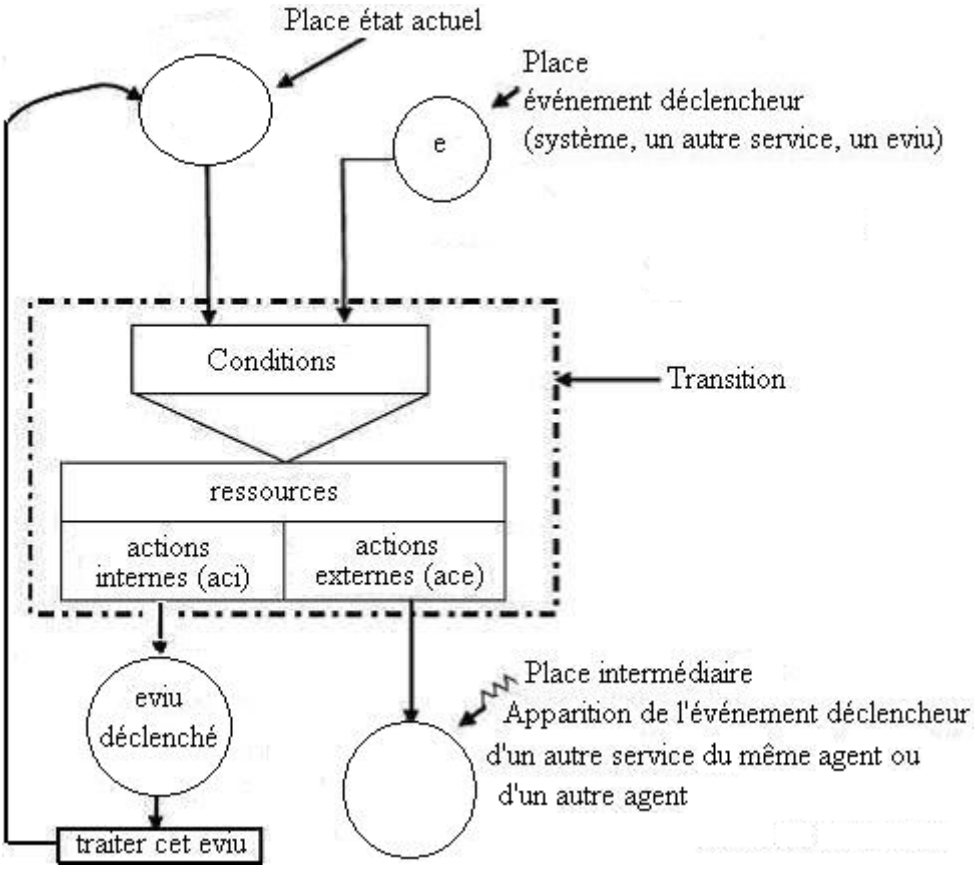

Figure 3.2 : RdP de modélisation de l'activité d'un service d'un agent. Ce RdP dérive de celui décrit dans [Ezzedine *et al*., 2003 ; 2006]

3) Le premier niveau est le plus élevé. Les événements à ce niveau correspondent à des *tâches unitaires* (ou *tâches* pour plus de concision) ; ce sont les tâches que l'utilisateur et/ou les agents du système doivent réaliser.

Une tâche peut être déclenchée par l'*utilisateur* ou par le *système* :

- Si une tâche est déclenchée par le système (*tâche système*), alors elle doit être initialisée par un événement de niveau intermédiaire appelé initiateur qui peut être un service d'un agent *application* ou d'un agent *interface*. Ce service est déclenché par le système. Ensuite, cet initiateur peut déclencher un ou plusieurs services ou EVIUs, c'est-à-dire qu'il n'y a pas d'interventions directes de l'utilisateur dans la réalisation des tâches systèmes. Par exemple, une tâche d'avertissement de perturbation pour l'utilisateur est initialisée par un service de détection de perturbation. Ce service est automatiquement déclenché par le système ; puis il déclenche un autre service d'avertissement de perturbation qui va déclencher un EVIU d'affichage d'une fenêtre d'avertissement destinée à avertir l'utilisateur de la perturbation détectée. Comme nous pouvons le constater, cette tâche concerne deux services et un EVIU qui sont au niveau abstrait

intermédiaire. Dans les systèmes interactifs de supervision, les tâches systèmes sont souvent réalisées lorsque : (1) le système informe l'opérateur de l'état courant de l'application pour qu'il puisse réaliser les fonctions de supervision, (2) le système avertit l'opérateur d'une perturbation ou d'un problème de l'application pour qu'il puisse réaliser les fonctions de commande ou de régulation.

- Si une tâche unitaire est déclenchée par l'utilisateur (*tâche utilisateur*), alors elle doit être initialisée par un ou plusieurs EVIUs séquentiels, déclenchés par l'utilisateur. Ces EVIUs peuvent déclencher d'autres EVIUs ou services pour réaliser cette tâche. La tâche « *Envoyer un message aux voyageurs en station* » que nous allons présenter dans la section suivante est un exemple de tâche utilisateur. Elle est séquentiellement déclenchée par l'utilisateur à travers trois EVIUs : *ImageStation_Click*, *BoîteTexteMessage_Changée*, *boutonOK_Click*. Dans les systèmes interactifs de supervision, les tâches utilisateur correspondent souvent aux activités de l'opérateur pour réaliser les fonctions de commande ou de régulation que le système lui fournit.

3.3.2. Relations entre les niveaux - exemple

Chaque événement de niveau élevé communique avec un ou plusieurs événements du niveau inférieur. La figure 3.3 illustre cette relation et les deux figures 3.4 et 3.5 montrent un exemple visant à l'illustrer.

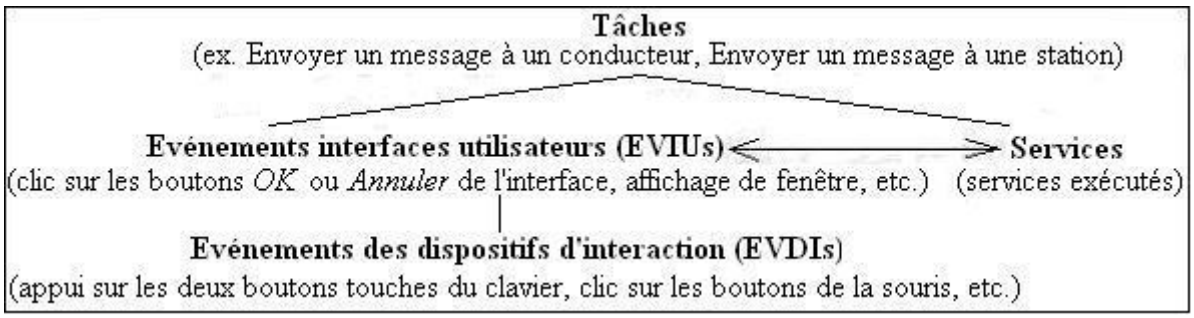

Figure 3.3 :Niveaux abstraits des événements

Comme l'illustre la figure 3.5, les trois EVDIs : le clic du bouton gauche de la souris sur le bouton *OK*, l'appui sur les deux touches *Alt O* ou sur la touche *Entrée* du clavier correspondent à un seul EVIU au niveau intermédiaire : *boutonOK_click*.

L'exemple correspond à une tâche du système SAI (Système d'Aide à l'Information pour la régulation du trafic) qui sera présenté dans le chapitre 4 de ce mémoire. Au niveau intermédiaire, les *services* sont exécutés et les *EVIUs* se produisent pour réaliser une *tâche* au niveau le plus élevé. La figure 3.4 illustre l'agent *interface Station* du SAI sur lequel l'utilisateur veut réaliser une tâche : « *Envoyer un message à une station* » qui est accomplie par les événements au niveau intermédiaire suivant (figure 3.5) :

- Cinq événements interfaces utilisateurs (EVIUs) peuvent se produire sur cet agent *interface Station* du SAI : *ImageStation_Click*, *FenetreStation_Affichée*, *BoîteTexteMessage_Changée*, *boutonOK_Click*.
- Les services de cet agent *interface Station* du SAI sont : le service *Afficher_Fenetre_Proprietes_De_Station* (il est déclenché par l'EVIU *ImageStation_Click* jouant le rôle d'événement déclencheur ; ce service va déclencher l'EVIU *FenetreStation_Affichée*, qui est son action interne) et le service *Envoyer_Un_Message_à_Station* (l'EVIU *boutonOK_Click* étant l'événement déclencheur de ce service).

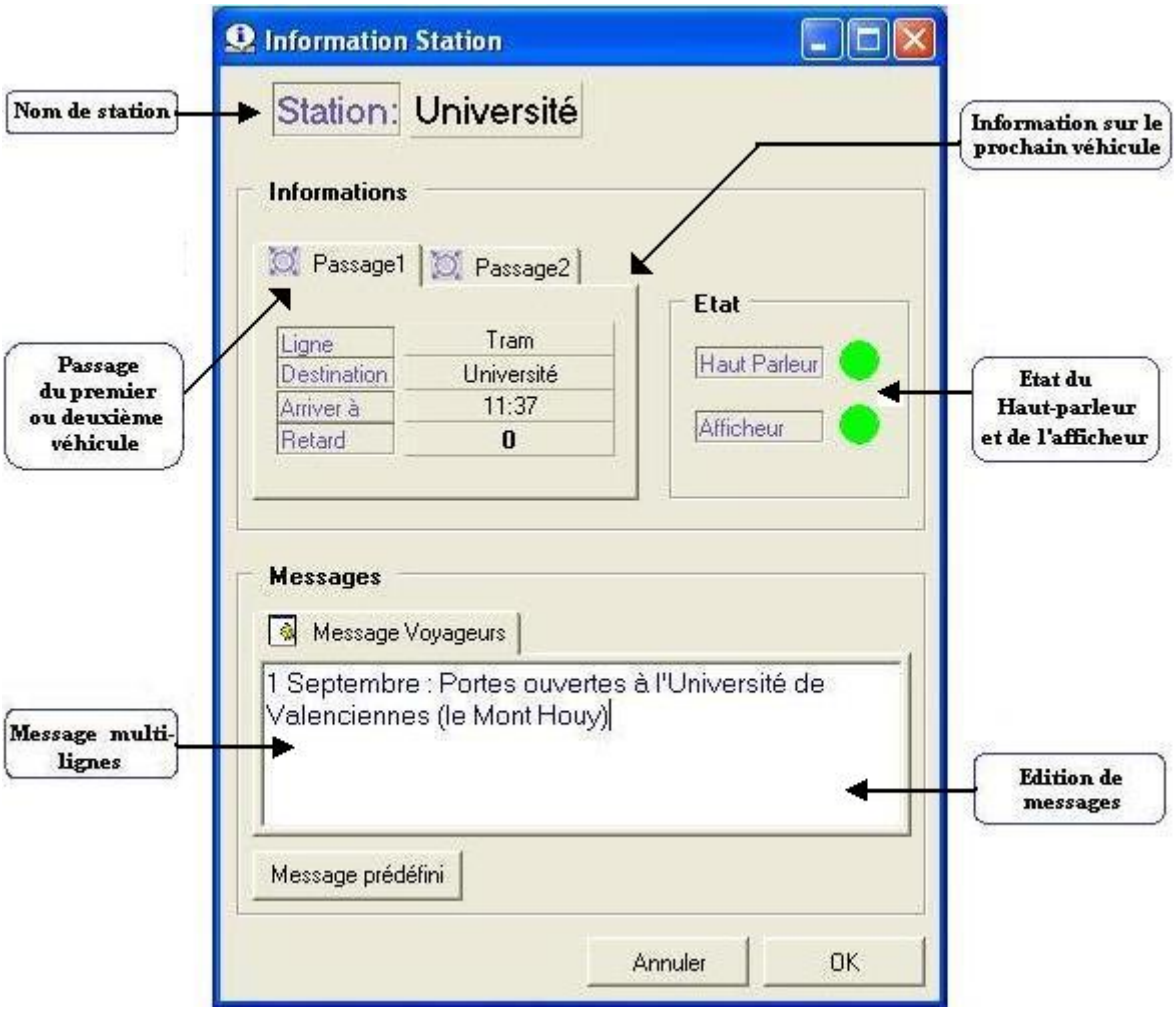

Figure 3.4 : Agent *interface* Station du SAI - réaliser la tâche : envoyer un message aux voyageurs en station

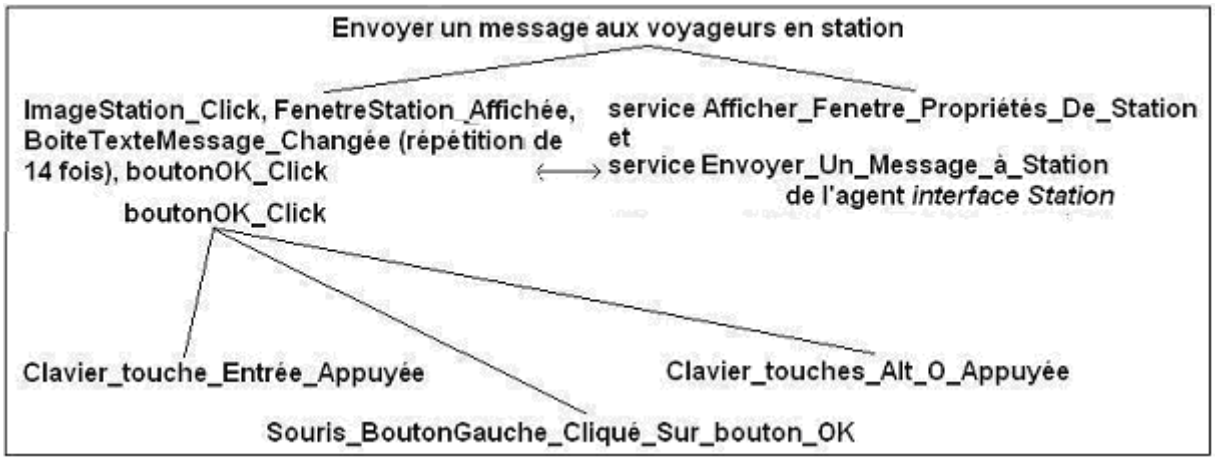

Figure 3.5 : Niveaux abstraits des événements - un exemple

3.4. Description formelle de l'architecture à base d'agents proposée

Pour proposer un environnement générique et configurable pour l'aide à l'évaluation des systèmes interactifs à base d'agents (objectif principal de cette thèse), il est nécessaire de

décrire formellement l'architecture à base d'agents des systèmes interactifs. En fait, une description formelle de cette architecture a été déjà présentée dans [Trabelsi, 2006] mais elle n'est pas complètement exploitable vis-à-vis de notre but. Pour cette raison, notre première tâche dans cette thèse est de proposer une autre description formelle plus complète qui permettra d'atteindre ce but. Les agents et les événements associés doivent être aussi formellement décrits. Nous présentons maintenant cette description.

3.4.1. Description d'un agent

Un agent *application* ou *contrôleur* est décrit par les éléments suivants (séparés par des '|') :

- un identificateur (ID)
- un nom
- une liste des IDs de ses services. Les IDs de cette liste sont séparés par des caractères point-virgule.

  La description d'un agent *application* ou agent *contrôleur* est la suivante :

  ID_agent | nom_agent | ID_service_1 ; ID_service_2 ;…; ID_service_N

  Dans cette description :

  ID_agent est représenté sous forme n-T dans laquelle :

  n : un numéro (*1*, *2*, *3*, etc.) correspond à l'indice de l'agent

  T : une des deux valeurs : *C*, *A* signifiant l'agent *Contrôleur* ou *Application*

  Par exemple, *4-A* est l'ID du quatrième agent *application*, *3-C* est l'ID du troisième agent *contrôleur*.

  nom_agent est une chaîne de caractères. *ID_service* sera présenté ci-dessous.

Un agent *interface* est décrit par les éléments suivants (séparés par des '|') :

- un ID
- un nom
- une liste des IDs de ses services. Les IDs de cette liste sont séparés par des caractères point-virgule.
- une liste des IDs des EVIUs associés. Les IDs de cette liste sont séparés par des caractères point-virgule.

  La description d'un agent *interface* est la suivante :

  ID_agent | nom_agent | ID_service_1 ; ID_service_2 ;…; ID_service_N | ID_EVIU_1 ; ID_EVIU 2 ;… ; ID_EVIU_M

  Dans cette description :

  ID_agent est représenté sous forme n-I dans laquelle :

  n : un numéro (*1*, *2*, *3*, etc.) correspond à l'indice de l'agent

  I signifiant l'agent *Interface*

  Par exemple, *2-I* est l'ID du deuxième agent *interface*

  nom_agent est une chaîne de caractères. *ID_EVIU* sera présenté ci-dessous.

Nous utilisons les descriptions de deux agents du système interactif SAI dans un but d'illustration :

3-I | « Station » | s1,3-I ; s2,3-I ; s3,3-I ; s4,3-I | eviu1,3-I ; eviu2,3-I ;…; eviu10,3-I

4-I | « Véhicule » | s1,4-I ; s2,4-I ;s3,4-I ;…; s8,4-I | eviu1,4-I ; eviu2,4-I ;…; eviu25,4-I

La première description correspond au trosième agent *interface* identifié par *3-I* et nommé par *Station*. Cet agent fournit quatre services identifiés respectivement par *s1,3-I* ; *s2,3-I* ; *s3,3-I* et *s4,3-I*. Il est aussi associé à un ensemble de dix EVIUs identifiés respectivement par *eviu1,3-I* ; *eviu2,3-I* ;…; *eviu10,3-I*.

La seconde description correspond au quatrième agent *interface* identifié par *4-I* et nommé par *Véhicule*. Cet agent fournit huit services identifiés respectivement par *s1,4-I* ; *s2,4-I* ; *s3,4-I ;…; s8,4-I*. Il est aussi associé à un ensemble de vingt-cinq EVIUs identifiés respectivement par *eviu1,4-I* ; *eviu2,4-I* ;…; *eviu25,4-I*.[15]

Le système SAI se compose de six agents *interface* ; il y a donc six descriptions correspondantes.

Nous pouvons constater que la description formelle des agents est insuffisante pour décrire un système interactif à base d'agents. Dans la description formelle d'un agent, nous connaissons seulement les IDs des services et des EVIUs associés mais nous ne connaissons pas du tout les détails de ces services et EVIUs. Pour avoir une description complète d'un système interactif à base d'agents, il est nécessaire de décrire formellement des services et des EVIUs associés. La combinaison des descriptions des agents, services et EVIUs nous fournit une description complète d'un système interactif à base d'agents et nous permet de proposer un environnement générique et configurable pour l'aide à l'évaluation de ces systèmes interactifs. Nous présentons la description formelle des EVIUs et des services maintenant.

3.4.2. Description d'un EVIU

Un EVIU est décrit par les éléments suivants (séparés par des '|') :

- un ID
- un nom
- ID de l'agent *interface* associé
- ID d'un événement déclencheur qui déclenche cet EVIU
- une liste des IDs des services déclenchés par cet EVIU (ce sont les services de même agent *interface* associé). Les IDs de cette liste sont séparés par des caractères point-virgule.
- une liste des IDs des EVIUs déclenchés par cet EVIU (ce sont les EVIUs de même agent *interface* associé). Les IDs de cette liste sont séparés par des caractères point-virgule.

La description d'un EVIU est la suivante :

ID_EVIU | nom_EVIU | ID_agent_associé, ID_événement déclencheur | ID_service_déclenché_1 ; ID_service_déclenché_2 ;…; ID_service_déclenché_N | ID_ EVIU_déclenché_1 ; ID_ EVIU_déclenché_2 ;…; ID_ EVIU_déclenché_M

Dans cette description :

[15] Les points de suspension sont utilisés ici dans un but de concision. Dans la description réelle, tous les IDs des services et des EVIUs de cet agent sont mentionnés.

ID_EVIU est représenté sous forme eviun,m-I dans laquelle :

n : un numéro (*1, 2, 3*, etc.), correspond à l'indice de cet EVIU

m-I est l'ID de l'agent *interface* associé

Par exemple, *eviu2,3-I* est l'ID du deuxième EVIU de l'agent *interface* identifié par *3-I* (troisième agent *interface* comme présenté ci-dessus).

ID_événement_déclencheur peut prendre l'une des trois valeurs : *utilisateur*, ID d'un service de l'agent *interface* associé, ID d'un autre EVIU de l'agent *interface* associé (cf. 3.3.1).

nom_EVIU est une chaîne de caractères. *ID_service* sera présenté ci-dessous.

Nous utilisons les descriptions de trois EVIUs du système interactif SAI dans un but d'illustration :

eviu9,3-I | « Bouton Annuler est cliqué » | 3-I | utilisateur | NULL | eviu3,3-I

eviu3,3-I | « Fenêtre de Propriétés de la Station est Fermée »| 3-I | eviu9,3-I | NULL | NULL

eviu20,4-I | « Boite d'avertissement de perturbation est ouverte » | 4-I | s3,4-I | NULL | NULL

La première description correspond à un EVIU identifié par *eviu9,3-I* et nommé par *Bouton Annuler est cliqué*. L'ID de l'agent *interface* associé est *3-I*. Cet EVIU est déclenché par *l'utilisateur*. Cet EVIU ne déclenche aucun service (décrit par *NULL*) mais il déclenche un autre EVIU identifié par *eviu 3,3-I*.

La deuxième description correspond à un EVIU identifié par *eviu3,3-I* et nommé par *Fenêtre de Propriétés de la Station est Fermée*. L'ID de l'agent *interface* associé est *3-I*. Cet EVIU est déclenché par un autre EVIU identifié par *eviu9,3-I* et il ne déclenche aucun service et aucun EVIU (décrit par *NULL*).

Ces deux premières descriptions expriment que : si l'utilisateur clique le bouton *Annuler* (*eviu9,3-I*) sur la fenêtre de propriétés de l'agent *interface Station (*agent *3-I),* alors cette fenêtre sera fermée (*eviu3,3-I*).

La troisième description correspond à un EVIU identifié par *eviu20,4-I* et nommé par *Boite d'avertissement de perturbation est ouverte*. L'ID de l'agent *interface* associé est *4-I*. Cet EVIU est déclenché par un service identifié par *s3,4-I* et il ne déclenche aucun service et aucun EVIU (décrit par *NULL*).

3.4.3. Description d'un service

Un service est décrit par les éléments suivants séparés par des '|' :

- un ID
- un nom
- ID de l'agent associé
- ID d'un événement déclencheur qui déclenche ce service
- conditions à vérifier pour déclencher ce service
- ressources éventuellement requises pour déclencher ce service

- une liste des IDs des EVIUs déclenchés par ce service (de l'agent associé si cet agent est un agent *interface*). Les IDs de cette liste sont séparés par des caractères point-virgule. Les EVIUs déclenchés par ce service sont appelés ses *actions internes* (*aci*).
- une autre liste des IDs des services déclenchés par ce service (du même agent ou d'un autre agent). Les IDs de cette liste sont séparés par des caractères point-virgule. Les services déclenchés par ce service sont appelés ses *actions externes* (*ace*). Les agents communiquent entre eux par les invocations entre leurs services.
- une autre liste des IDs des agents correspondants dont les services sont déclenchés par ce service. Les IDs de cette liste sont séparés par des caractères point-virgule.

La description d'un service est la suivante :

ID_service | nom_service | ID_agent_associé | événement déclencheur | conditions | ressources | ID_EVIU_déclenché_1 ; ID_EVIU_déclenché_2 ; ID_EVIU_déclenché_3 ; ID_EVIU_déclenché_4 ; ID_EVIU_déclenché_5 ;…; ID_EVIU_déclenché_M | ID_service_déclenché_1 ; ID_service_déclenché_2 ;…; ID_service_déclenché_N | ID_agent_correspondant_au_service_déclenché_1 ; ID_agent_correspondant_au_service_déclenché_2 ;…. ID_agent_correspondant_au_service_déclenché_N

Dans cette description :

ID_service est représenté sous forme sn,m-T dans laquelle :

n : un numéro (*1, 2, 3*, etc.) correspond à l'indice du service

m-I est l'ID de l'agent associé

Par exemple, *s2,3-I* est l'ID du deuxième service de l'agent *interface* identifié par *3-I* , s4,5-A est l'ID du quatrième service de l'agent *application* identifié par 5-A.

ID_événement_déclencheur peut prendre l'une des trois valeurs : *système* (ce service est automatiquement exécuté), ID d'un autre service du même agent ou d'un autre agent), ID d'un EVIU de l'agent associé si cet agent est un agent *interface* (cf. 3.3.1 au-dessus).

nom_service, conditions, ressources sont décrites par des chaînes de caractères.

Nous utilisons la description d'un service du système SAI dans un but d'illustration :

s3,4-I | « Détecter les perturbation des véhicules » | 4-I | systeme | « apparition d'une perturbation du véhicule » | « agent interface 4-I est déjà chargé » | eviu20,4-I | NULL | NULL

Cette description correspond à un service identifié par *s3,4-I* et nommé par *Détecter les perturbation des véhicules*. L'ID de l'agent associé est *4-I*. Ce service est déclenché par *système* (c'est-à-dire qu'il est exécuté automatiquement) lorsqu'un véhicule est perturbé. Ce service ne déclenche aucun service (décrit par *NULL*) mais il déclenche un EVIU identifié par *eviu 20,4-I*.

3.4.4 . Méta-modèle des systèmes interactifs à base d'agents

La formalisation des systèmes interactifs à base d'agents ci-dessus peut être représentée d'une autre manière sous forme d'un diagramme de classe UML. Ce diagramme est essentiellement un méta-modèle de description des systèmes interactifs à base d'agents. Nous en proposons une première version.

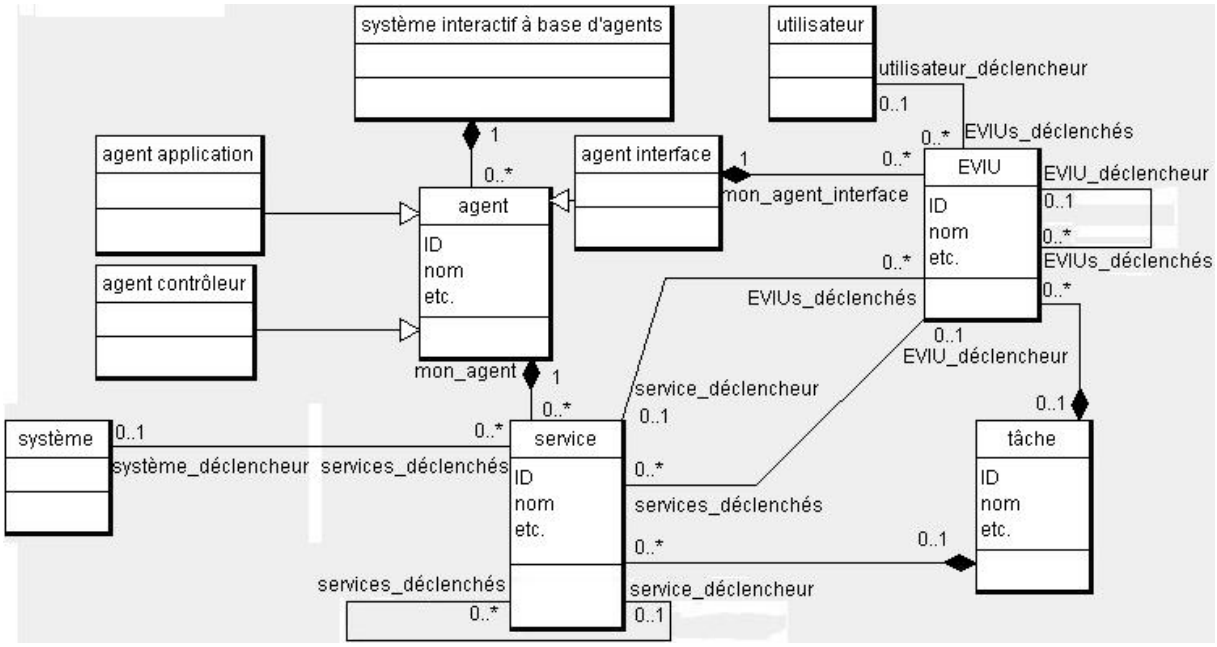

Figure 3.6 : Méta-modélisation des systèmes interactifs à base d'agents

Les contraintes peuvent être décrites en langage OCL (Object Constraint Language) :

context EVIU
inv : if self.EVIUs_déclenchés->size() >= 1
then
self. EVIUs_déclenchés ->forAll(EVIU_déclenché | EVIU_déclenché.mon _agent_interface = self.mon_agent_interface)
endif

inv : if self.services_déclenchés->size() >= 1
then
self. services_déclenchés ->forAll(service_déclenché | service_déclenché.mon_ agent = self. mon_agent_interface)
endif

3.4.5. Exemples

Nous présentons ici une partie des descriptions formelles des agents, des EVIUs et des services du système interactif à base d'agents SAI que nous allons présenter ultérieurement dans ce mémoire. Ces exemples viennent d'être présentés isolément ci-dessus mais il faut les combiner pour avoir une vue plus complète du SAI.

3-I | « Station » | s1,3-I ; s2,3-I ; s3,3-I ; s4,3-I | eviu1,3-I ; eviu2,3-I ;…; eviu10,3-I

4-I | « Véhicule » | s1,4-I ; s2,4-I ; s3,4-I ;…; s8,4-I | eviu1,4-I ; eviu2,4-I ;…; eviu25,4-I

eviu9,3-I | « Bouton Annuler est cliqué » | 3-I | utilisateur | NULL | eviu3,3-I

eviu3,3-I | « Fenêtre de Propriétés de la Station est Fermée »| 3-I | eviu9,3-I | NULL | NULL

eviu20,4-I | « Boite d'avertissement de perturbation est ouverte » | 4-I | s3,4-I | NULL | NULL

s3,4-I | « Détecter les perturbation des véhicules » | 4-I | système | « apparition d'une perturbation du véhicule » | « agent interface 4-I est déjà chargé » | eviu20,4-I | NULL | NULL

etc.

Ces exemples expriment des informations relatives aux deux agents *interface* du SAI (*Station* et *Véhicule*), à quelques services et EVIUs associés, et aux interactions entre eux :

- Si l'utilisateur clique le bouton *Annuler* (*eviu9,3-I*) sur la fenêtre de propriétés de l'agent *interface Station (*agent *3-I),* alors cette fenêtre sera fermée (*eviu3,3-I*).
- Lorsqu'un véhicule est perturbé, le service *Détecter Perturbation Des Véhicules* (*s3,4-I*) est exécuté automatiquement et il va déclencher l'ouverture d'une boîte d'avertissement de perturbation pour l'utilisateur (*eviu20,4-I).*

Un système interactif à base d'agents peut être spécifié par un ensemble de descriptions de ce type qui correspondent aux activités (sous forme des interactions entre des services et des EVIUs associés) des agents du système. Nous présentons maintenant l'environnement EISEval proposé et développé.

3.5. Structure de l'environnement proposé

La figure 3.6 illustre la structure de l'environnement d'évaluation EISEval proposé dans le cadre de cette thèse. C'est une structure modulaire.

Cet environnement a été déjà introduit dans [TRAN et al., 2007] et détaillé dans [TRAN et al., 2008a] et [TRAN et al., 2008b]. Il est composé de sept modules qui ne communiquent pas directement entre eux. Ainsi, le développeur peut modifier les uns sans affecter les autres. Nous présentons chaque module maintenant.

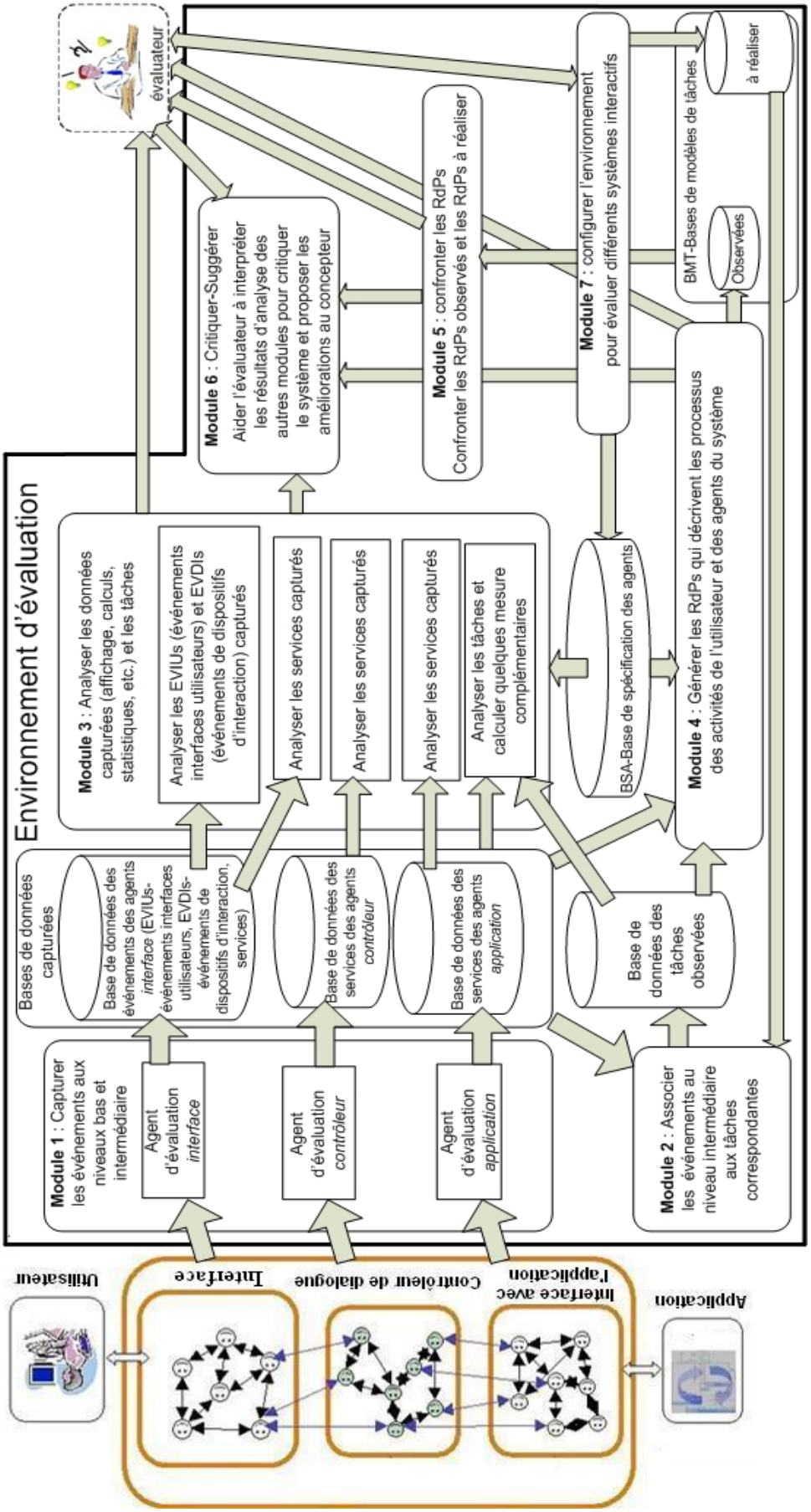

Figure 3.7 : Structure de l'environnement d'évaluation EISEval proposé

3.5.1. Module 1 : Capturer les événements aux niveaux base et intermédiaire

La figure 3.7 illustre ce premier module.

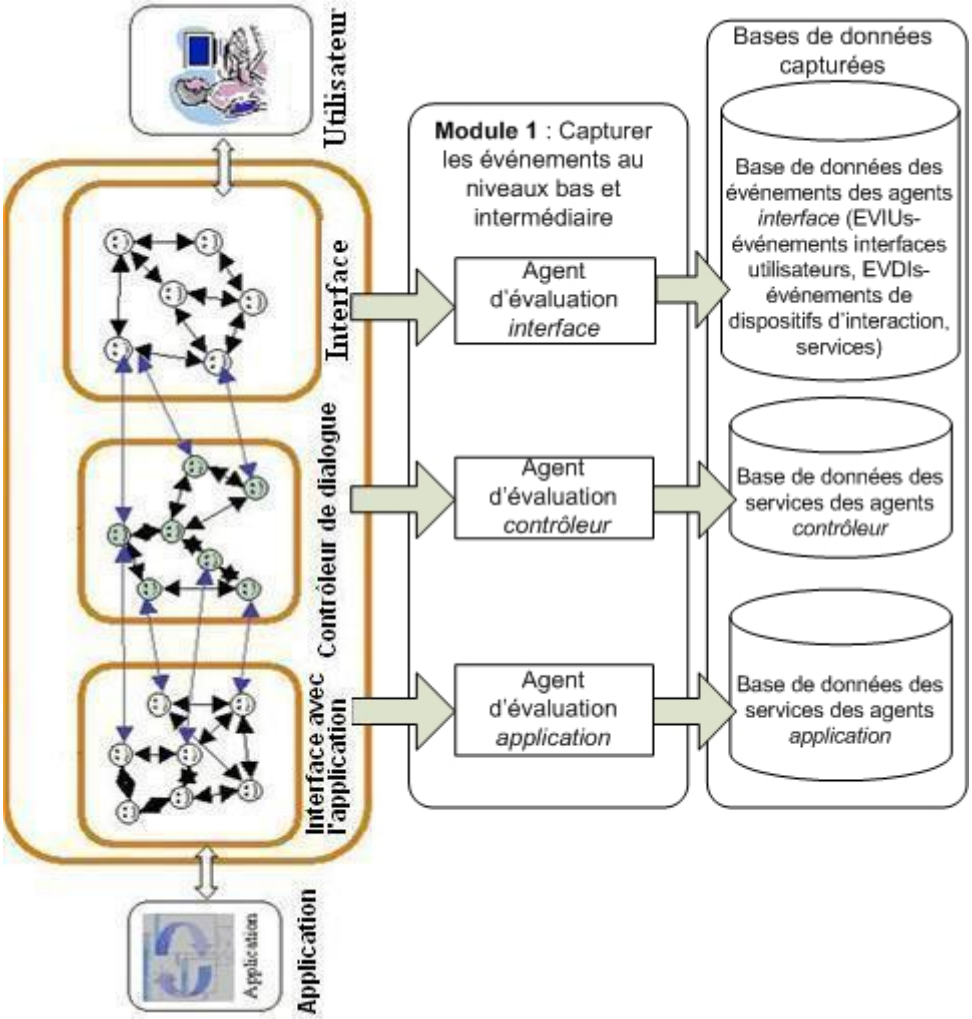

Figure 3.8 : Module 1 - capturer les événements se produisant dans le système interactif à évaluer

Le module 1 est chargé de capturer les événements se produisant dans le système interactif à base d'agents à évaluer. Ces événements capturés peuvent être : les EVDIs (événement de dispositifs d'interaction) au niveau abstrait le plus bas, et/ou les EVIUs se produisant sur les agents *interface*, et/ou les services exécutés par les agents au niveau intermédiaire.

Ce module stocke ces données dans des bases afin que les autres modules puissent les récupérer et les analyser ultérieurement, comme l'illustrent les figures 3.6 et 3.7.

Ce module est composé de trois agents d'évaluation. Pour réaliser les fonctions de capture et de stockage des données, chaque agent d'évaluation est associé à chaque type d'agent (au lieu d'un seul agent spécifique du système interactif à base d'agents à évaluer comme avec le mouchard MESIA). Les trois agents d'évaluation du module 1 sont :

- L'agent d'évaluation *interface* qui capture et stocke dans une base de données, les EVDIs et les EVIUs qui se sont produits, de même que les services exécutés par les agents *interface*.
- L'agent d'évaluation *contrôleur* qui capture et stocke dans une base de données, les services exécutés par les agents *contrôleur*.

- L'agent d'évaluation *application* qui capture et stocke dans une base de données, les services exécutés par les agents *application*.

Le premier problème que nous devons résoudre, afin d'évaluer des systèmes interactifs à base d'agents, est le couplage entre le système interactif à évaluer et l'environnement d'évaluation. Le principe de base de ce couplage est changé par rapport à MESIA, de telle façon que cet environnement proposé soit générique. En effet, avec ce principe de couplage, les agents d'évaluation ne dépendent plus du nombre, de la structure et du contenu des agents *interface* concernés du système à évaluer. Comme déjà précisé, on peut aussi utiliser l'environnement proposé pour évaluer des systèmes interactifs qui ne sont pas à base d'agents. En effet, on considère, dans ce cas, qu'un tel système est composé d'un seul agent *interface*, et le module 1 capture les événements pour les analyser ultérieurement.

Les interactions entre les agents, sous forme des services, sont aussi capturées, pas seulement les interactions entre l'utilisateur et l'interface comme avec les mouchards traditionnels. Eventuellement, si le système interactif à évaluer est un système local ou si l'évaluateur s'intéresse seulement à l'évaluation de l'interface utilisateur, et non aux propriétés non-fonctionnelles du système (comme le temps de réponse, la fiabilité, etc.), alors il faut seulement capturer les EVIUs des agents *interface*.

Ce module 1 peut aussi fonctionner comme une tâche de fond. Pour capturer les événements, il faut le lancer juste après le lancement du système interactif à évaluer ; il n'est pas nécessaire de lancer les autres modules de l'environnement dans la mesure où ce module est construit comme un système indépendant des autres modules (ceux-ci pouvant être lancés plus tard pour analyser les données capturées par le module 1). L'évaluateur ne doit plus lancer le système entier pour ne capturer que les données comme avec le MESIA. Donc, des ressources sont économisées.

Rappelons que tous les modules de l'environnement sont indépendants. Donc, on peut modifier ce module sans affecter les autres et vice versa. La séparation entre la capture et l'analyse des données est assurée.

3.5.2. Module 2 : Associer les événements au niveau intermédiaire (services, EVIUs) aux tâches correspondantes

La figure 3.8 illustre l'activité de ce module. Il récupère les événements au niveau intermédiaire (services, EVIUs) capturés par le module 1 pour que l'évaluateur puisse les associer aux tâches correspondantes au niveau abstrait (le plus élevé). Les associations créées correspondent aux tâches que l'utilisateur a réalisées lorsqu'il a utilisé le système interactif.

La figure 3.9 est une capture d'écran de l'interface du module 2. Cette interface montre une partie des données capturées (par le module 1) lors d'une expérimentation exploitant cet environnement pour l'évaluation du système interactif à base d'agents SAI. Notons que les détails de cette expérimentation seront présentés dans le chapitre suivant.

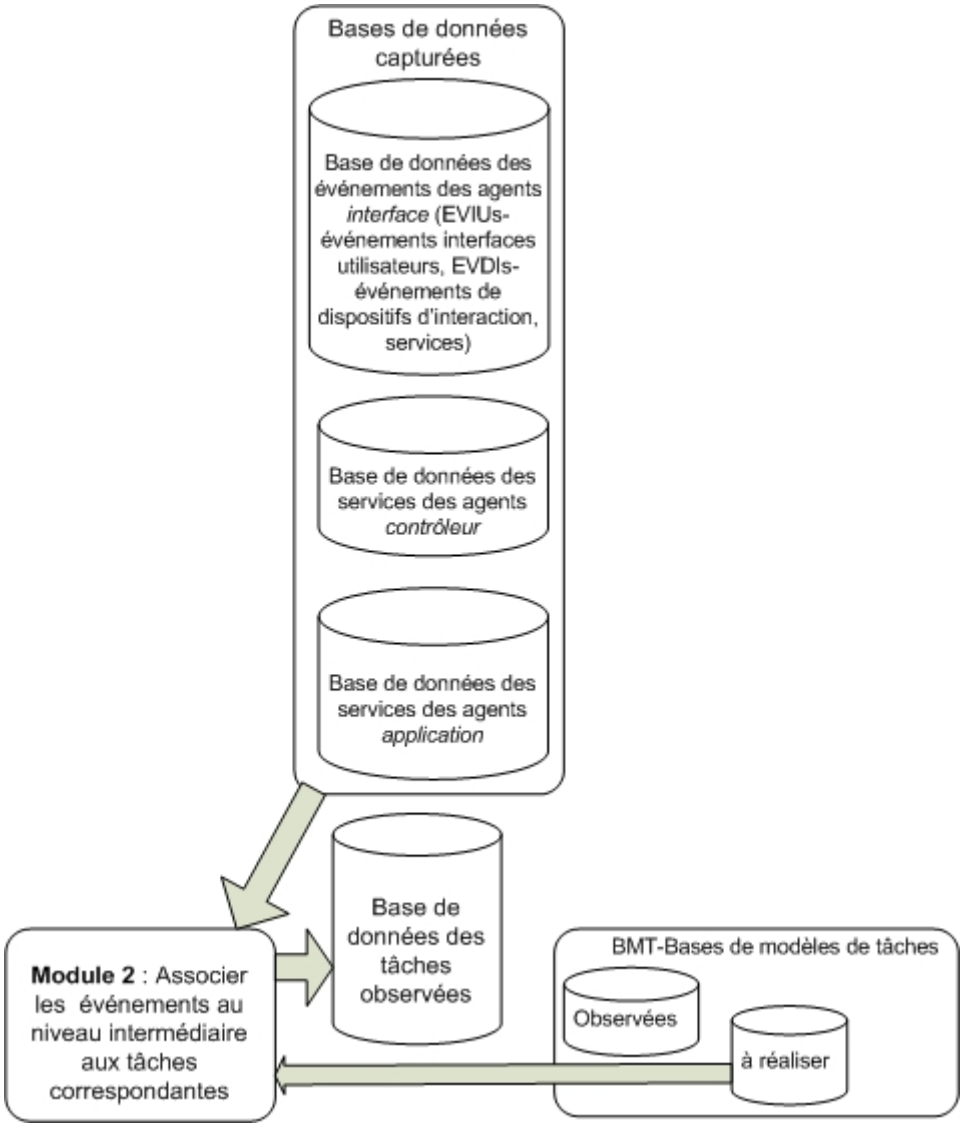

Figure 3.9 : Module 2 : associer les événements au niveau intermédiaire (services, EVIUs) aux tâches correspondantes

L'activité de ce module est montrée sur son interface (comme l'illustre la figure 3.9) :

- Le module 2 affiche les EVIUs déclenchés par l'utilisateur et les services déclenchés par le système en partie gauche de l'interface.
- Le module 2 accède à la base de tâches susceptibles d'être réalisées, afin de les récupérer et les afficher en partie droite de l'interface. Les tâches susceptibles d'être réalisées sont spécifiées par le concepteur du système ; c'est pourquoi, nous pouvons les appeler : *tâches à réaliser* ou *tâches théoriques* ou *tâches prévue*s.
- L'évaluateur peut associer les EVIUs et les services affichés en partie gauche aux tâches correspondantes qui sont affichées en partie droite pour créer les associations affichées dans la partie du bas.
- Comme nous venons de l'aborder, les associations créées sont bien les tâches que l'utilisateur a déjà réalisées lorsqu'il a utilisé le système interactif. Nous pouvons les appeler : *tâches observées* ou *tâches réelles*. L'évaluateur peut enregistrer des telles tâches dans la base de données des tâches observées pour être analysées ultérieurement par les autres modules de l'environnement.

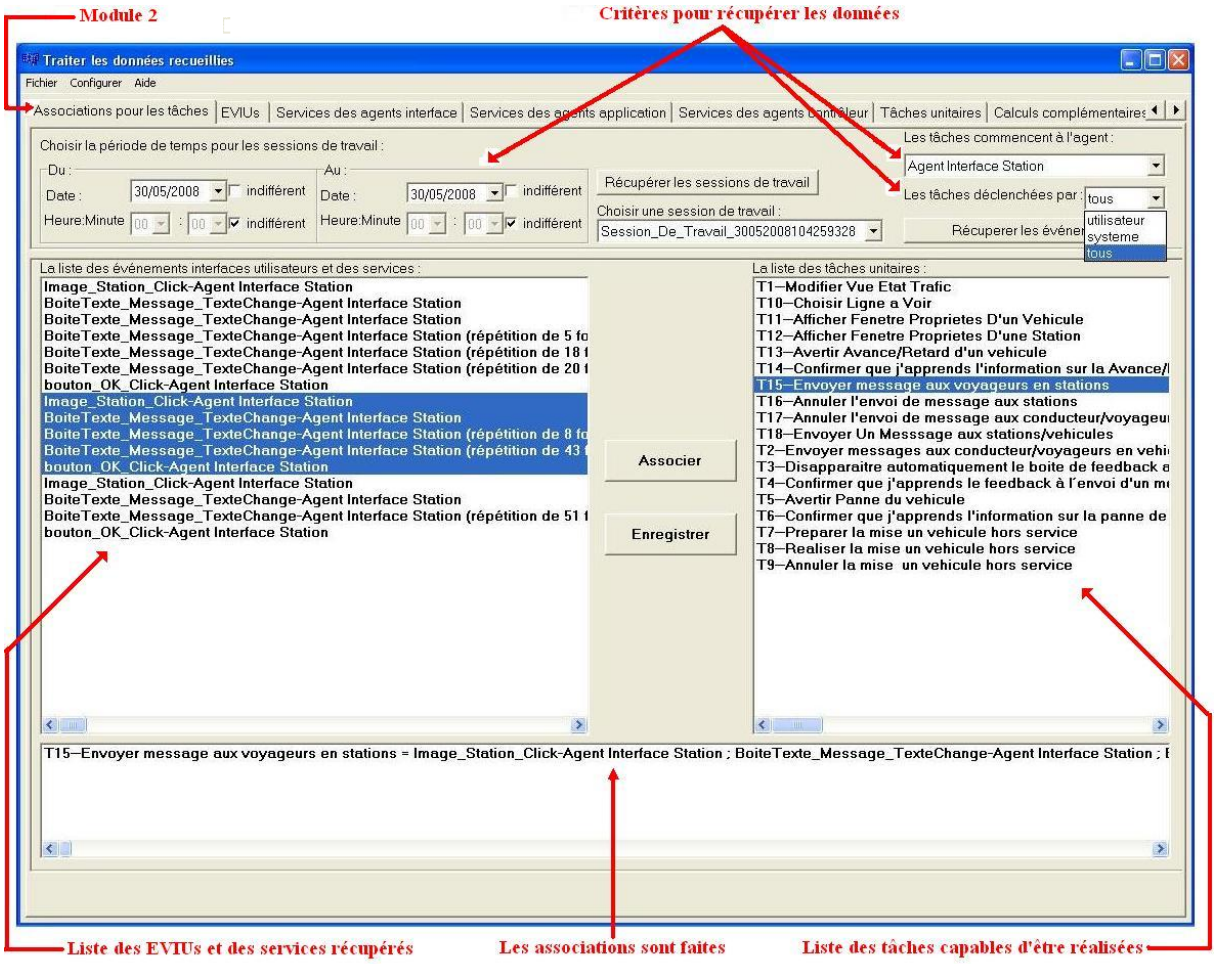

Figure 3.10 : Capture d'écran de l'interface du module 2 - exemple

Dans l'exemple de la figure 3.9, l'évaluateur peut associer les EVIUs comme *ImageStation_Click*, *BoîteTexteMessage_TexteChangée*, *boutonOK_Click* à la tâche correspondante « *Envoyer un message aux voyageurs en station* » pour créer une association. Les EVIUs et les services déclenchés par ces EVIUs comme le service *Afficher_Fenetre_Proprietes_De_Station* (dont l'EVIU *ImageStation_Click* est l'événement déclencheur et l'EVIU *FenetreStation_Affichée* est déclenché par ce service), le service *Envoyer_Un_Message_à_Station* (dont l'EVIU *boutonOK_Click* est l'événement déclencheur), ils sont automatiquement traités par le module 2 quand l'association est créée. Cette association signifie que la tâche « *Envoyer un message aux voyageurs en station* » correspond à une des tâches réalisées en réalité par l'utilisateur ; l'évaluateur peut l'enregistrer dans la base de données des tâches observées pour une analyse ultérieure.

3.5.3. Module 3 : Analyser les données capturées et les tâches

La figure 3.10 illustre l'activité du module 3. Ce module récupère les données capturées par le module 1 (les EVIUs, les EVDIs, les services) et les données des tâches observées créées par le module 2 pour les analyser (statistiques, calculs des mesures, etc.). Ensuite, il affiche les résultats d'analyse sous des formes compréhensibles pour l'utilisateur. Ces résultats d'analyse peuvent être interprétés par l'évaluateur à l'aide du module 6 pour critiquer le système et suggérer les améliorations nécessaires au concepteur. Les figures ci-dessous sont

des captures d'écran du module 3 qui montrent des résultats d'analyses réalisées sur les données capturées du sujet 1 parmi dix sujets pendant une expérimentation.

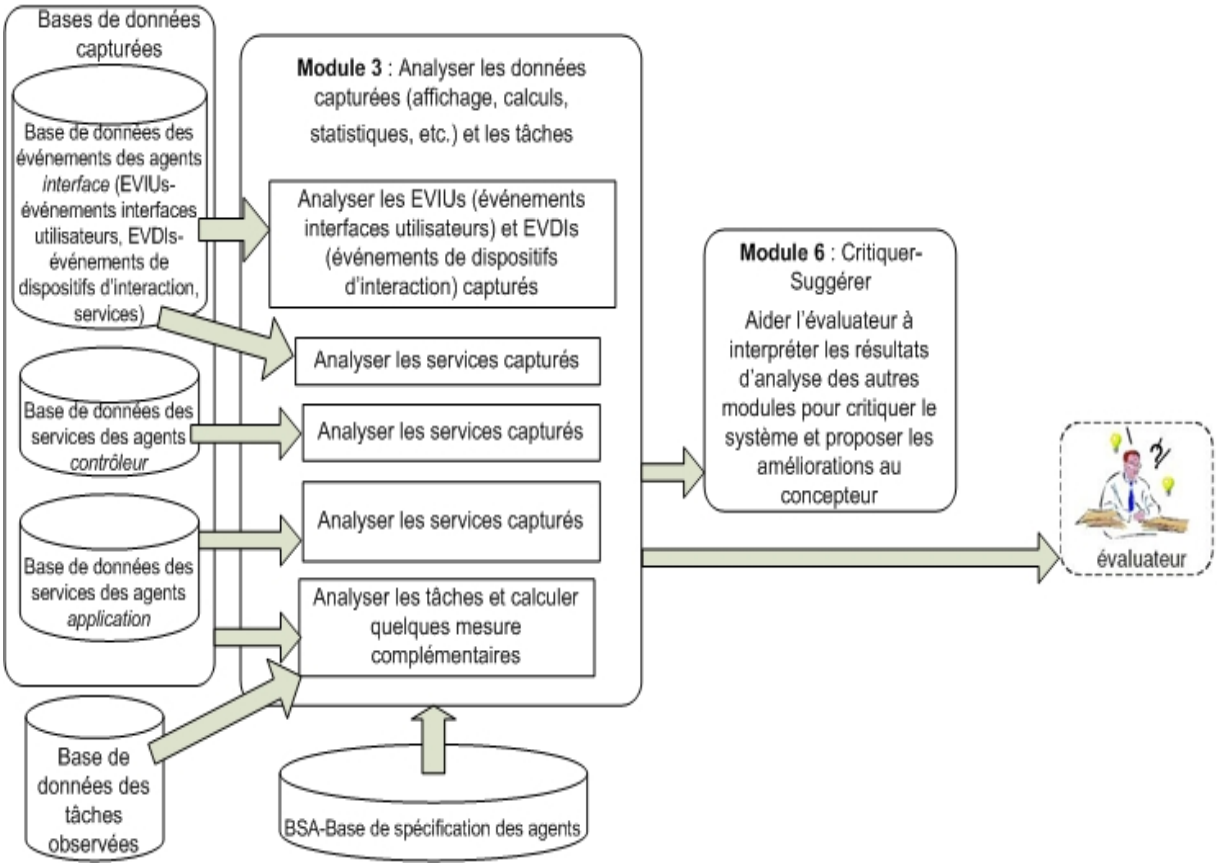

Figure 3.11 : Module 3 : analyser les données capturées par le module 1 et les tâches

a) Analyse des EVIUs

Le module 3 permet de compter le nombre des EVIUs qui se sont produits sur un agent *interface* quelconque sélectionné ou sur tous les agents *interface* du système interactif. Le pourcentage d'apparition de chaque EVIU est calculé aussi. Ces résultats d'analyse peuvent être affichés sous forme de tableau ou sous forme graphique. La fréquence d'apparition des EVIUs est utile pour l'évaluateur afin d'évaluer l'interface et pour suggérer des améliorations nécessaires. Les figures 3.11 et 3.12 représentent des captures d'écran du module 3 où le nombre d'apparitions des EVIUs sur l'agent *interface Etat_Trafic* du SAI est comptabilisé et où le pourcentage d'apparition de ces EVIUs est aussi calculé. La figure 3.12 représente une partie de la capture d'écran du module 3 affichant le résultat d'analyse sous forme graphique. La figure 3.11 représente la capture d'écran du module 3 affichant le résultat d'analyse sous forme de tableau. Comme l'illustre cette figure 3.11, le module 3 peut fournir à l'évaluateur des informations détaillées sur chaque EVIU qui s'est produit (moment d'apparition de l'EVIU, événement déclencheur de cet EVIU, services et/ou EVIUs déclenchés par cet EVIU, etc.).

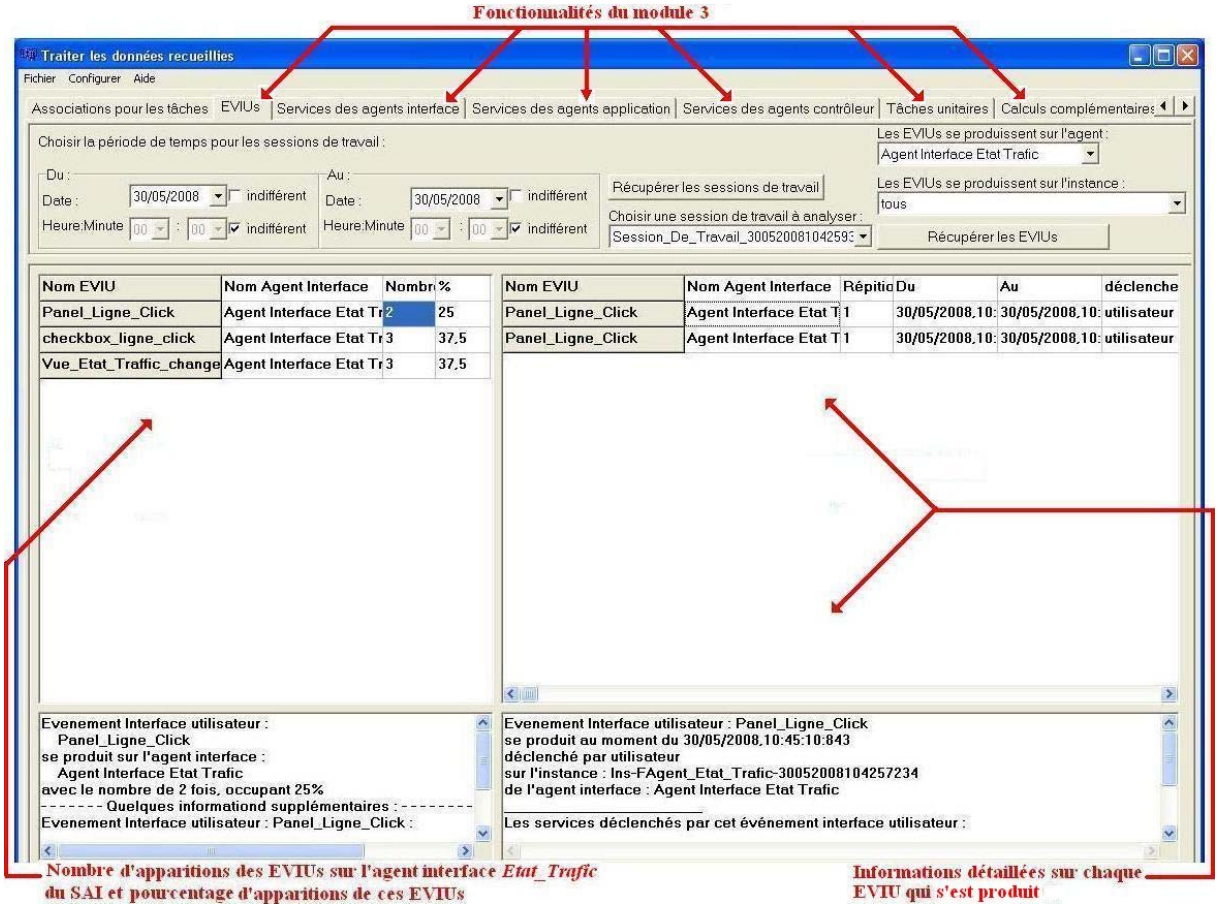

Figure 3.12 : Capture d'écran du module 3 relativement au nombre d'apparitions des EVIUs sur un agent *interface* sélectionné ou sur tous les agents *interface*, et au calcul de pourcentages d'apparition de ces EVIUs (format tableau).

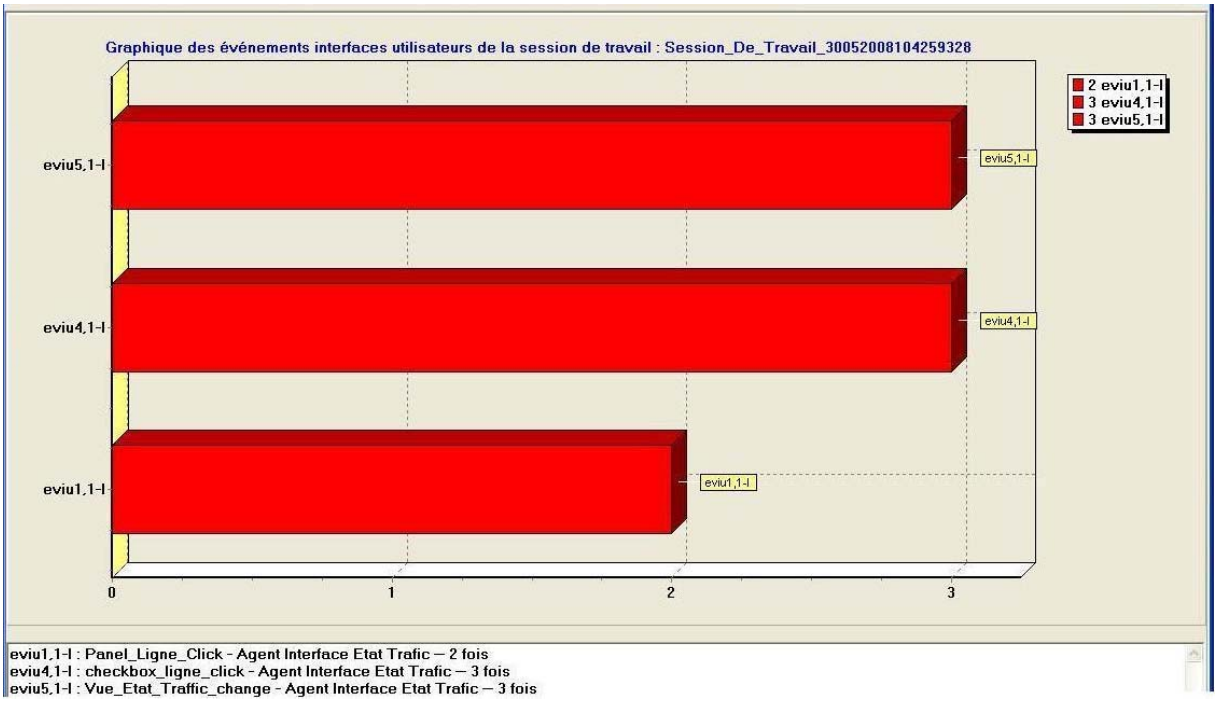

Figure 3.13 : Capture d'écran du module 3 relativement au nombre d'apparitions des EVIUs sur un agent *interface* sélectionné ou sur tous les agents *interface*, et au calcul de pourcentages d'apparition de ces EVIUs (format graphique).

b) Analyse des services

Le module 3 peut comptabiliser le nombre des services exécutés d'un agent quelconque sélectionné (agent *interface*, *contrôleur* ou *application*) ou de tous les agents. Le pourcentage d'exécution de ces services est calculé aussi. Ces résultats d'analyse peuvent être affichés sous forme de tableau ou sous forme graphique. La fréquence d'exécution des services est utile pour l'évaluateur afin d'évaluer le système et pour suggérer des améliorations nécessaires. Les figures 3.13 et 3.14 représentent des captures d'écran du module 3 où le nombre des services exécutés de l'agent *interface Etat_Trafic* du SAI est comptabilisé et où le pourcentage d'exécution de ces services est calculé. Comme l'illustre la figure 3.13, le module 3 peut aussi fournir à l'évaluateur des informations détaillées sur chaque service exécuté (moment d'exécution du service, événement déclencheur de ce service, services et/ou EVIUs déclenchés par ce service, etc.).

Le module 3 peut déterminer le résultat d'exécution des services. Un service peut exécuter une ou plusieurs actions (les actions internes qui déclenchent les EVIUs de l'agent associé et/ou les actions externes qui déclenchent les autres services). Si un service ne peut pas exécuter toutes ses actions (à cause d'une raison quelconque, par exemple, le système a échoué, il n'y a pas de réseau, les ressources ne sont pas suffisantes, etc.), alors le résultat d'exécution de ce service est considéré en « *échec complet* ». Par exemple, un service de détection de perturbation peut échouer dans le déclenchement d'un service d'avertissement de perturbation pour l'utilisateur. Si un service peut exécuter toutes ses actions, alors le résultat d'exécution de ce service est considéré en « *succès complet* ». Si un service ne peut exécuter que quelques actions parmi un ensemble, alors le résultat d'exécution de ce service est considéré en « *succès partiel* ». Comme nous venons de l'aborder, le module 3 peut aussi fournir à l'évaluateur des informations détaillées sur chaque service exécuté comme l'illustre la figure 3.13. Le résultat d'exécution d'un service sous forme textuelle constitue une partie de cette information. Le module 3 peut fournir :

- le nombre de succès et/ou d'échecs de l'exécution d'un service quelconque sélectionné d'un agent quelconque sélectionné.
- le nombre de succès et/ou d'échecs de l'exécution de tous les services d'un agent quelconque sélectionné.
- le nombre des succès et/ou d'échecs de l'exécution de tous les services de tous les agents.

Ce calcul concernant le nombre de succès de l'exécution des services est très utile pour l'évaluateur afin d'évaluer la fiabilité et le fonctionnement du système. La figure 3.15 affiche une partie de la capture d'écran du nombre de succès et/ou d'échecs de l'exécution de tous les services de l'agent *interface Etat_Trafic* du SAI sous forme graphique. La figure 3.20 affiche la capture d'écran de quelques calculs complémentaires effectués par le module 3. Ces calculs contiennent le nombre de succès et/ou d'échecs de l'exécution liés à tous les services de tous les agents.

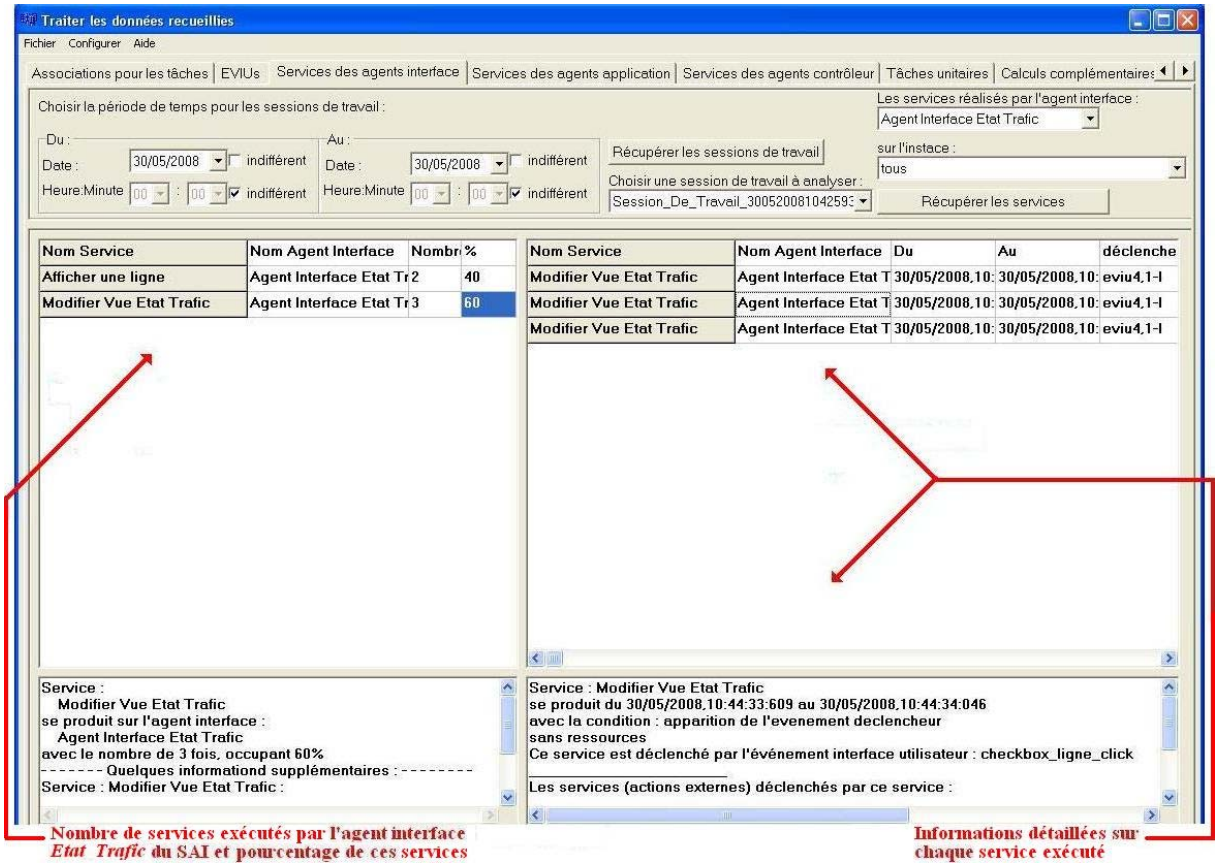

Figure 3.14 : Capture d'écran du module 3 relativement au nombre des services exécutés d'un agent sélectionné ou de tous les agents, et au calcul de pourcentages d'exécution de ces services (format tableau)

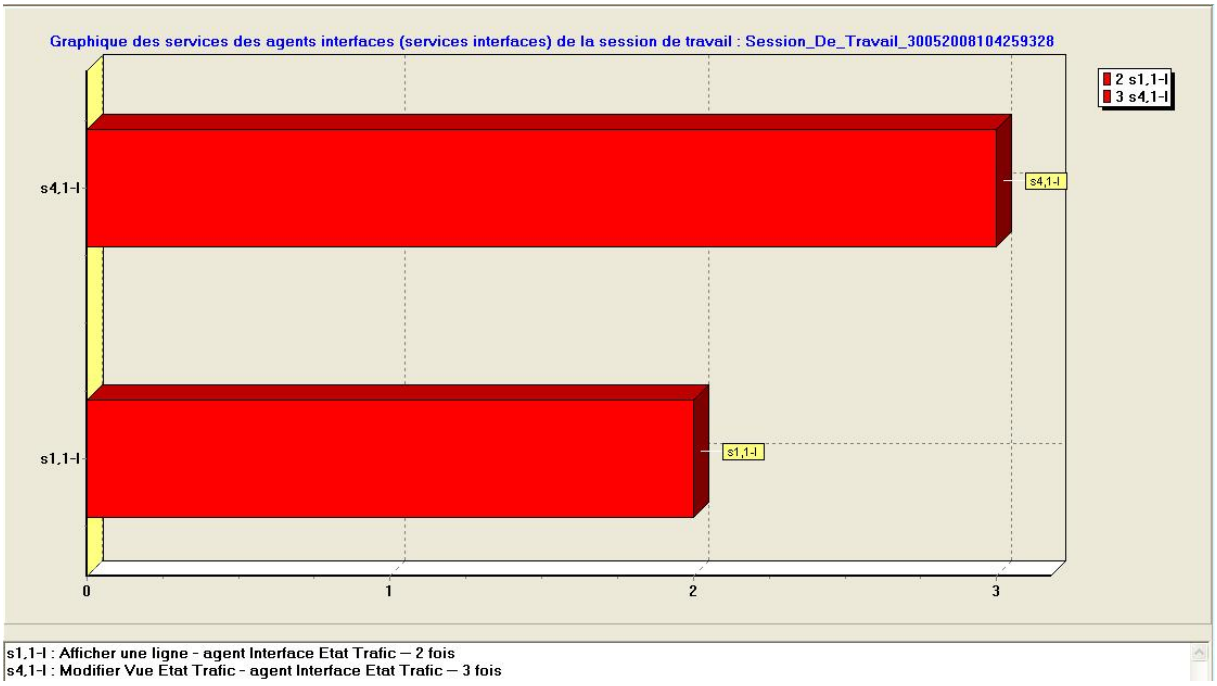

Figure 3.15 : Capture d'écran du module 3 relativement au nombre des services exécutés d'un agent sélectionné ou de tous les agents, et au calcul de pourcentages d'exécution de ces services (format graphique).

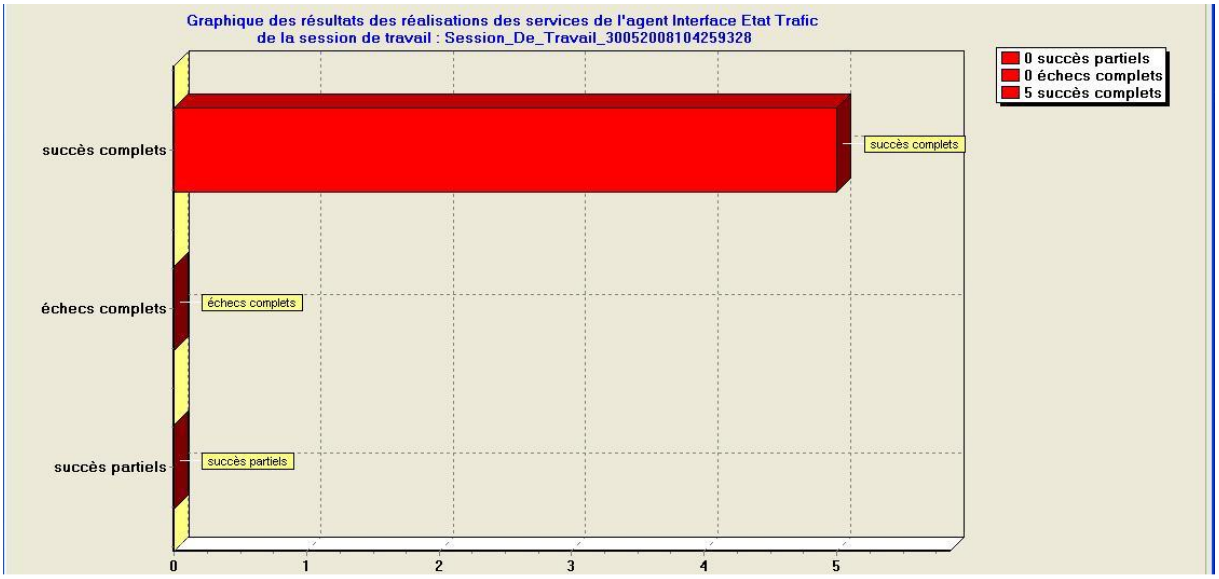

Figure 3.16 : Capture d'écran du module 3 relativement au nombre de succès ou/et d'échecs de d'exécution de tous les services d'un agent quelconque sélectionné (format graphique).

c) Analyse des tâches

Le module 3 peut comptabiliser le nombre de tâches que l'utilisateur a réalisées. Ces tâches sont appelées *tâches observées* ou *tâches réelles*. Le pourcentage de réalisation de ces tâches est calculé aussi. Ces résultats d'analyse peuvent être affichés sous forme de tableau ou sous forme graphique. La fréquence de réalisation des tâches est utile pour l'évaluateur afin d'évaluer les trois aspects du système et de suggérer des améliorations nécessaires. Les figures 3.16 et 3.17 représentent des captures d'écran du module 3, où le nombre des tâches réalisées est comptabilisé et où le pourcentage de réalisation de ces tâches est calculé. La figure 3.17 montre une partie de la capture d'écran affichant ce résultat d'analyse sous forme graphique. La figure 3.16 représente la capture d'écran affichant ce résultat d'analyse sous forme de tableau. Le module 3 peut fournir à l'évaluateur des informations détaillées sur chaque tâche réalisée (moment de début et de fin de la réalisation de cette tâche, résultat de réalisation de cette tâche, etc.), cf. figure 3.16. Les tâches peuvent aussi être affichées dans l'ordre chronologique comme l'illustre la figure 3.18 pour informer l'évaluateur du processus de réalisation des tâches. Notons que ces informations sont exploitées par le module 6 pour évaluer le système.

Le résultat de réalisation d'une tâche peut conduire à un *succès* ou un *échec*. Si une tâche a échoué, alors, la cause peut avoir pour origine le *système* et/ou *l'utilisateur*:

- *Système*. L'exécution d'un ou de plusieurs services nécessaires pour la réalisation de cette tâche a échoué. Par exemple, une tâche d'avertissement de perturbation pour l'utilisateur est initialisée par un service de détection de perturbation qui doit déclencher un autre service d'avertissement de perturbation. Mais en réalité, le résultat d'exécution du service de détection de perturbation est un échec parce qu'il a échoué dans sa tentative de déclenchement du service d'avertissement de perturbation. Dans ce cas, le module 3 marque en « *échec* » le résultat d'exécution de cette tâche d'avertissement de perturbation.

- *Utilisateur*. L'utilisateur a effectué de fausses manipulations pour la réalisation d'une tâche, et donc la tâche a échoué à cause de l'utilisateur. Par exemple, l'utilisateur doit envoyer une commande ou un message dont le contenu est « A » à l'application qu'il supervise mais l'utilisateur s'est trompé, il a déjà saisi et envoyé un message dont le

contenu a été « B ». Dans ce cas, cette tâche a échoué à cause de l'utilisateur et le module 3 ne peut pas détecter ce type d'échec. C'est bien l'évaluateur qui doit détecter le succès ou l'échec de cette tâche, par une méthode quelconque, et ensuite, le module 3 permet à l'évaluateur de marquer en « *échec* », le résultat de réalisation de cette tâche d'envoi.

Le module 3 peut compter le nombre de succès et d'échecs de la réalisation d'une tâche quelconque sélectionnée ; il peut afficher le résultat sous forme graphique. Le module 3 peut aussi compter le nombre de succès et d'échecs de la réalisation de toutes les tâches avec affichage sous forme graphique (comme l'illustre la figure 3.19) ou textuelle (cf. figure 3.20). Ces deux figures montrent le nombre des succès et échecs de la réalisation de toutes les tâches. La figure 3.20 montre une capture d'écran avec affichage de quelques calculs complémentaires à ce sujet. S'il y a des tâches qui ont échoué, le module 3 peut compter le nombre d'échecs ayant pour origine le *système* et *l'utilisateur*. Là encore, le résultat peut être affiché sous forme textuelle ou graphique. Ce calcul concernant le nombre des succès de la réalisation des tâches est utile pour l'évaluateur afin d'évaluer le fonctionnement du système et la performance de l'utilisateur.

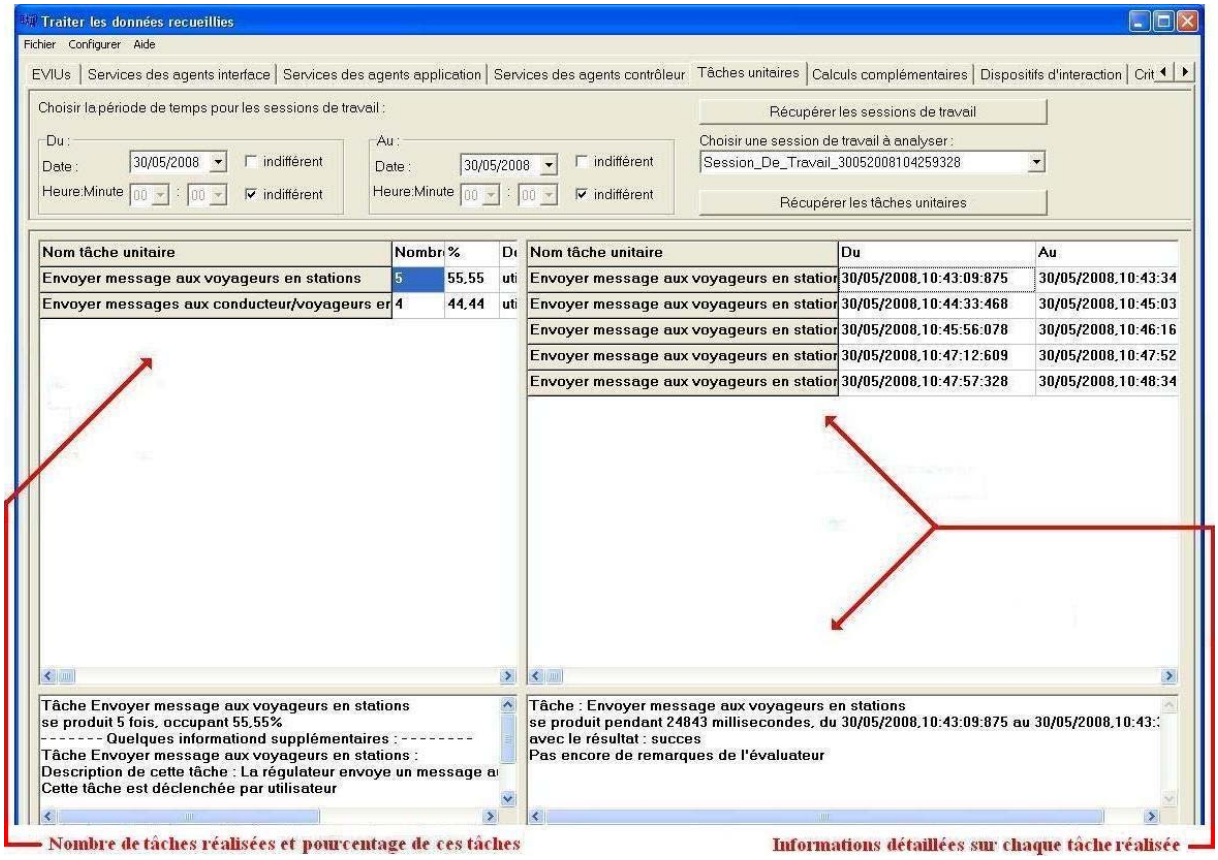

Figure 3.17 : Capture d'écran du module 3 relativement au nombre de tâches réalisées et au calcul de pourcentages de réalisation de ces tâches (format tableau).

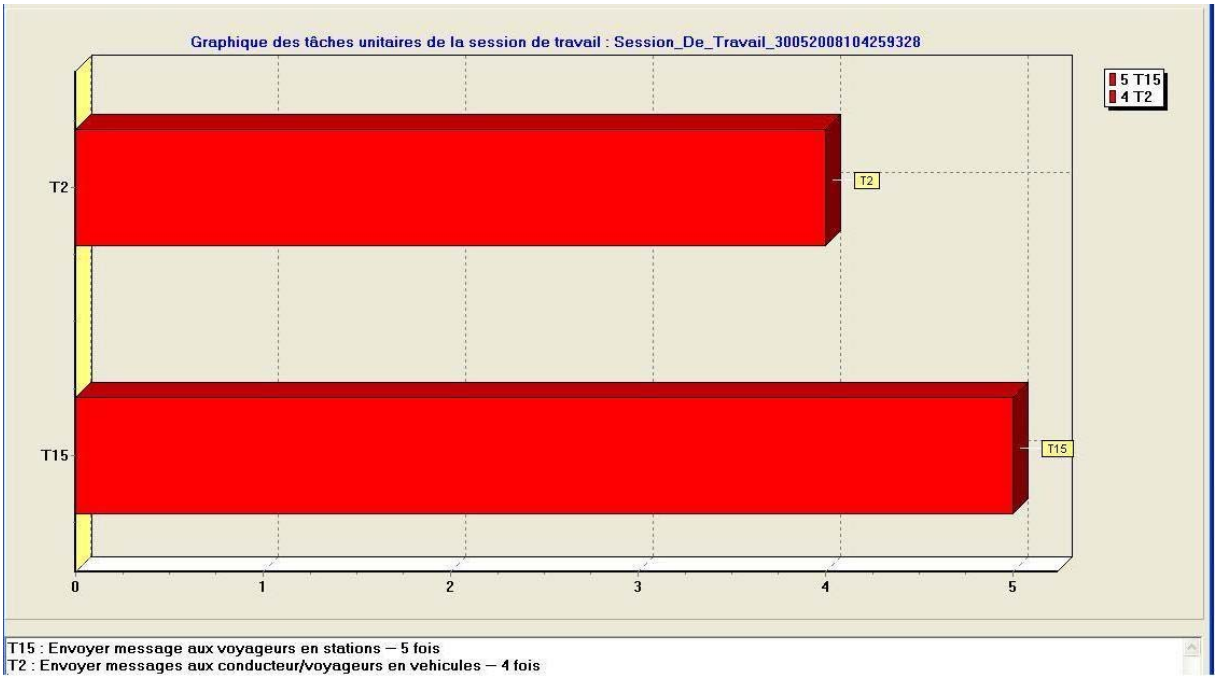

Figure 3.18 : Capture d'écran du module 3 relativement au nombre de tâches réalisées et au calcul de pourcentages de réalisation de ces tâches (format graphique).

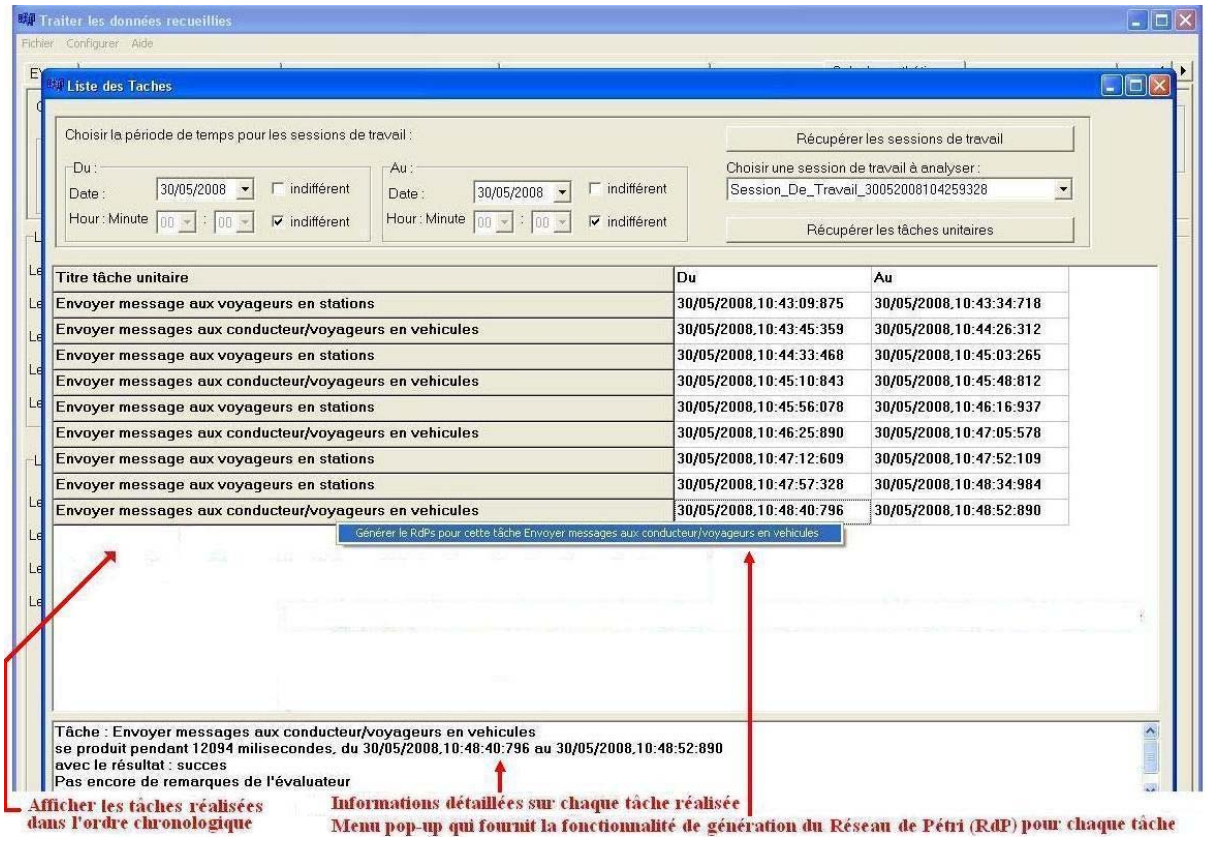

Figure 3.19 : Capture d'écran du module 3 – Les tâches sont affichées dans l'ordre chronologique.

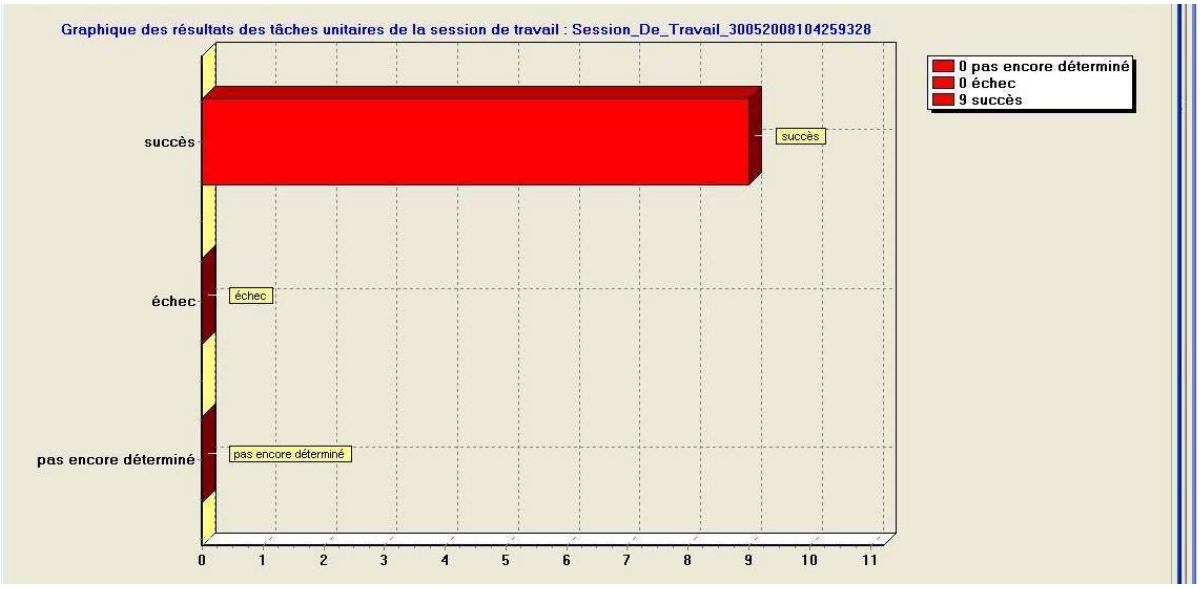

Figure 3.20 : Capture d'écran du module 3 relativement au nombre de succès et d'échecs liés à la réalisation de toutes les tâches.

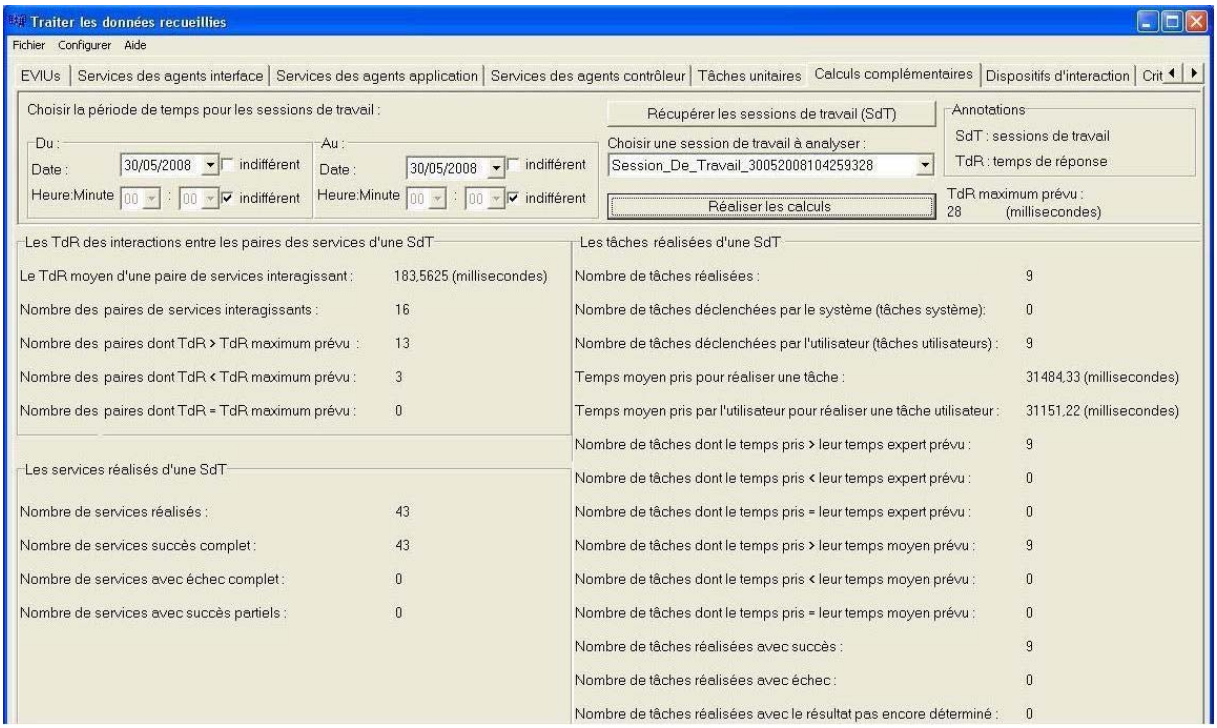

Figure 3.21 : Capture d'écran du module 3 relativement aux mesures complémentaires effectuées par le module 3.

d) Calcul de quelques mesures complémentaires

Le module 3 peut effectuer un ensemble de calculs liés aux services et tâches et ensuite, faire la comparaison entre les valeurs calculées et les valeurs théoriques prévues par le concepteur comme :

- Le nombre d'interactions entre les services,

- Le temps de réponse (TdR) moyen des interactions entre les services. Cette mesure est utile pour évaluer la performance du système. La comparaison entre le TdR réel et le TdR théorique prévu par le concepteur informe l'évaluateur si le système est aussi rapide que spécifié par le concepteur. Le module 3 permet de compter le nombre de TdRs plus/moins/aussi long que le TdR théorique comme visible en figure 3.20.
- Le nombre des services exécutés,
- Le nombre de succès et d'échecs de l'exécution de tous les services.
- Le nombre total de tâches réalisées, le nombre de tâches systèmes et de tâches utilisateurs réalisées.
- Le nombre de succès et d'échecs de la réalisation de toutes les tâches (comme abordé plus haut). S'il y a des tâches qui ont échoué, alors le nombre d'échecs causés par le système, l'utilisateur ou les deux sont comptabilisés et affichés.
- Le temps moyen que l'utilisateur et le système ont pris pour réaliser une tâche et le temps moyen que l'utilisateur a pris pour réaliser une tâche utilisateur.
- Pour chaque tâche, le concepteur peut avoir prévu le temps nécessaire pour sa réalisation. Le concepteur peut prévoir deux types de temps : le temps *expert* qui est le temps acceptable pour réaliser une tâche dans le cas d'un utilisateur expérimenté et le temps *moyen* qui est le temps acceptable pour réaliser une tâche dans le cas d'un utilisateur moyen. Le module 3 permet la comparaison entre les temps réellement mis et les temps prévus ; il permet de compter le nombre de tâches dont le temps effectif est plus/moins/aussi long que le temps expert/moyen prévu (cf. figure 3.20). Cette confrontation est utile pour l'évaluateur afin de détecter les problèmes du système et d'étudier certaines propriétés de l'utilisateur.

3.5.4. Modules 4 et 5 : Générer et confronter les réseaux de Pétri (RdPs)

a) Rappel sur les réseaux de Pétri

Les réseaux de pétri (RdPs) ont été proposés au début des années soixante par C.A. Petri [Petri, 1962]. Cet outil graphique et mathématique permet d'analyser et de formaliser le comportement des systèmes dynamiques où les événements et la concurrence se produisent souvent. Les RdPs sont très utilisés dans le domaine de la modélisation et de la conception des systèmes à événements discrets et ils permettent aussi d'évaluer la performance des systèmes. Les notions comme la synchronisation, la lecture, le partage de ressource, la capacité limitée, le parallélisme, la mémorisation, etc., peuvent être modélisés [David et Alla, 1992]. Trois types d'objets constituent les RdPs ordinaires :

- Les places qui représentent les états possibles du système. Une place est graphiquement représentée par un cercle.
- Les transitions qui représentent les opérateurs de changement d'état du système. Une transition est graphiquement représentée par un trait ou une boîte.
- Les arcs orientés qui sont utilisés pour les liaisons entre les places et les transitions. S'il y a un arc orienté qui fait la liaison d'une place vers une transition, alors cette place est une place d'entrée de cette transition. S'il y a un arc orienté qui fait la liaison d'une transition vers une place, alors cette place est une place de sortie de cette transition.

En fait, plusieurs variantes des réseaux de Petri existent comme les réseaux colorés [Jensen, 1992], les réseaux temporisés [David et Alla, 1992], etc. Dans notre approche, les places et les transitions d'un RdP ordinaire sont utilisées pour décrire les actions de

l'utilisateur (sous forme des EVIUs qui sont déclenchés par l'utilisateur et qui se produisent sur les agents *interface*) et les actions du système (sous forme des services exécutés par des agents) pour réaliser une tâche quelconque. Les modules 4 et 5 permettent de générer et de comparer les RdPs qui décrivent les actions réelles de l'utilisateur et du système pour réaliser une tâche. Nous appelons ces RdPs : *RdPs observés* ou *RdPs réels*. Le concepteur peut aussi utiliser les RdPs pour spécifier les différents chemins pour réaliser une tâche. Nous appelons ces RdPs : *RdPs à réaliser* ou *RdPs théoriques* ou *RdPs prévus*.

b) Modules 4 et 5

La figure 3.21 illustre l'activité de ces deux modules.

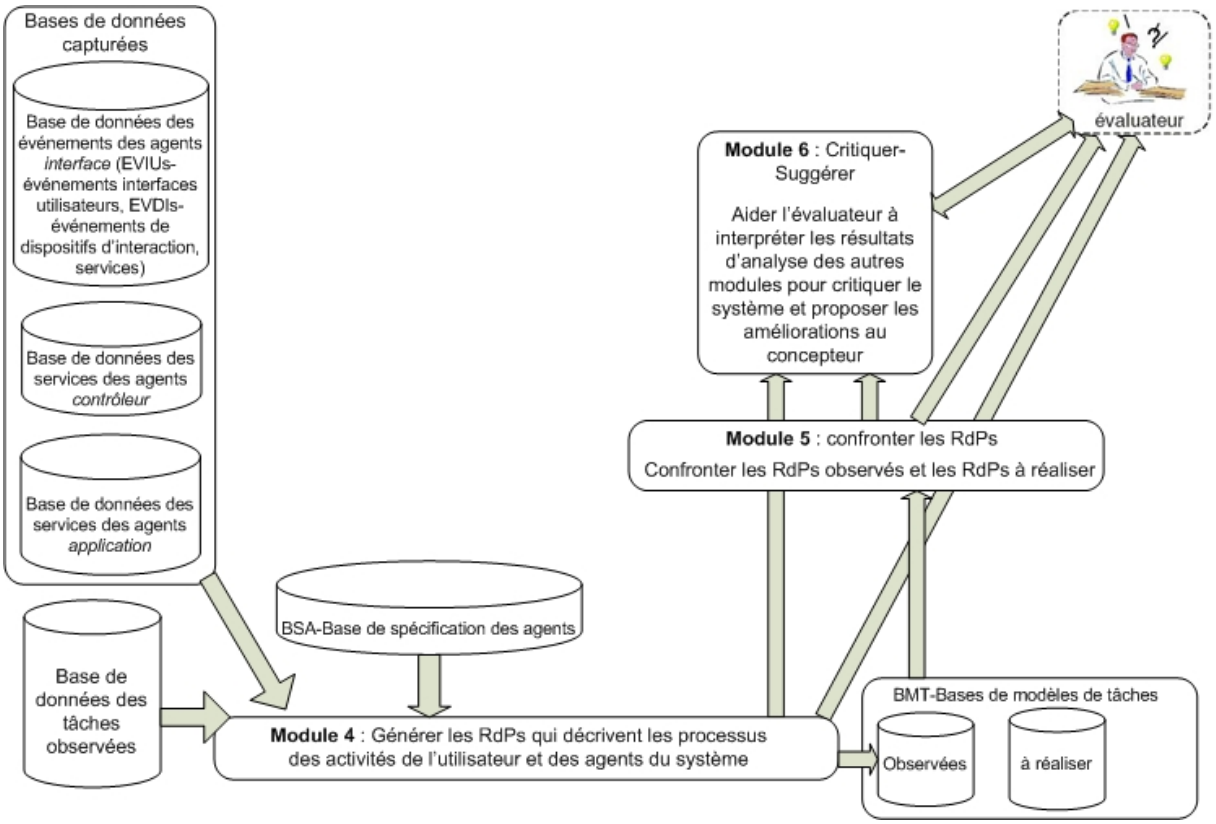

Figure 3.22 : Module 4 et 5 - Générer et confronter les réseaux de Pétri (RdPs)

Ces deux modules permettent à l'évaluateur de générer les RdPs et de les confronter. Comme l'illustre la figure 3.21, le module 4 exploite les événements au niveau intermédiaire (services, EVIUs) capturés et enregistrés dans les bases de données par le module 1, la BSA (Base de Spécification des Agents), et la base de données des tâches observées afin de générer le réseau de Pétri (appelés *RdPs observés* ou *RdPs réels*) pour les tâches. Un *RdP observé* ou *RdP réel* reconstitue le processus des activités réelles de l'utilisateur et des agents du système interactif pour réaliser une tâche quelconque. En effet, chaque RdP généré décrit les actions réelles de l'utilisateur (sous forme des EVIUs) et les actions réelles du système (sous forme des services exécutés par les agents) pour réaliser une tâche. Ces RdP sont stockés dans une base des tâches observées. La base des tâches à réaliser (la base des tâches théorique) créée par le concepteur contient les RdPs qui décrivent les processus des activités qu'il a prévus pour réaliser les tâches. Les RdPs générés sont décrits à l'aide de PNML (Petri Net Markup Language) (Voir Annexe 1).

Ces RdPs générés sont utiles pour l'évaluateur. En effet, ils fournissent à l'évaluateur une vision de toutes les actions effectuées par l'utilisateur et les agents du système en situation

réelle. Donc, l'évaluateur dispose d'une aide pour détecter des problèmes et inconvénients liées aux interactions et au système. L'évaluateur peut effectuer la confrontation entre les RdPs générés (*RdPs observés* ou *RdPs réels*) et les RdPs prévus par le concepteur du système (*RdPs théoriques* ou *RdPs prévus* ou *RdPs à réaliser*) ou entre les RdPs générés relatifs aux différents utilisateurs. Cette confrontation peut être effectuée pour chaque tâche. La génération et la confrontation des RdPs peuvent aider l'évaluateur dans ses activités :

- Détection des échecs du système lors de l'exécution des services des agents. Par exemple, dans le cas d'un système de supervision, l'évolution du processus supervisé doit être mise à jour sur l'interface en temps réel pour que le superviseur puisse connaître l'état courant du processus. Un service de supervision du déplacement des véhicules en temps réel peut échouer dans le déclenchement d'un autre service qui est chargé d'afficher la position courante des véhicules sur l'interface. En conséquence, la position courante des véhicules n'est pas mise à jour sur l'interface et le superviseur ne connaît pas exactement la position courante des véhicules. Bien que les résultats d'exécution des services puissent être fournis par le module 3 sous forme textuelle, grâce aux RdPs générés, l'évaluateur peut constater d'une façon visuelle le processus d'activité du système.
- Pour une tâche qui peut être réalisée selon plusieurs chemins possibles au sens des RdPs décrits par le concepteur, le RdP généré pour un utilisateur fait connaître à l'évaluateur le chemin que l'utilisateur a choisi en réalité. L'évaluateur peut déterminer si le chemin choisi par l'utilisateur est optimal ou non.
- Détection des mauvaises actions et des manipulations inutiles de l'utilisateur qui sont utiles pour l'évaluteur afin d'identifier les problèmes et les inconvénients possibles de l'interface, d'évaluer la performance des utilisateurs, de superviser l'évolution de la performance d'un utilisateur et comparer la performance des différents utilisateurs.
- Grâce à l'étude des actions de l'utilisateur, des chemins choisis afin de réaliser les tâches, etc., l'évaluateur peut étudier certaines propriétés de l'utilisateur comme des préférences, ses habitudes d'utilisation du système, les erreurs que l'utilisateur commet souvent lorsqu'il utilise le système, etc. Les données sur les comportements de l'utilisateur peuvent être utilisées pour enrichir les connaissances sur l'utilisateur. Ces connaissances sont très utiles au concepteur du système pour l'aider à concevoir un système plus familier pour l'utilisateur. Ces connaissances peuvent aussi contribuer à la personnalisation de l'information si le concepteur veut concevoir un système interactif personnalisé (SIP) [Anli, 2006 ; Petit-Rozé, 2003].

Nous présentons ici un exemple issu d'une expérimentation réalisée au sein du laboratoire LAMIH avec dix sujets. Pendant cette expérimentation qui sera détaillée dans le chapitre suivant, l'environnement proposé a été appliqué pour évaluer le système SAI – un système de supervision du transport public terrestre Valenciennois (Tram, Bus). Les sujets ont dû travailler avec le SAI pour réaliser neuf tâches d'envoi des messages aux stations/véhicules et réaliser quelques traitements des perturbations des véhicules. Cette expérimentation permet à l'évaluateur de critiquer le SAI et de proposer au concepteur des améliorations.

Les RdPs générés pour les utilisateurs sont souvent très complexes avec plusieurs manipulations inutiles et des mauvaises actions. Nous choisissons maintenant le RdP le plus simple dans un but d'illustration. Cet exemple montre le *RdP de la tâche à réaliser* prévue par le concepteur et le *RdP observé* du sujet 2.

La figure 3.22 montre la capture d'écran de la confrontation entre le RdP du sujet 2 généré dans notre expérimentation pour réaliser la tâche "*Envoyer un message aux voyageurs en station*" et le RdP théorique prévu par le concepteur pour la réaliser [16].

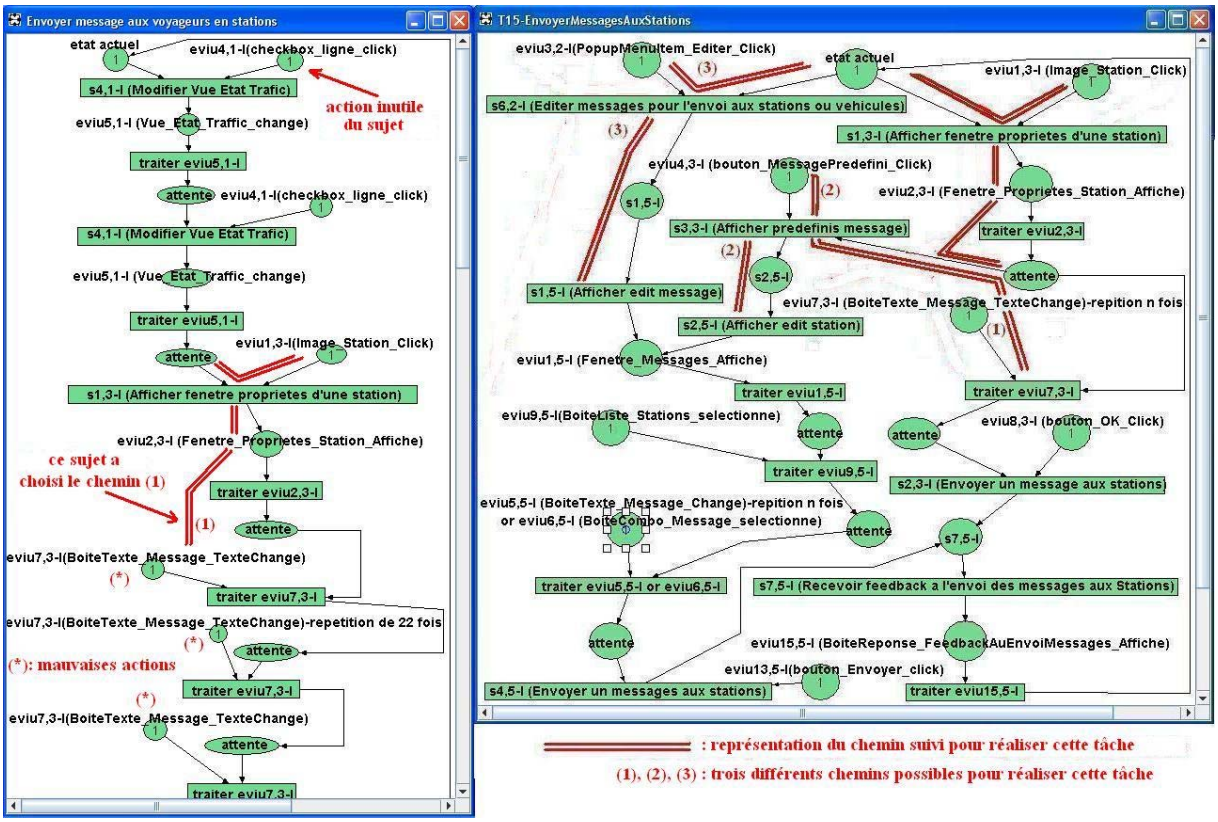

Figure 3.23 : Confrontation entre le RdP du sujet 2 pour réaliser la tâche "Envoyer un message aux voyageurs en station" et le RdP théorique prévu par le concepteur pour cette tâche.

A travers l'examen des RdPs illustrés ci-dessus, l'évaluateur peut percevoir les faits suivants :

- Il y a trois chemins possibles pour réaliser la tâche "*Envoyer un message aux voyageurs en station*". On peut distinguer des transitions et états communs par lesquels deux ou les trois chemins doivent passer, mais il y a aussi des transitions et états qui sont particuliers à un chemin. Le sujet 2 a choisi le chemin (1) dans son *RdP observé* illustré en partie gauche parmi ces trois chemins possibles. Dans le système interactif SAI, c'est un chemin optimal pour réaliser cette tâche si l'utilisateur veut envoyer le message à une seule station.
- Lors de la réalisation de cette tâche, le sujet 2 a effectué une action inutile (précisée en figure 3.21). Cette action est représentée par un EVIU identifié par *eviu4,1-I (checkbox_ligne_click)*. Il faut que le sujet 2 effectue une seule fois cette action pour accéder à la ligne de la station à laquelle il veut envoyer un message ; mais en réalité, le RdP généré de ce sujet (cf. partie gauche) montre que ce sujet a effectué deux fois cette action. Donc, sa première action est inutile.
- Lors de la réalisation de cette tâche, le sujet 2 a aussi effectué des actions considérées comme mauvaises. En effet, l'évaluateur peut les constater par la répétition des EVIUs

[16] Sur cette figure, eviuM,N-I est l'événement interface utilisateur M de l'agent *interface* N ; sM,N-I est le service M de l'agent *interface* N. En partie droite, les deux droites parallèles associées aux indices (1), (2), (3) du RdP théorique représentent trois différents chemins possibles que le concepteur a prévu pour réaliser cette tâche. En partie gauche, les symboles (*) du RdP du sujet 2 représentent les actions considérées comme mauvaises de ce sujet.

BoiteTexteMessageTexteChangé (marqué par le symbole (*) en figure 3.21). Les saisies du texte de ce sujet avec le clavier n'ont pas été continues mais discrètes. En effet, il a saisi quelques caractères et puis, il s'est arrêté ; ensuite, il a continué à saisir, et il s'est arrêté, etc. Le temps pris pour réaliser les tâches peut être plus long à cause de ces interruptions.

En réalité, l'évaluateur peut superviser le travail d'un utilisateur pendant une période qui peut être longue (heures, journées) en utilisant les RdPs générés ; il peut étudier si cet utilisateur choisit souvent les chemins optimaux pour réaliser les tâches et s'il a des habitudes en termes de mauvaises manipulations ou actions inutiles.

Nous pouvons distinguer les types d'erreur qu'on peut détecter à l'aide des RdPs en particulier et de l'environnement EISEval en général. Ces erreurs sont liées à la réalisation d'une tâche quelconque :

- Erreurs provenant du système : ces erreur sont liées à l'échec de l'exécution des services des agents pendant la réalisation d'une tâche quelconque.
- Erreurs provenant de l'utilisateur. On peut distinguer quelques sous-types:
  - Manipulations redondantes ou inutiles : elles ne causent aucune dommage mais elles ne sont pas nécessaires pour réaliser une tâche quelconque. Dans ce type d'erreur, il faut faire attention à la répétition d'une action inutile.
  - Actions ou navigations erronées : un chemin incorrect a été choisi pour réaliser une tâche. Si ce chemin continue, la tâche ne peut pas accomplie. Un chemin incorrect est composé des actions erronées effectuées par l'utilisateur.
  - Navigations non optimales : Un chemin non-optimal a été choisi pour réaliser une tâche quelconque
  - Mauvaises actions, éventuellement liées à de mauvaises habitudes : les actions sont nécessaires pour réaliser une tâche mais l'utilisateur les a effectuées d'une manière inadéquate (par exemple, re-saisir le même texte au lieu de l'utilisation du « copier/coller », re-sairir un texte disponible et sélectionnable au lieu de le sélectionner, la saisie du texte est discrète au lieu d'être continue, etc.).

La génération des RdPs, la confrontation entre eux et les résultats d'analyse issus du module 3 peuvent être interprétés grâce au module 6 qui permet de les associer à des critères déterminés afin d'aider l'évaluateur à critiquer le système et à suggérer des améliorations au concepteur.

3.5.5. Module 6 : Critiquer le système et proposer au concepteur des suggestions d'amélioration

La figure 3.23 illustre le module 6. Les modules 3, 4 et 5 facilitent l'analyse des données capturées. L'évaluateur doit interpréter ces résultats d'analyse (RdPs générés des modules 4-5, autres résultats d'analyse issus du module 3) pour critiquer le système et pour suggérer des améliorations au concepteur. Le module 6 a pour objectif d'aider l'évaluateur à ce sujet. Pour l'instant, ce module fournit seulement à l'évaluateur des indications nécessaires pour interpréter les résultats d'analyse. Cette interprétation va permettre d'évaluer trois aspects différents déjà abordés du système (interface utilisateur, quelques propriétés non fonctionnelles du système, quelques propriétés relatives à l'utilisateur).

Le module 6 fournit à l'évaluateur une liste de critères qui sont associés aux résultats d'analyse des autres modules (3, 4, 5) pour l'aider à critiquer le système et à proposer des améliorations au concepteur. Ces critères d'évaluation du module 6 doivent respecter quelques principes :

- Les critères d'évaluation doivent être évalués complètement ou partiellement en interprétant les résultats d'analyse provenant des autres modules. Plusieurs règles ergonomiques ne peuvent pas être acceptées et ajoutées au module 6 parce qu'on ne peut pas les évaluer en utilisant les résultats d'analyse issus des autres modules. Par exemple, les règles comme *Utiliser le soulignement avec parcimonie* [Galiz,1988] ou *Les phrases de l'affichage textuel doivent être distinctes et lisibles* [Marcus, 1991] ne peuvent pas être acceptées. Ces règles sont souvent évaluées à l'aide d'outils utilisant les règles ou par d'autres méthodes, par exemple, un questionnaire.
- La liste des critères d'évaluation du module 6 doit être ouverte et modifiable ; il doit être possible d'en ajouter de nouveaux ou de modifier les critères existants.
- Les sources des critères d'évaluation du module 6 ne sont pas limitées. Le module 6 peut accepter les critères d'évaluation qui viennent des sources disponibles ou qui sont déterminés par l'évaluateur lui-même. En effet, un critère d'évaluation du module 6 peut être un critère ergonomique ([Bastien et Scapin, 1993]) ; ou une règle ergonomique générale et indépendante des domaines ([Scapin, 1986 ; Smith et Mosier, 1986 ; Vanderdonckt, 1994], etc.) ; ou un guide de style (style guide) propre à un système, un environnement ou une organisation particulière, par exemple, Apple[17] ; ou un attribut de qualité ([Lee et Hwang, 2004]), etc.
- Les critères d'évaluation peuvent être génériques ou spécifiques au système interactif à évaluer. Les critères d'évaluation spécifiques concernent souvent le métier et les concepts du domaine du système à évaluer.

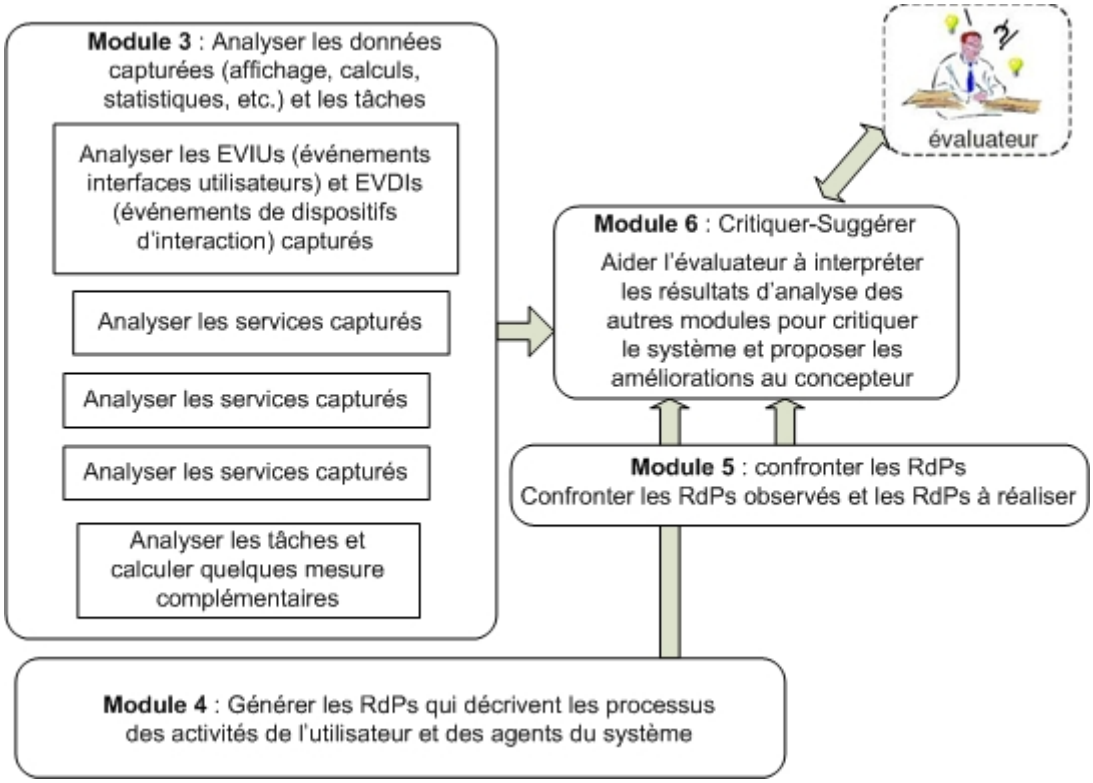

Figure 3.24 : Module 6 - Aider l'évaluateur à critiquer le système et à suggérer des améliorations

Pour l'instant, le module 6 fournit une liste de critères d'évaluation qui contient :

[17] http://developer.apple.com/documentation/UserExperience/Conceptual/AppleHIGuidelines/XHIGIntro/XHIGIntro.html

- Quelques critères ergonomiques provenant de [Bastien et Scapin, 1993]. D'après [Bastien et al.,1999], les huit critères et treize sous-critères de [Bastien et Scapin, 1993] s'avèrent plus efficaces que les normes de dialogue ISO 9241-10 pour détecter des problèmes ergonomiques sur une IHM. Pour notre part, nous considérons les critères de Bastien et de Scapin comme une source très utile pour l'évaluation d'interfaces utilisateur.
- Quelques attributs de qualité de système multi-agent provenant de [Lee et Hwang, 2004]. Ces attributs visent les propriétés non fonctionnelles du système comme le temps de réponse, la fiabilité, la complexité.
- Quelques critères d'évaluation spécifiques au système SAI. Ces critères ont été ajoutés au module 6 quand l'environnement a été utilisé pour évaluer le SAI pendant une expérimentation que nous allons présenter dans le chapitre suivant.

Un critère d'évaluation du module 6 est structué en quatre parties comme l'illustre la figure 3.24. Nous les présentons maintenant.

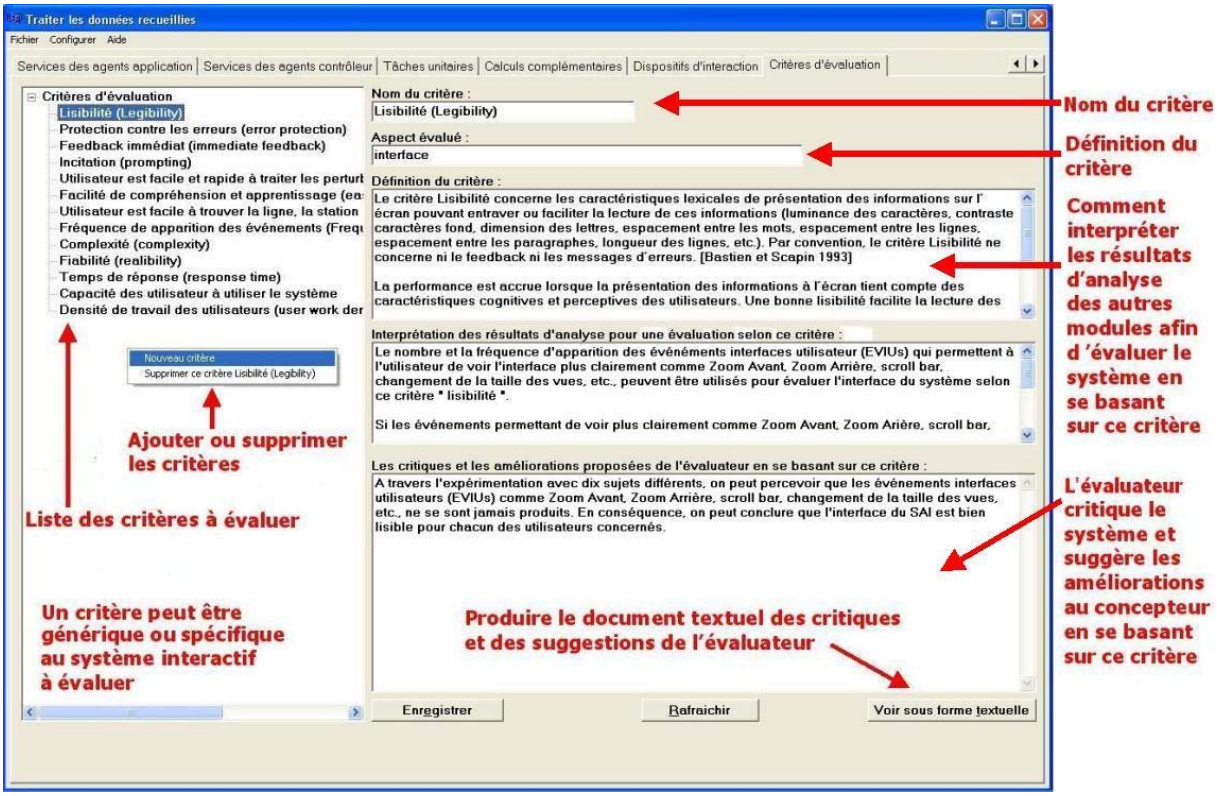

Figure 3.25 : Capture d'écran du module 6

Un critère est composé des 4 parties :

- *Nom du critère* (par exemple, *lisibilité* sur la figure 3.24 ci-dessus)
- *Définition du critère*. Cette partie fournit à l'évaluateur une définition de ce critère.
- *Interprétation des résultats d'analyse pour une évaluation selon ce critère*. Cette partie fournit à l'évaluateur une indication concrète sur la manière d'exploiter des résultats d'analyse pour évaluer le système selon ce critère. Par exemple : *le nombre et la fréquence d'apparition des événéments interfaces utilisateur (EVIUs) qui permettent à l'utilisateur de voir l'interface plus clairement comme Zoom Avant, Zoom Arrière, scroll bar, changement de la taille des vues, etc., peuvent être utilisés pour évaluer l'interface du système selon ce critère « lisibilité »*.

- *Critiques et améliorations proposées par l'évaluateur en se basant sur ce critère.* Cette partie permet à l'évaluateur de saisir ses critiques sur le système à évaluer et ses propositions d'amélioration en se basant sur ce critère. Il est possible de générer et de sauvegarder toutes les évaluations selon un document textuel. Par exemple, voici les critiques de l'évaluateur relatives à la *lisibilité* de l'interface du système SAI : *A travers l'expérimentation avec dix sujets différents, on peut percevoir que les événements interfaces utilisateurs (EVIUs) comme Zoom Avant, Zoom Arrière, scroll bar, changement de la taille des vues, etc., ne se sont jamais produits. En conséquence, on peut conclure que l'interface du SAI est bien lisible pour chacun des utilisateurs concernés*.

Le module 6 est comme un indicateur pour l'évaluateur. En effet, l'évaluateur peut se baser sur les critères d'évaluation du module 6 (qui sont associés aux résultats d'analyse des autres modules) pour critiquer le système et pour proposer les améliorations nécessaires au concepteur. Les détails de tous ces critères d'évaluation, les critiques et les propositions d'amélioration de l'évaluateur pour le système SAI en se basant sur ces critères d'évaluation et les résultats d'analyse des données de l'expérimentation seront présentés dans le chapitre suivant.[18]

Notons ainsi que l'amélioration du module 6 fait partie des perspectives que nous allons présenter dans le dernier chapitre.

3.5.6. Module 7 : Configurer l'environnement pour évaluer différents systèmes interactifs

Le module 7 permet à l'évaluateur de configurer l'environnement afin de pouvoir évaluer différents systèmes interactifs. L'évaluateur doit saisir un ensemble d'informations relatives au système interactif à évaluer ; par exemple, des informations générales, des informations concernant les tâches qu'on peut réaliser en utilisant ce système (*tâches prévues* ou *tâches théoriques* ou *tâches à réaliser*), la BSA (agents, services, EVIUs) (Voir Annexe 2). La figure 3.25 illustre le fonctionnement de ce module. Les deux figures 3.26 et 3.27 illustrent deux captures d'écran du module 7.

[18] Notons que [Bernonville *et al*., 2006 ; Bernonville, 2008] ont présenté une méthode appelée ErgoPNets qui permet à l'évaluateur d'associer les Réseaux de Pétri (correspondant à la dynamique du système interactif concerné) aux critères ergonomiques de Bastien et Scapin [Bastien and Scapin, 1994] afin de montrer les problèmes d'utilisabilité. Bien qu'il y ait quelques points communs, cette méthode est différente de celle exploitée dans le module 6 : (1) ErgoPNets vise seulement l'identification des problèmes ergonomiques de l'interface utilisateur (qui est seulement un des trois aspects visés par le module 6). En plus, les spécificités de l'architecture à base d'agents des systèmes interactifs ne sont pas pris en compte par ErgoPNets. (2) ErgoPNets ne prend en compte que les Réseaux de Pétri (RdPs) prévus par le concepteur (*RdPs théoriques* ou *RdPs à réaliser*) et aussi ceux améliorés selon les recommandations de l'évaluateur. Elle ne prend pas en compte les RdPs générés de l'utilisateur (*RdPs réels* ou *RdPs observés*). En fait, ces *RdPs réels* (notamment la confrontation entre eux – entre *RdPs réels* et *RdPs théoriques* ou entre *RdPs réels* des différents utilisateurs) peuvent aider l'évaluateur à détecter plus de problèmes que les *RdPs théoriques* parce que ces *RdPs réels* décrivent les actions de l'utilisateur de même que les réactions de l'IHM en situation réelle.

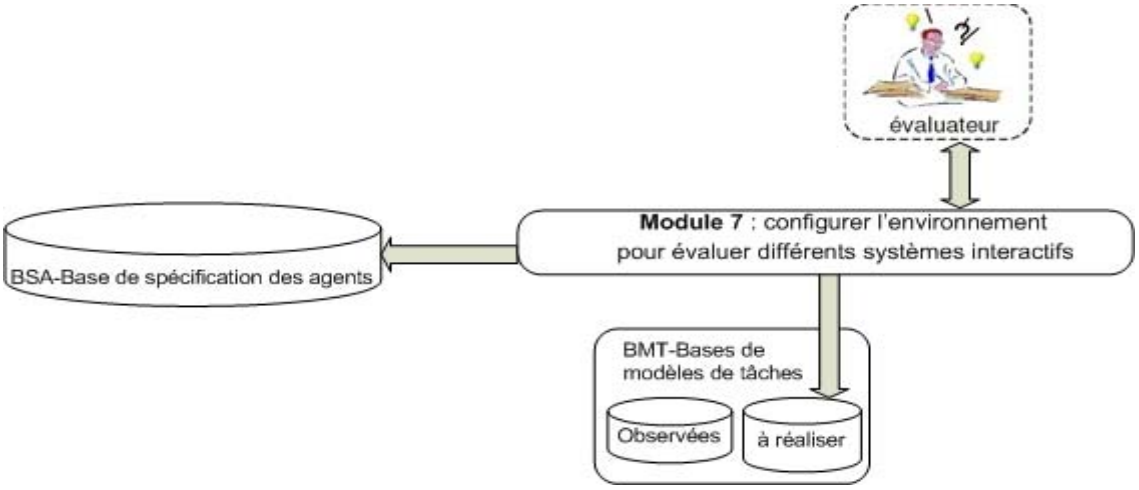

Figure 3.26 : Module 7 - Configurer l'environnement pour évaluer différents systèmes interactifs

La figure 3.26 illustre une capture d'écran du module 7 qui permet à l'évaluateur de saisir des informations générales sur le système interactif à évaluer, par exemple : le nom du système (le SAI-Système d'Aide à l'Information dans sa version 2003-2005 dans ce cas), la description de celui-ci (le SAI est un système de supervision du transport public terrestre Valenciennois, etc., dans ce cas). Cette information a pour objectif d'expliquer le système interactif à évaluer.

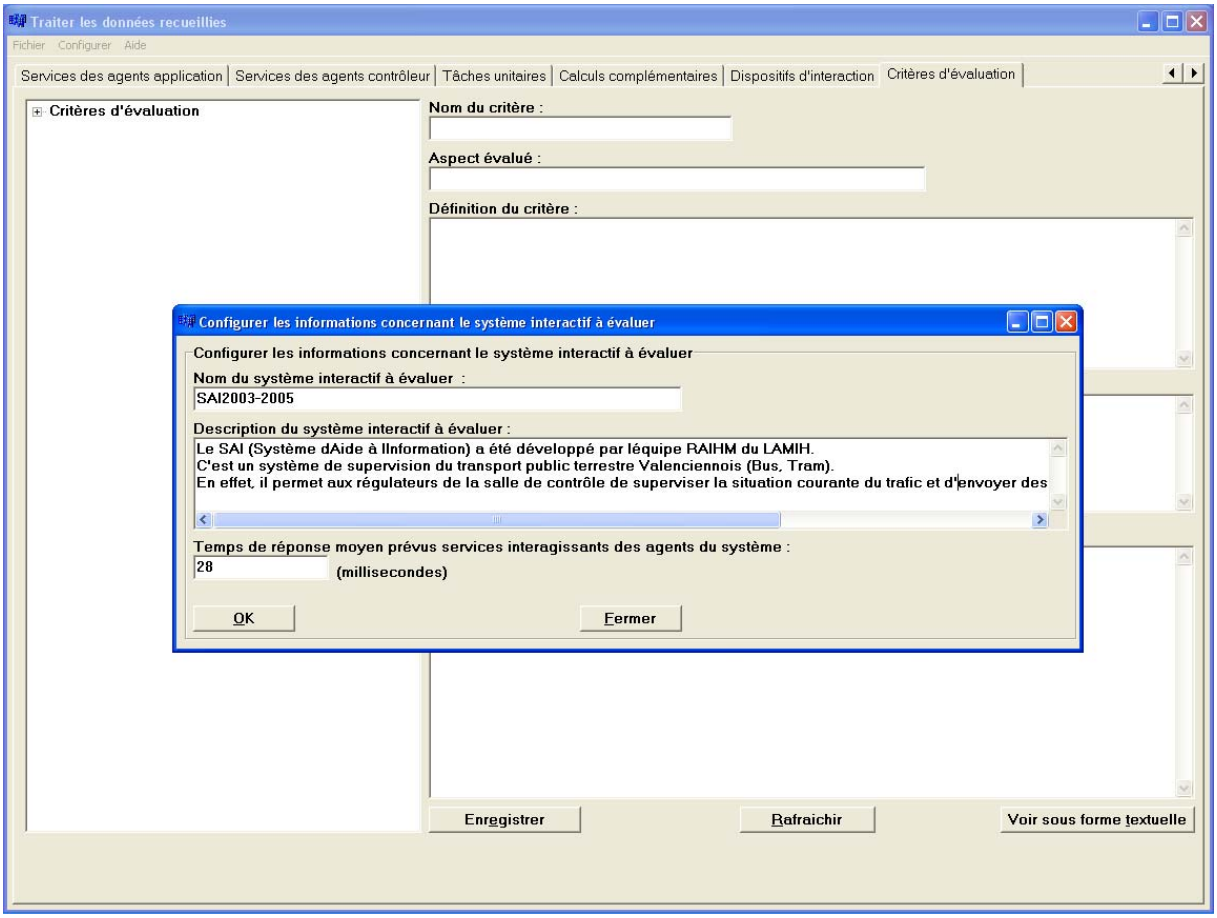

Figure 3.27 : Capture d'écran du module 7 - saisie d'informations générales sur le système interactif à évaluer

La figure 3.27 illustre une autre capture d'écran du module 7 qui permet à l'évaluateur de saisir des informations concernant les tâches qu'on peut réaliser en utilisant le système interactif à évaluer. Ce sont les *tâches prévues* par le concepteur. Comme l'illustre cette figure, il y a deux paramètres importants que nous avons présentés ci-dessus (cf. section du module 3) : (1) « *Le temps moyen acceptable pour réaliser cette tâche* » (2) « *Le temps expert pour réaliser cette tâche* ».

Le module 3 fait la confrontation entre les durées prises par l'utilisateur en réalité pour réaliser une tâche et ces deux durées prévues par le concepteur pour sa réalisation. Ce module 3 permet de compter le nombre de tâches dont le temps pris est plus/moins/aussi long que le temps expert/moyen prévu comme l'illustre la figure 3.20 (cf. section 3.5.3 relative au module 3). Cette confrontation est utile pour l'évaluateur afin de détecter les problèmes du système et d'étudier certaines propriétés de l'utilisateur.

Ce module permet à l'évaluateur de saisir la BSA (Base de Spécification des Agents) du système interactif. Une description de la BSA est fournie en Annexe 2.

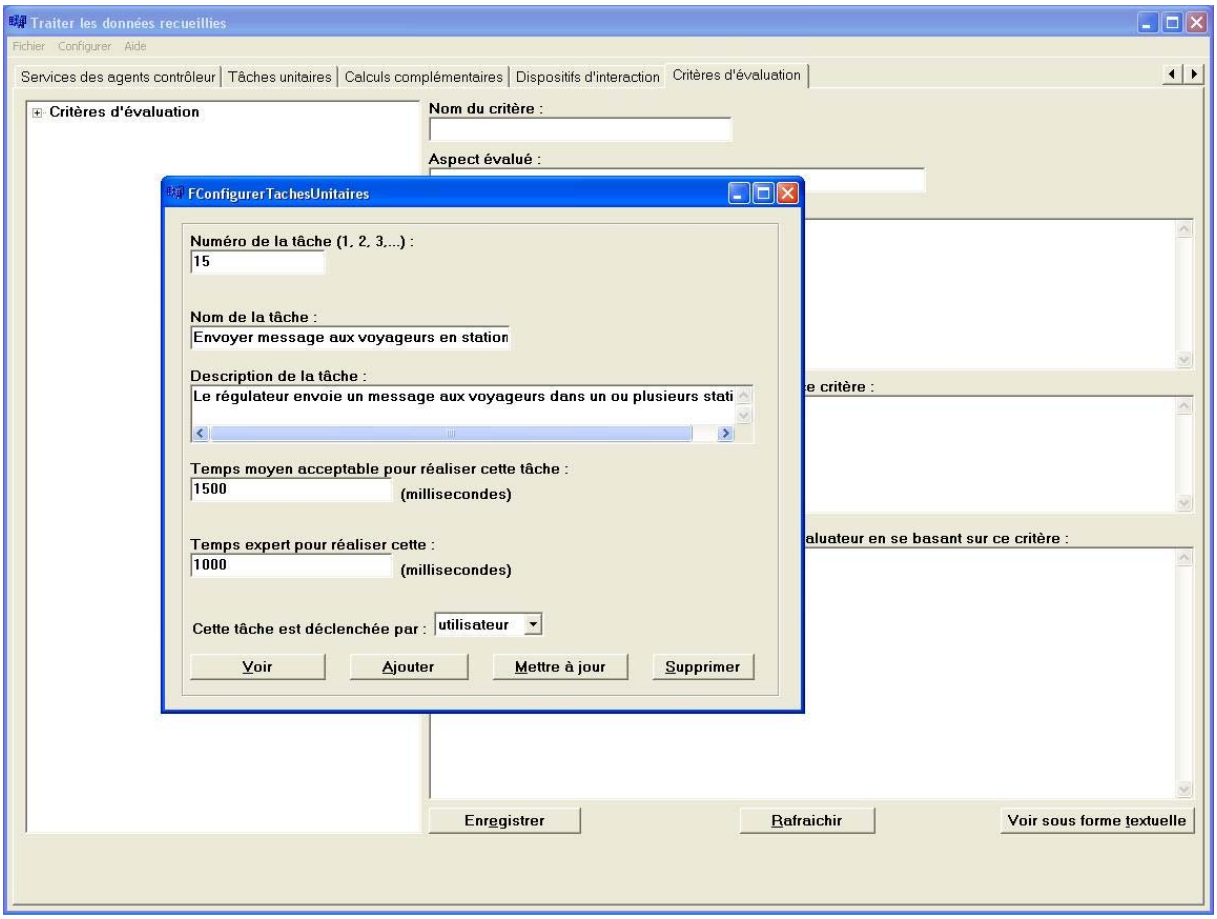

Figure 3.28 : Capture d'écran du module 7 - saisie d'informations concernant les tâches qu'on peut réaliser en utilisant le système interactif à évaluer

4. Conclusion

Dans ce chapitre, nous avons présenté notre environnement d'évaluation EISEval. D'abord, les limitations du mouchard MESIA (proposé dans le cadre des travaux précédents au laboratoire) et aussi des mouchards électroniques traditionnels ont été discutées. Ensuite, nous avons présenté les principes de l'environnement EISEval. Le respect de ces principes permet à cet environnement de prendre en compte les spécificités de l'architecture à base

d'agent des systèmes interactifs, de remédier aux limitations des mouchards électroniques traditionnels et d'aller plus loin qu'eux pour évaluer différents aspects d'un système interactif (interface, quelques propriétés non fonctionnelles du système et certaines propriétés de l'utilisateur). De plus, l'environnement EISEval fournit aussi quelques fonctionnalités originales par rapport aux mouchards traditionnels.

Pour concevoir l'environnement EISEval, nous avons proposé une description formelle de l'architecture à base d'agents des systèmes interactifs qui a été aussi présentée dans ce chapitre. Enfin, les sept modules indépendants qui se composent l'environnement EISEval ont été présentés. Le principe de couplage entre l'environnement EISEval et le système interactif à évaluer a été abordé dans la section relative au module 1. En effet, ce module est chargé de capturer les événements se produisant dans le système interactif à évaluer et de les stocker dans des bases afin que les autres modules puissent les récupérer et les analyser ultérieurement. Le module 2 permet de déterminer les tâches que l'utilisateur a réalisées lorsqu'il a utilisé le système interactif. Quelques analyses des événements capturés (classifications, statistiques, calculs de quelques mesures complémentaires) sont effectuées par le module 3. Les résultats d'analyse sont affichés sous des formes compréhensibles pour l'utilisateur. Les modules 4-5 permettent de générer les Réseaux de Pétri (RdPs) et de les confronter. Un RdP généré reconstitue le processus des activités réelles de l'utilisateur (sous forme des EVIUs) et des agents du système interactif (sous forme des services qu'ils exécutent) pour réaliser une tâche quelconque. On peut comparer les RdPs générés avec les RdPs prévus par le concepteur ou comparer les RdPs générés de différents utilisateurs. Ces RdPs et les résultats d'analyse issus du module 3 peuvent être interprétés grâce au module 6 qui permet de les associer à un ensemble de critères déterminés afin d'aider l'évaluateur à critiquer le système et à suggérer au concepteur des améliorations. Le module 6 est ouvert et modifiable, donc l'évaluateur peut ajouter de nouveaux critères ou modifier ou supprimer les critères existants. Le module 7 permet à l'évaluateur de configurer l'environnement afin de pouvoir évaluer différents systèmes interactifs.

Pour illustrer les activités de ces modules, nous avons utilisé les données d'une expérimentation mettant en oeuvre l'environnement EISEval pour évaluer le SAI. Cette expérimentation permet de mettre en lumière les avantages et problèmes du SAI et de proposer les améliorations nécessaires. Cette expérimentation permet aussi de tester l'environnement EISEval proposé. Nous allons la présenter dans le chapitre suivant.

Chapitre 4 : Utilisation de l'environnement proposé pour l'évaluation d'un système interactif dans le domaine des transports

Sommaire

1. Introduction

Une expérimentation a été réalisée au sein de notre laboratoire LAMIH afin d'évaluer l'utilisation du système SAI (Système d'Aide à l'Information voyageurs). Pendant cette expérimentation, l'environnement d'évaluation EISEval que nous avons proposé et développé a été appliqué. Les sujets, participants à l'expérimentation, ont dû travailler avec le SAI pour réaliser neuf tâches d'envoi de messages aux stations/véhicules et de traitement de perturbations relatives aux véhicules. Cette expérimentation a permis à l'évaluateur de critiquer le SAI et de proposer des améliorations à son concepteur.

Avant de présenter cette expérimentation, nous introduisons ici le projet de recherche SART (Système d'Aide à la Régulation du Trafic) qui s'inscrit dans le cadre du programme : Technologies Avancées pour les Transports (TAT) du Groupement Régional Nord-Pas de Calais pour la Recherche dans les Transports (G.R.R.T). Ce projet a pour partenaire industriel, TRANSVILLES exploitant du réseau de transport public terrestre Valenciennois (Tramway/Bus). Plusieurs laboratoires de recherche participent à celui-ci : ESTAS/INRETS, LAGIS/EC-Lille-USTL, LAMIH/UVHC (équipes « Raisonnement Automatique et Interaction Homme-Machine » et « Systèmes de Production »). Il est soutenu par la région Nord-Pas-de-Calais et le FEDER (Fonds Européen de Développement Régional).

2. Contexte de l'exploitation de l'environnement d'évaluation proposé

2.1. Projet SART

Le projet SART a pour objectif de concevoir et réaliser un système qui aide les régulateurs en salle de contrôle à mieux accomplir leurs tâches en modes normal et dégradés de fonctionnement du réseau de transport. Il doit permettre de minimiser le temps d'attente des voyageurs, en mode dégradé, dans les pôles d'échange et permettre aussi de leur assurer, dans la mesure du possible, la continuité des déplacements dans les réseaux multimodaux. Il s'agit donc d'améliorer la qualité du service rendu aux voyageurs et de les maintenir informés [Hayat *et al*., 2001].

D'après [Fayech, 2003], la régulation est le processus d'adéquation en temps réel des tableaux de marche des différents modes de transport aux conditions réelles d'exploitation. La régulation du trafic est une régulation globale du réseau de transport collectif. Le système d'aide à la régulation du réseau de transport du projet SART est constitué de trois sous-systèmes comme l'illustre la figure 4.1 :

- **SAE** (Système d'Aide à l'Exploitation) : ce système permet le suivi en temps réel de l'exploitation du réseau de transport urbain ; il ne cesse d'évoluer grâce notamment aux progrès technologiques dans les domaines des communications et du traitement de l'information. Le SAE peut traiter des quantités très importantes d'informations, mais, malgré son aide très précieuse, les régulateurs n'arrivent parfois pas à prendre en compte le grand nombre d'informations véhiculées par ce système dans la prise de décision, en cas de perturbations [SART, 2007]. Dans le cadre de l'expérimentation que nous allons présenter dans la partie suivante, le SAE consiste en un simulateur de la circulation du réseau de transport public terrestre développé avec Quest (Queue Event Simulation Tool), un outil de simulation à événements discrets du DELMIA (http://www.delmia.com). Ce simulateur détermine automatiquement, en temps réel, l'emplacement des véhicules et les différentes perturbations sur le réseau.

- **SAD** (Système d'Aide à la Décision) : ce système est une solution logicielle utilisée pour aider les régulateurs humains dans leurs prises de décision et la résolution de problèmes complexes. Quand le SAD reçoit une perturbation du SAE, il est chargé de l'analyser et de fournir aux régulateurs des solutions de régulation possibles. Les régulateurs peuvent choisir la solution leur semblant la plus adéquate (selon leur expérience), ou ils peuvent aussi ne choisir aucune solution.
- **SAI** (Système d'Aide à l'Information) : c'est un système interactif conçu pour aider le régulateur à superviser le réseau de transport public terrestre. Il présente les informations sur l'état courant du trafic aux régulateurs et leur permet d'envoyer des informations ou commandes nécessaires aux conducteurs des véhicules, de même qu'aux voyageurs dans les stations et dans les véhicules. Pendant l'expérimentation, l'environnement d'évaluation a été utilisé pour évaluer ce système. Nous présentons maintenant ce système avant de détailler cette expérimentation ainsi que ses résultats.

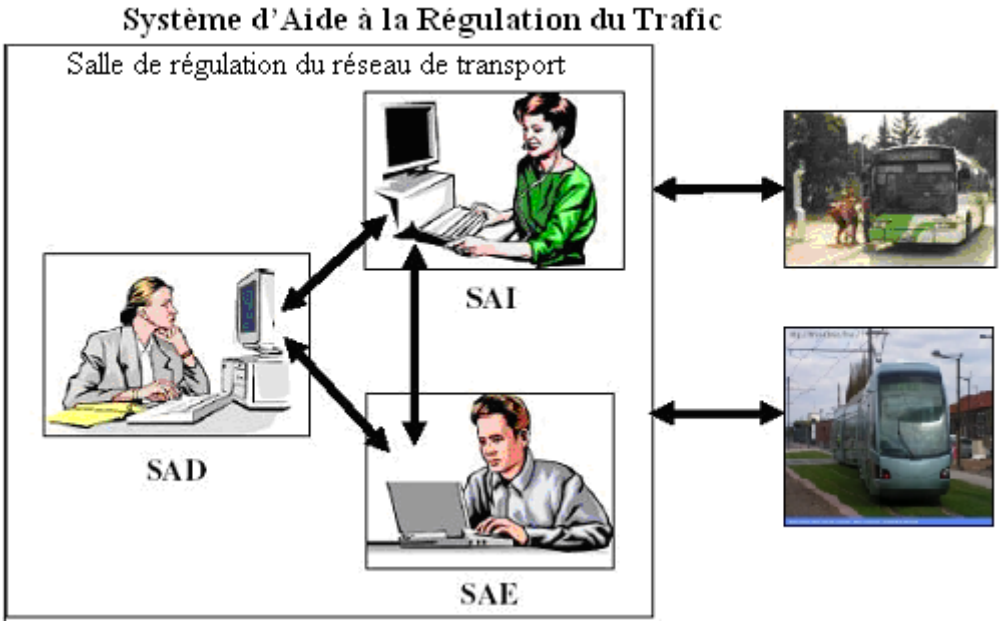

Figure 4.1 : Trois sous-systèmes du Système d'Aide à la Régulation du Trafic (SART)

2.2. Le système interactif à base d'agents SAI

Le système interactif à base d'agents évalué, appelé SAI est un prototype destiné aux régulateurs en salle de contrôle. Ce système SAI est composé de six agents *interface* (*Etat_Trafic*, *Etat_Ligne*, *Station*, *Véhicule*, *Message*, *Vue_Globale*). Nous présentons la vue d'ensemble de ce système ; les lecteurs intéressés peuvent consulter [Ezzedine *et al*., 2003 ; Trabelsi *et al*., 2004 ; Trabelsi, 2006] pour plus de détails.

a) Agent *Interface Etat_Trafic*

Cet agent permet aux régulateurs de connaître l'état courant de la circulation des véhicules (retard, avance) d'une façon visuelle et synthétique. La surveillance est réalisée en temps réel. La figure 4.2 illustre cet agent.

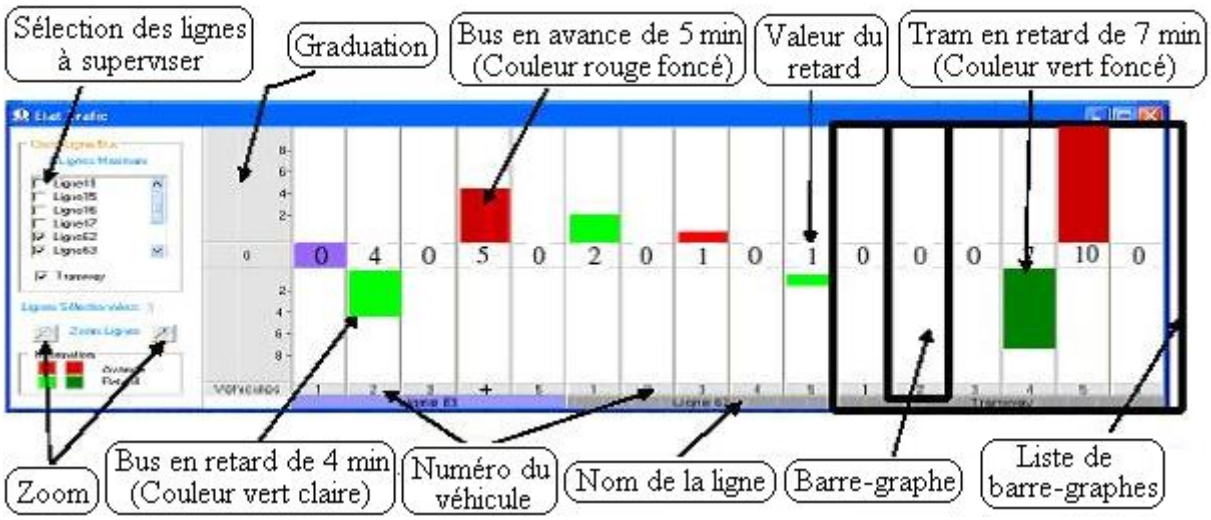

Figure 4.2 : Agent *Interface Etat_Trafic* du SAI [Trabelsi, 2006]

b) Agent *Interface Etat_Ligne*

La figure 4.3 illustre cet agent.

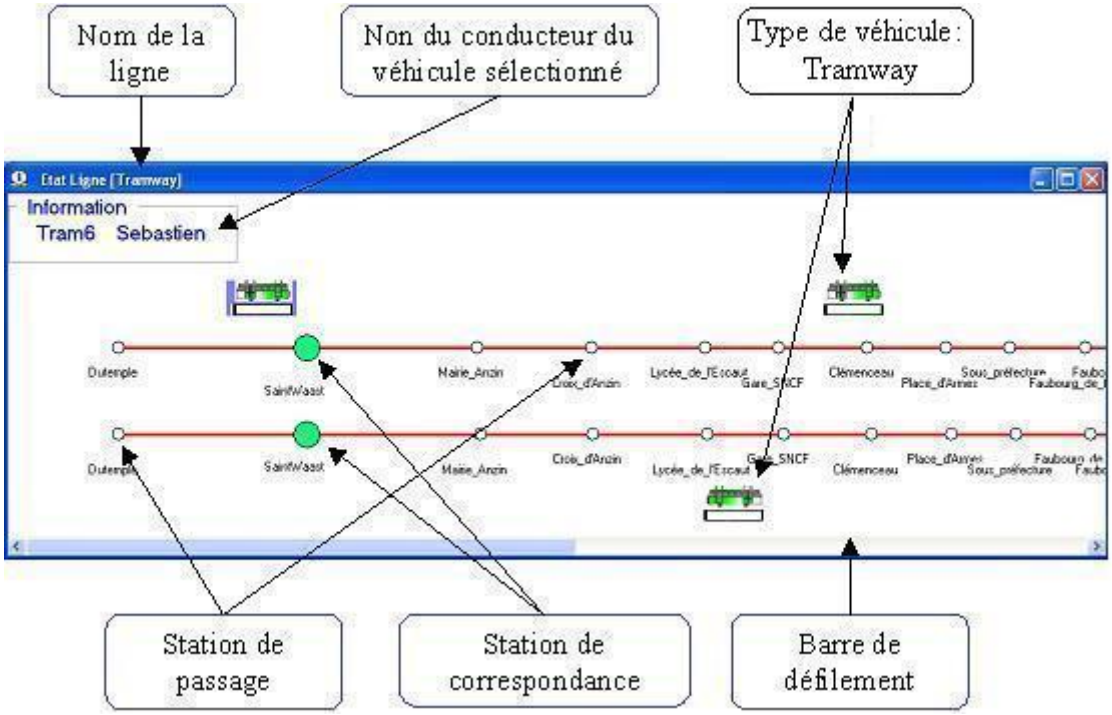

Figure 4.3 : Agent *Interface Etat_Ligne* du SAI [Trabelsi, 2006]

Cet agent affiche une ligne de transport quelconque. Il permet au régulateur de chosir une ligne à superviser. Cet agent est composé de plusieurs éléments comme des stations ou des véhicules de la ligne affichée. Le régulateur peut ouvrir n'importe quelle ligne en cliquant sur la représentation de cette ligne sur l'agent *interface Etat_Trafic*.

c) Agent *Interface Station*

Cet agent affiche les propriétés détaillées d'une station quelconque et il permet aux régulateurs d'envoyer des messages aux voyageurs en attente dans cette station. Le régulateur

peut accéder à cet agent en cliquant sur la représentation de cette station dans l'agent *interface Etat_Ligne* associé. La figure 4.4 illustre cet agent.

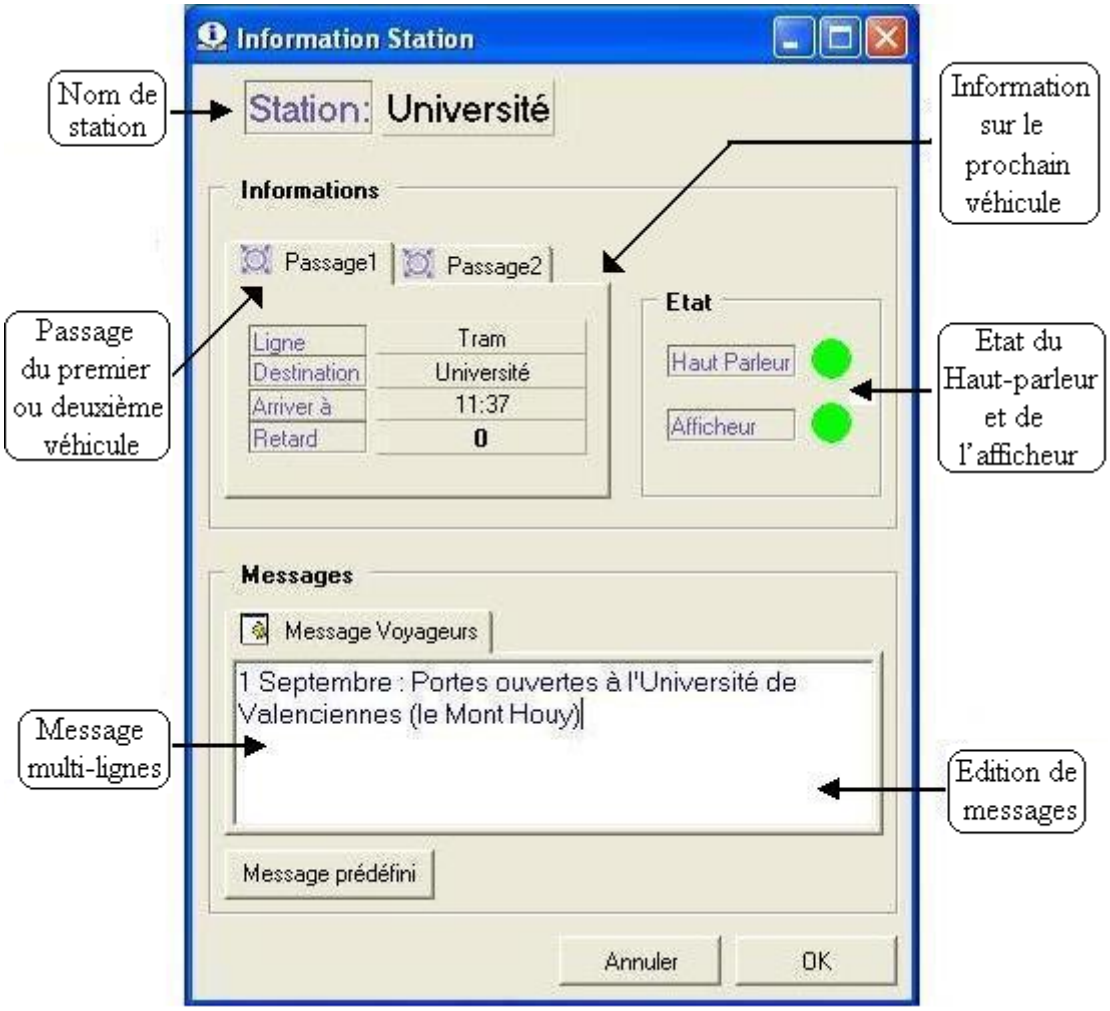

Figure 4.4 : Agent *Interface Station* du SAI [Trabelsi, 2006]

d) Agent *Interface Véhicule*

Similairement, cet agent affiche les propriétés détaillées d'un véhicule quelconque ; il permet aux régulateurs d'envoyer un message aux voyageurs à l'intérieur de ce véhicule ou à son conducteur. Le régulateur peut accéder à cet agent en cliquant sur la représentation de ce véhicule dans l'agent *interface Etat_Ligne* associé. La figure 4.5 illustre cet agent.

e) Agent *Interface Vue_Globale*

Pour l'instant, cet agent fournit au régulateur une vision globale du trafic dans le réseau comme l'illustre la figure 4.6. Dans sa version actuelle, cet agent ne possède pas encore de fonctions interactives.

f) Agent *Interface Message*

Cet agent fournit aux régulateurs une vue synthétique des messages à envoyer aux véhicules ou aux stations. Il permet aux régulateurs de choisir un ou plusieurs véhicules ou stations d'une ou plusieurs lignes différentes pour leur envoyer un message. Cet agent fournit aussi une liste des messages disponibles ; le régulateur peut y accéder en utilisant un menu pop-up sur l'agent *interface*. Cette figure 4.7 illustre cet agent.

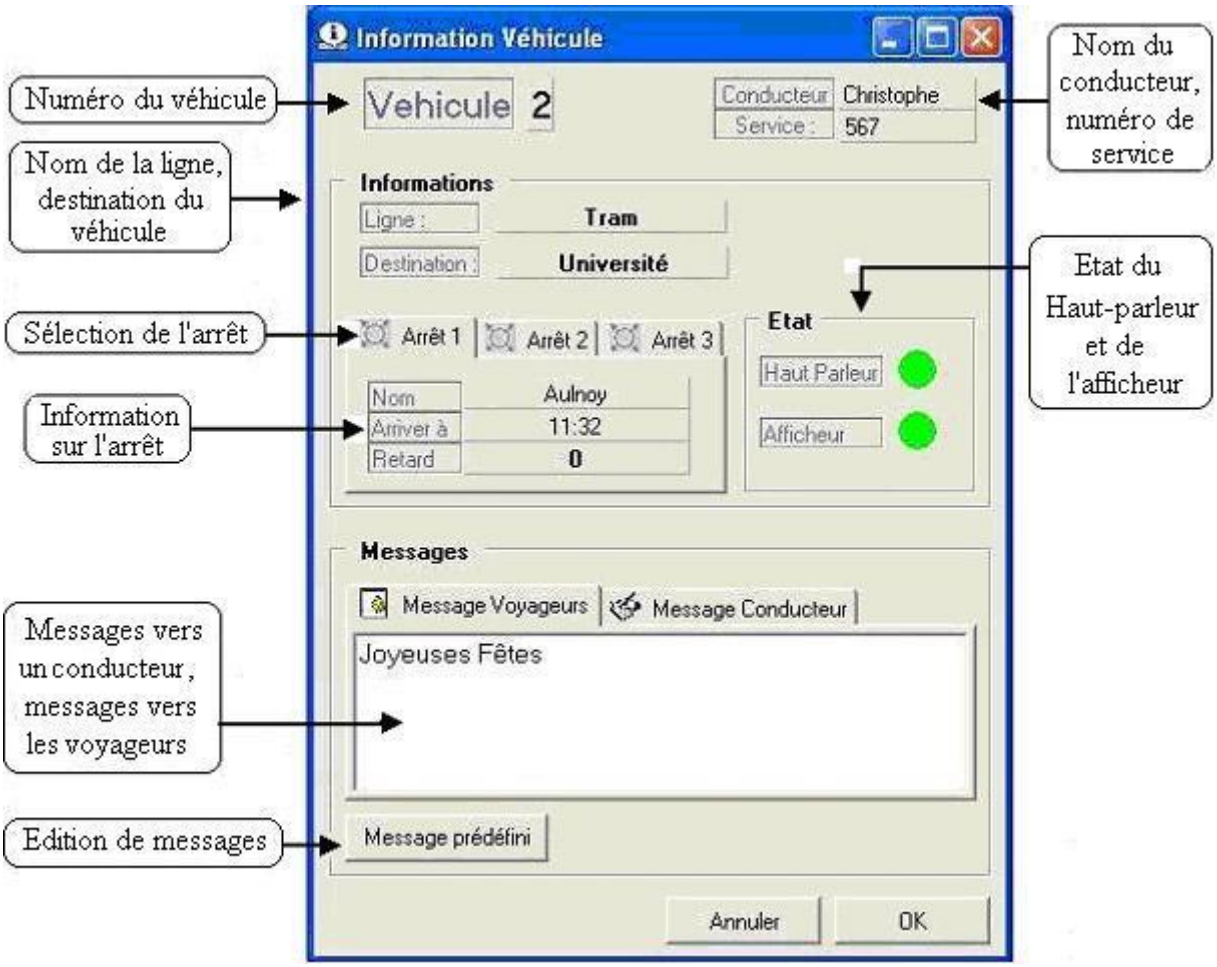

Figure 4.5 : Agent *Interface Véhicule* du SAI [Trabelsi, 2006]

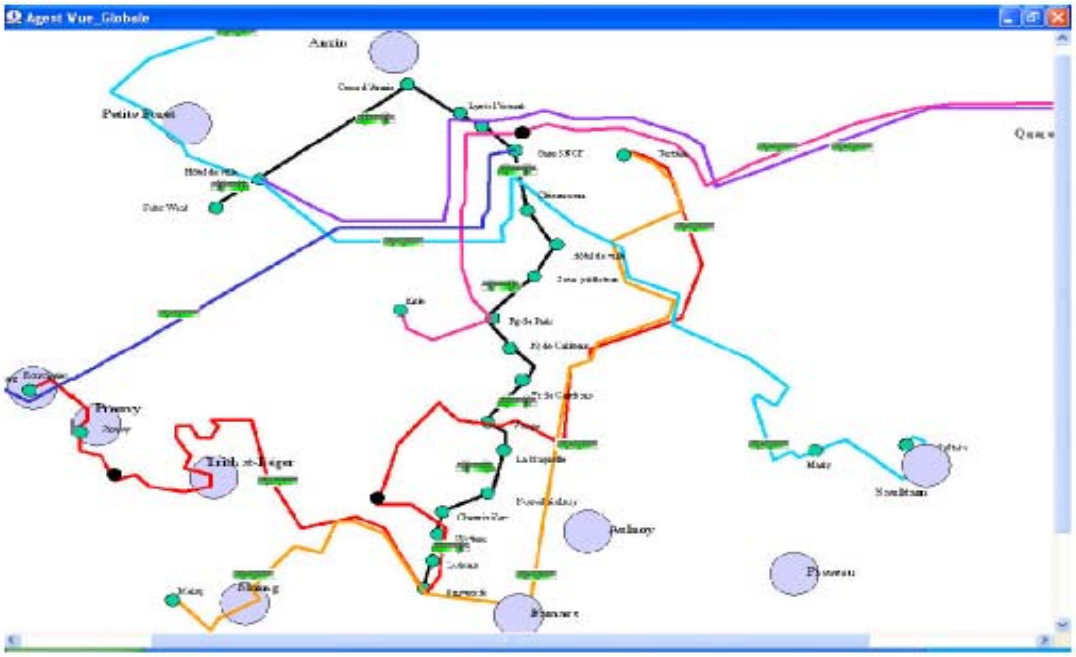

Figure 4.6 : Agent *Interface Vue_Globale* du SAI [Trabelsi, 2006]

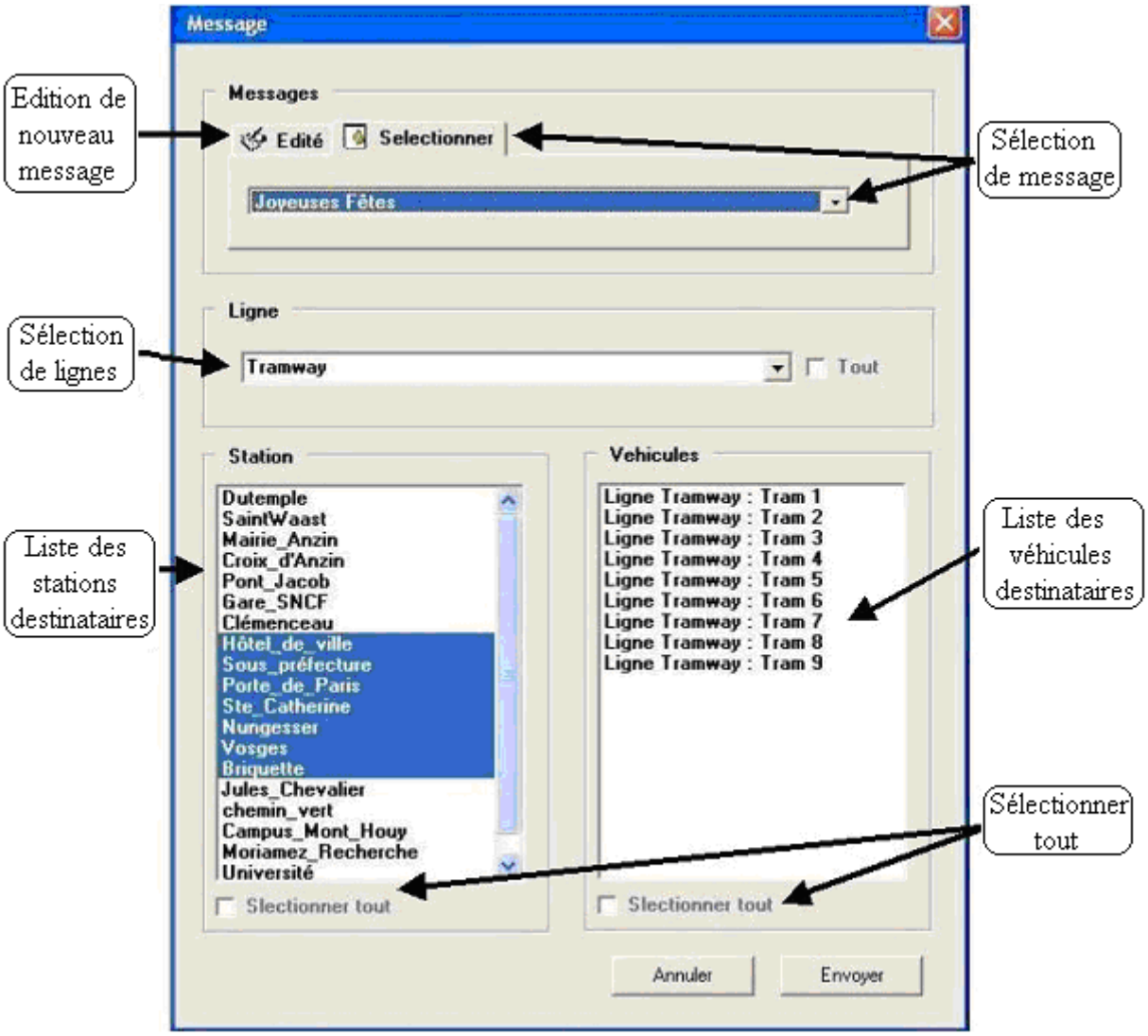

Figure 4.7 : Agent *Interface Message* du SAI [Trabelsi, 2006]

3. Application de l'environnement d'évaluation EISEval proposé pour évaluer le SAI

3.1. Préparation de l'évaluation

Une expérimentation a été réalisée au sein de notre laboratoire avec dix sujets. Ces sujets sont une femme et neuf hommes âgés de vingt-quatre à trente ans (25 ans en moyenne). Ils sont doctorants, ingénieurs ou futurs ingénieurs en Informatique ou en Automatique. Donc, ils sont habitués à l'utilisation des logiciels informatiques en général mais ils ne sont pas des utilisateurs expérimentés en régulation. La préparation se compose de deux étapes : (1) le déploiement des modules et (2) la configuration et le lancement de ceux-ci.

3.1.1. Déploiement des modules

Il s'agit d'abord de déployer quatre modules : le SAI, le SAD, le SAE et l'environnement d'évaluation EISEval proposé, sur différents machines. La figure 4.8 illustre la configuration de l'expérimentation.

Le SAI et le SAD ont été installés sur une première machine. C'est sur celle-ci que les sujets de l'expérimentation (prenant le rôle de régulateur) ont eu à interagir avec ces deux

modules. Le simulateur SAE a été installé sur une deuxième machine ; rappelons que le SAE simule la circulation du réseau de transport public. L'environnement EISEval a été installé sur une troisième machine avec la visée d'aider l'évaluateur à évaluer le SAI et à suggérer des améliorations au concepteur du SAI en se basant sur ses critiques. Tous ces systèmes communiquent entre eux à travers le mécanisme de socket[19].

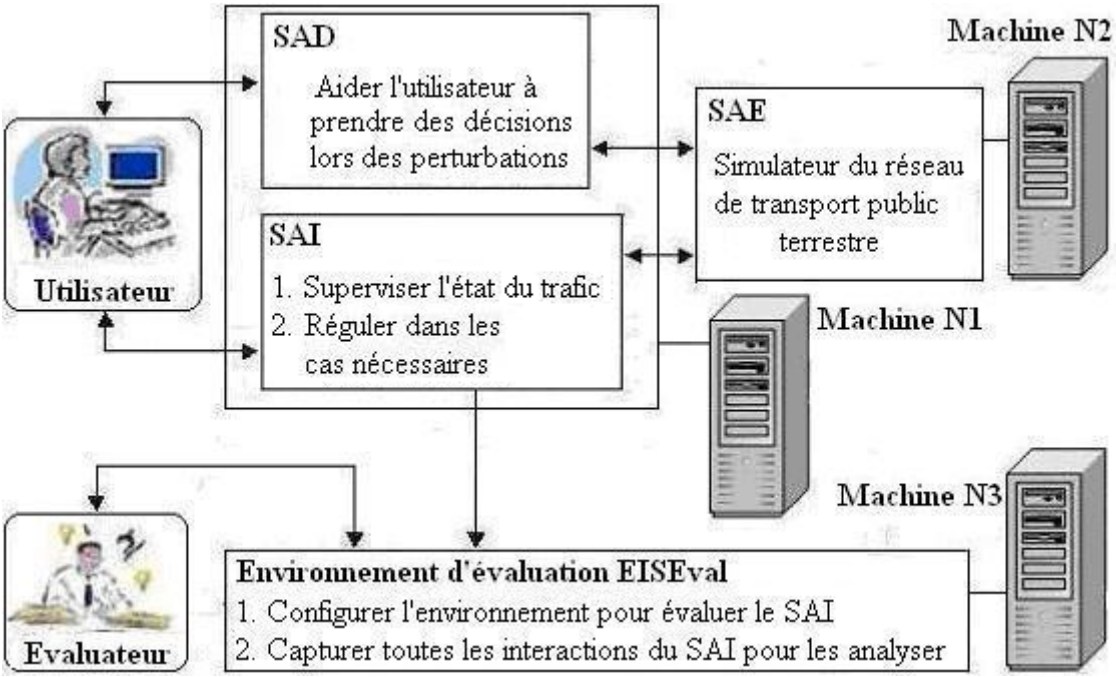

Figure 4.8 : Configuration de l'expérimentation

3.1.2. Configuration et lancement des modules

Cette étape concerne les activités suivantes :

- **Configurer l'environnement EISEval**. Après avoir installé ces 4 modules sur les trois machines, l'environnement EISEval a été configuré pour évaluer le SAI. L'évaluateur a dû saisir un ensemble d'informations relatives au SAI en utilisant le module 7 de l'environnement (cf. 3.5.6 du chapitre 3).
- **Lancement des modules**. On doit lancer les modules avant qu'un sujet quelconque participant à l'expérimentation effectue ses tâches. Le lancement se déroule de cette façon :

  Au début, le module 1 de l'environnement est lancé (il ne faut pas lancer les modules restants (modules 2, 3, 4, 5, 6) de l'environnement). Ensuite, le SAI et le SAD sont lancés. (l'ordre de ces lancements étant optionnel). Enfin, le SAE est lancé (le SAE simule la circulation des véhicules dans le réseau). Chaque sujet supervise ce processus de circulation et suit un scénario constitué de tâches que nous allons présenter dans la section suivante. Rappelons que le module 1 capture les interactions du SAI (interactions entre l'utilisateur et le SAI sous forme des EVIUs, interactions entre les agents sous forme des services) et stocke les données correspondantes dans des bases de données pour l'analyse ultérieure par les autres modules de l'environnement.

[19] socket : la communication logique entre deux systèmes reliés au réseau Internet, d'après la définition du [Lexique d'Internet-http://www.apiguide.net/08aide/lexique.htm]

3.2. Scénario de l'expérimentation

Pendant l'expérimentation, les informations concernant les positions des véhicules pendant leurs déplacements sont envoyées, en temps réel, au SAI par le Simulateur SAE. Les sujets ont dû réaliser neuf tâches de régulation et de notification. Le tableau 4.1 décrit ces tâches. Les six premières sont relativement simples à réaliser, les trois dernières sont assez complexes. Toutes les tâches concernent l'envoi de messages aux stations et/ou aux véhicules. En cas de perturbations (retard, panne de véhicule), le SAE envoie ces perturbations au SAI et au SAD. Le tableau 4.2 décrit les traitements que les sujets doivent faire en cas de perturbations. Après avoir réalisé une tâche ou un traitement de perturbation, les sujets doivent attendre le retour du SAI.

Tableau 4.1 : Les neuf tâches de régulation et de notification à effectuer

Tâches	**Description des tâches** (sans retour du SAI, la tâche est à refaire)
T1	Envoyer un message à la Station « Gare SNCF » de la ligne tramway : « Arrêt du prochain tramway 2 minutes en station »
T2	Envoyer ces deux messages au véhicule N3 de la ligne Tram: « Arrêtez-vous 2 minutes à la Gare SNCF » pour son conducteur et « Nous allons arrêter 2 minutes à la Gare SNCF » pour ses voyageurs
T3	Envoyer un message à la Station « St Wast » de la ligne 15 : « Arrêt du prochain bus 15 de 3 minutes en station »
T4	Envoyer ces deux messages au véhicule N4 de la ligne 15: « Arrêtez-vous 3 minutes à St Wast » pour son conducteur et « nous allons arrêter 3 minutes à St Wast » pour ses voyageurs
T5	Envoyer un message à la Station « Vaillant » de la ligne 16 : « Arrêt du prochain bus 16 de 3 minutes en station »
T6	Envoyer deux messages au véhicule N5 de la ligne 16: « Arrêtez-vous 3 minutes à la station Vaillant » pour son conducteur et « nous allons arrêter 3 minutes à la station Vaillant » pour ses voyageurs
T7	Envoyer un message à 3 Stations « Canada », « Ardenne », « Concorde » de la ligne 62 : « Arrêt non desservi demain et après demain en raison de travaux »
T8	Envoyer un message à toutes les stations de toutes les lignes : « Risque de perturbation le lundi de la semaine prochaine »
T9	Envoyer un message à tous les véhicules de toutes les lignes : « Joyeuses Fêtes »

Tableau 4.2 : Traitements à réaliser pour réagir aux perturbations que le SAI a reçues du SAE

Perturbation reçue du SAE	**Traitements à réaliser en cas de perturbation** (sans retour du SAI, la tâche est à refaire)
Retard d'un véhicule de X (X minutes inférieur ou égal à 7 minutes)	1. Fermer la fenêtre d'avertissement du retard 2. Utiliser le SAI pour envoyer les deux messages suivants à ce véhicule : - « En retard de X minutes, veuillez accélérer » pour son conducteur. - « Attention, vitesse plus rapide pour récompenser le retard » pour ses voyageurs.
Retard d'un véhicule de X minutes (X supérieur à 7 minutes)	1. Fermer la fenêtre d'avertissement de retard 2. Utiliser le SAI pour envoyer les deux messages suivants à ce véhicule : - « Vous êtes en retard de X minutes » pour son conducteur - « Attention, retard de X minutes » pour ses voyageurs 3. Envoyer le message à la station prochaine de ce véhicule : « Retard de X minutes », puis appliquer la solution de régulation proposée par le SAD[20]
Panne d'un véhicule.	1. Fermer la fenêtre d'avertissement de panne 2. Envoyer les deux messages suivants à ce véhicule : - « Le service de dépannage arrive dans 10 minutes » pour son conducteur - « Bus en panne, merci de votre compréhension » pour ses voyageurs

S'il n'y pas de message de retour du SAI, alors les sujets doivent refaire cette tâche ou ce traitement. L'environnement d'évaluation capture les données durant l'expérimentation, puis il les analyse. Ces résultats d'analyse doivent aider l'évaluateur à détecter des problèmes liés

[20] Les lecteurs peuvent consulter [Sidi, 2006] pour prendre connaissance du SAD qui propose les solutions de régulation pour le retard des véhicules.

au système SAI, à ses utilisateurs et à proposer des améliorations au concepteur du SAI. Ces résultats d'évaluation sont présentés plus loin.

Après la capture et le stockage des données par le module 1, l'évaluateur peut utiliser l'environnement pour évaluer le SAI selon la démarche décrite dans la section suivante.

3.3. Démarche d'évaluation d'un système interactif en utilisant l'environnement EISEval

Le module 6 de l'environnement d'évaluation fournit à l'évaluateur des indications nécessaires pour effectuer ses tâches (cf. 3.5.5 du chapitre 3). En effet, l'évaluateur se base sur les critères d'évaluation fournis par le module 6 (ces critères sont associés aux résultats d'analyse issus des modules 3, 4, 5 de l'environnement d'évaluation) pour critiquer le système interactif proposer des améliorations au concepteur. Ces critères d'évaluation peuvent être génériques ou spécifiques. Dans ce cas d'évaluation du SAI, l'évaluateur a ajouté à la liste des critères du module 6 deux critères d'évaluation spécifiques au SAI :

« *Facilité de l'utilisateur à trouver la ligne, la station ou le véhicule souhaité* » et « *Facilité et rapidité de l'utilisateur à traiter les perturbations reçues des véhicules* »

Rappelons que chaque critère d'évaluation du module 6 est composé de quatre parties (cf. 3.5.5 du chapitre 3) : *nom du critère* ; *définition du critère* ; *interprétation des résultats d'analyse pour une évaluation selon ce critère* (il s'agit des associations entre les critères d'évaluation fournis par le module 6 et les résultats d'analyse issus des modules 3, 4, 5 pour fournir à l'évaluateur une indication concrète sur la manière d'exploiter ces résultats d'analyse) ; *critiques et améliorations proposées par l'évaluateur en se basant sur ce critère*. L'évaluateur a critiqué le SAI et proposé au concepteur du SAI des améliorations en se basant sur les critères d'évaluation fournis. Dans le cas de cette expérimentation, nous avons joué le rôle de l'évaluateur. La figure 4.9, sous forme d'un diagramme d'activité UML, illustre la démarche de l'évaluation d'un système interactif en utilisant l'environnement EISEval.

Selon cette démarche, l'évaluateur accède à chaque critère d'évaluation du module 6 pour prendre connaissance de la manière d'exploiter les résultats d'analyse issus des modules 3, 4 et 5 pour évaluer le système selon ce critère. Ensuite, l'évaluateur accède aux résultats d'analyse associés pour les interpréter selon l'indication qu'il a reçue. Enfin, le module 6 permet à l'évaluateur de saisir des critiques et des améliorations en se basant sur le critère concerné. Après avoir suivi la démarche, l'évaluateur peut produire un document imprimable contenant les résultats d'évaluation du système (critiques, améliorations proposées).

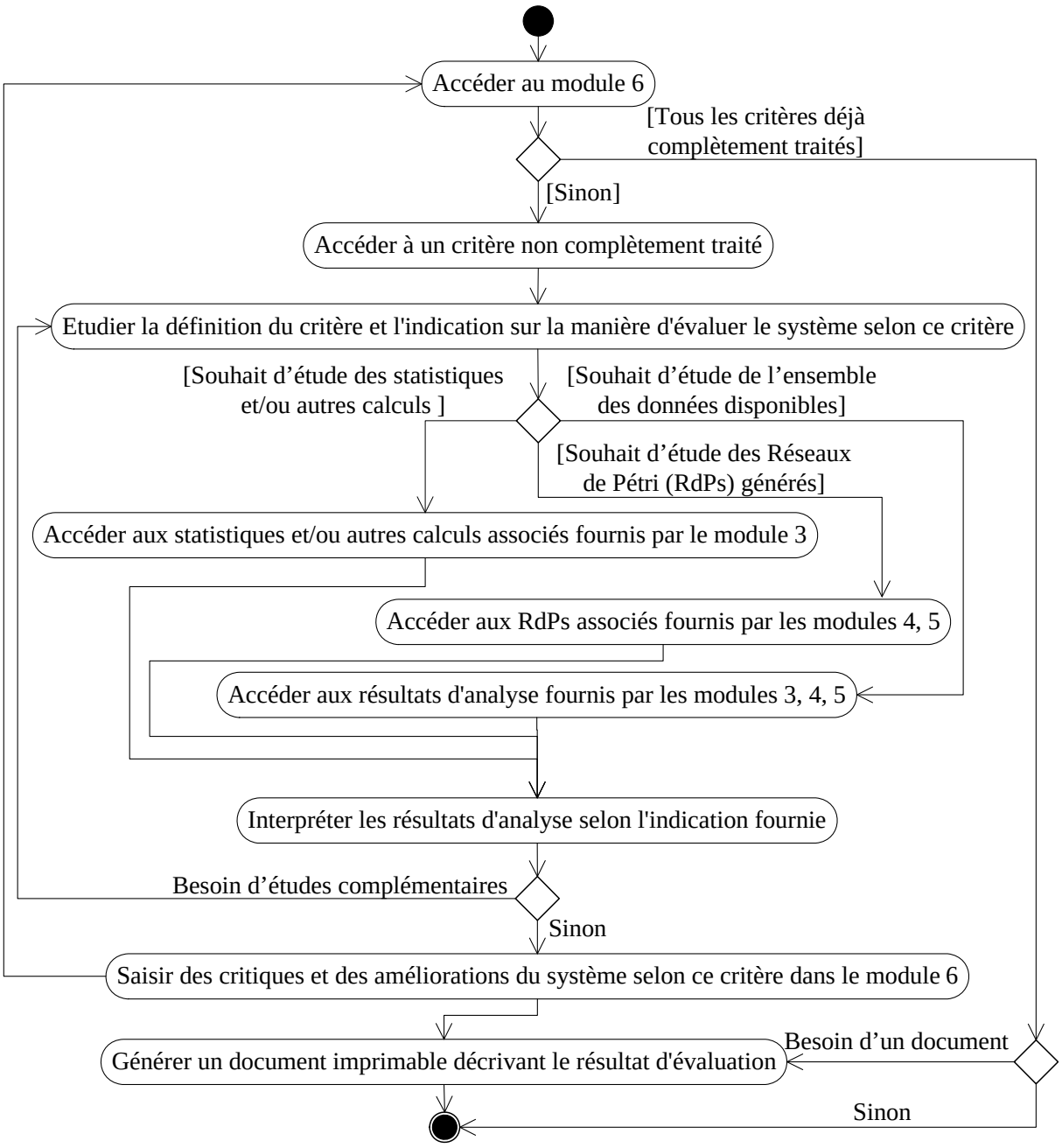

Figure 4.9 : Démarche d'évaluation d'un système interactif en utilisant l'environnement EISEval

4. Résultats d'évaluation du SAI en se basant sur les critères du module 6

Dans cette section, les résultats d'évaluation du SAI en se basant sur les critères du module 6 seront détaillés. Chaque critère sera passé en revue. Des captures d'écran des résultats d'analyse issus des modules 3, 4, 5 seront présentées. Les RdPs générés sont souvent très complexes, particulièrement losque les sujets ont effectué beaucoup de manipulations inutiles en situation réelle. Donc, les captures complètes d'écran de ces RdPs auraient pu sembler trop chargées pour les lecteurs. Afin d'être pédagogique et concis, pour chaque RdP présenté dans la section suivante, nous montrerons seulement une partie nécessaire de sa capture d'écran. Une de nos perspectives sera d'ailleurs de faciliter à l'évaluateur l'étude de ces RdPs complexes (cf. chapitre 5).

4.1. Critère : Lisibilité

Le critère *Lisibilité* concerne les caractéristiques lexicales de présentation des informations sur l'écran pouvant entraver ou faciliter la lecture de ces informations (luminance des caractères, contraste des caractères, dimension des lettres, espacement entre les mots, espacement entre les lignes, espacement entre les paragraphes, longueur des lignes, etc.). Par convention, le critère *Lisibilité* ne concerne ni le feedback ni les messages d'erreurs [Bastien et Scapin, 1993].

La performance est accrue lorsque la présentation des informations à l'écran tient compte des caractéristiques cognitives et perceptives des utilisateurs. Une bonne *lisibilité* facilite la lecture des informations présentées. Ainsi par exemple, les lettres sombres sur fond clair sont plus faciles à lire que l'inverse ; le texte présenté en lettres majuscules et minuscules est lu plus rapidement que le texte présenté seulement en lettres majuscules. [Bastien et Scapin, 1993]

4.1.1. Interprétation des résultats d'analyse pour une évaluation selon ce critère

L'évaluateur peut utiliser les résultats d'analyse de l'environnement d'évaluation pour évaluer partiellement le système selon ce critère :

- Si la fréquence d'apparition des événements interfaces utilisateurs (EVIUs) qui permettent à l'utilisateur de voir l'interface plus clairement (comme *Zoom Avant*, *Zoom Arrière*, changement de la taille des vues ou des fenêtres, etc.) est élevée, alors l'évaluateur peut conclure que la *lisibilité* de l'interface du système n'est pas bonne.[21]
- L'évaluateur peut aussi utiliser les réseaux de pétri (RdPs) générés par le module 4 et le résultat d'analyse des événements interfaces utilisateurs (EVIUs) du module 3 pour évaluer ce critère. Ces RdPs générés reconstituent les processus des activités réelles de l'utilisateur et du système pour réaliser les tâches. A travers ces processus, l'évaluateur peut détecter les EVIUs correspondant aux présentations des informations. Si l'intervalle de temps entre l'apparition d'un EVIU correspondant à l'affichage d'une fenêtre et l'apparition d'un autre EVIU correspondant à l'action suivante de l'utilisateur sur cette fenêtre est long, alors l'évaluateur peut interpréter que l'utilisateur a éventuellement passé beaucoup de temps sur cette fenêtre avant d'effectuer l'action suivante. Dans ce cas, l'évaluateur peut poser cette question : « pourquoi l'utilisateur a-t-il pris si longtemps pour effectuer l'action suivante ? », il peut penser que la mauvaise *lisibilité* est une des raisons possibles. Donc, une autre méthode d'évaluation, par exemple un questionnaire ou une verbalisation, peut être utilisée en complément afin de connaître la raison et pour avoir une évaluation plus complète et exacte relativement à ce critère.

4.1.2. Critiques et améliorations proposées par l'évaluateur en se basant sur ce critère

A travers l'expérimentation avec les dix sujets, en utilisant le résultat d'analyse du module 3 de l'environnement (la figure 4.10 montre une partie de la capture d'écran à ce sujet), l'évaluateur peut constater que les EVIUs *Zoom Avant*, *Zoom Arrière* n'ont jamais été déclenchés par l'ensemble des sujets. En conséquence, on peut supposer que l'interface du système SAI est bien *lisible* et que le sujet n'a aucune difficulté à lire les informations sur l'interface du SAI [22]. Pour aller plus en profondeur à ce critère, des études complémentaires seraient nécessaires.

[21] Déjà présenté dans le chapitre 3 dans un but d'illustration du module 6 (cf. 3.5.5).
[22] Déjà présenté dans le chapitre 3 dans un but d'illustration du module 6 (cf. 3.5.5).

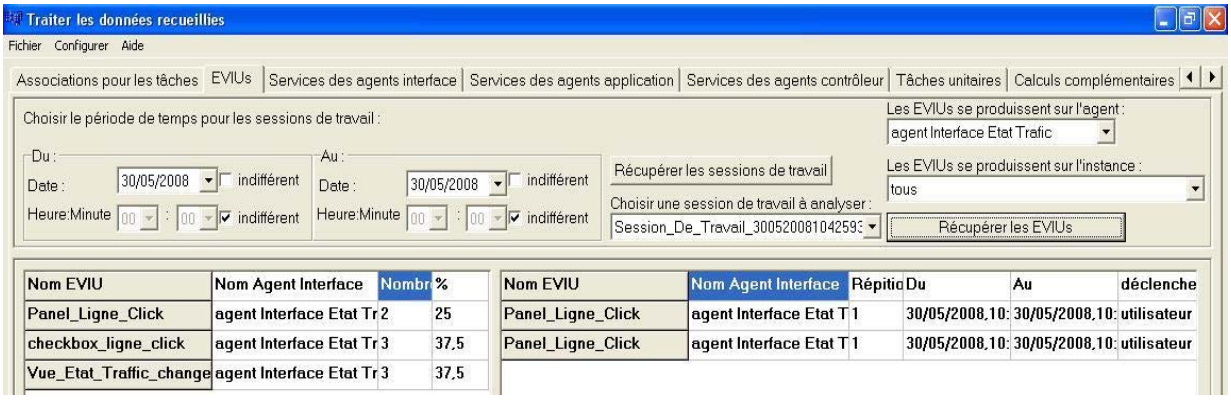

Figure 4.10 : Extrait de la capture d'écran du résultat d'analyse des EVIUs du module 3

4.2. Critère : Protection contre les erreurs

Le critère *Protection Contre les Erreurs* concerne les moyens mis en place pour détecter et prévenir les erreurs d'entrée de données ou de commandes ou les actions aux conséquences néfastes. Il est préférable de détecter les erreurs lors de la saisie plutôt que lors de la validation : ceci évite de perturber la planification [Bastien et Scapin, 1993]. D'après ces auteurs, il existe une différence entre *Protection Contre les Erreurs* et *Incitation* : plusieurs façons d'offrir une protection contre les erreurs sont possibles. On peut par exemple mettre en place un mécanisme automatique de vérification des entrées. Ainsi, au moment de la validation des entrées, un message d'erreur apparaît si le format d'entrée n'est pas conforme à ce qui est attendu. Il s'agit dans ce cas-ci du critère *Protection Contre les Erreurs*. Une autre façon consiste à fournir une information renseignant les utilisateurs sur le type de données attendues ou encore sur le format de ces entrées : il s'agit alors du critère *Incitation*. Ces deux mécanismes peuvent aussi coexister [Bastien et Scapin, 1993].

4.2.1. Interprétation des résultats d'analyse pour une évaluation selon ce critère

Quand l'utilisateur commet une erreur (saisie de fausses données, fausses actions) mais que les EVIUs correspondant aux affichages de messages d'erreur, et affichages de fenêtres d'erreur ne se produisent pas, alors l'évaluateur ne peut pas conclure que selon le critère *Protection Contre les Erreurs*, le système est bon. Dans le cas contraire, ce critère est assuré dans le système.

4.2.2. Critiques et améliorations proposées par l'évaluateur en se basant sur ce critère

Après cette expérimentation, en utilisant le résultat d'analyse des EVIUs du module 3, l'évaluateur peut constater que les EVIUs correspondant aux affichages de messages d'erreur ne se sont jamais produits lorsque les sujets ont saisi des données ; en conséquence, l'évaluateur pourrait tirer deux conclusions:

- Les sujets n'ont pas commis d'erreurs en saisissant les données.
- Les principes de prévention des erreurs du système interactif SAI se sont montrés efficaces durant l'expérimentation, en empêchant les sujets de commettre des erreurs de saisie des données durant les situations rencontrées.

En fait, dans le SAI, il y un seul cas où les utilisateurs doivent saisir des données. Quand ils veulent envoyer un message aux stations ou aux véhicules, ils doivent le composer mais il n'y a aucune contrainte sur le format de données saisies. Donc, l'évaluation selon ce critère

pour les scénarios prédéfinis du système SAI n'a pas permis de mettre de problème particulier en évidence.

4.3. Critère : Facilité de l'utilisateur à trouver les lignes, les stations ou les véhicules souhaités

Celui-ci peut être considéré comme un critère métier spécique au SAI. Ce critère évalue si l'interface du SAI permet à l'utilisateur de trouver facilement une ligne, station ou véhicule dans les cas d'envoi de message à une station ou un véhicule d'une ligne quelconque.

4.3.1. Interprétation des résultats d'analyse pour une évaluation selon ce critère

Les manipulations inutiles de l'utilisateur avant de trouver une ligne, une station ou un véhicule peuvent être utilisés pour évaluer le système selon ce critère. A travers les RdPs générés qui reconstituent les processus des activités réelles de l'utilisateur et du système pour réaliser les tâches d'envoi d'un message, l'évaluateur peut savoir si l'utilisateur a effectué ces manipulations inutiles (sous forme des EVIUs) avant de trouver ce qu'il cherchait. Si le nombre d'apparition de ces EVIUs est élevé, alors l'évaluateur peut conclure que l'utilisateur a éprouvé des difficultés pour les trouver.

4.3.2. Critiques et améliorations proposées par l'évaluateur en se basant sur ce critère

Les problèmes suivants sont détectés à l'aide de l'environnement :

A) L'évaluateur a pu constater que les sujets ont effectué beaucoup de **manipulations inutiles** avant de trouver la ligne, la station ou le véhicule concerné. En effet, plusieurs sujets ont été dans ce cas lors de tâches d'envoi de message. Cette difficulté des sujets pour la recherche des lignes, des stations et des véhicules concernés est exprimée par plusieurs manipulations inutiles que ces sujets ont effectuées pour trouver les lignes adéquates. Avant de présenter ces manipulations inutiles des sujets, il s'agit d'expliquer comment accéder à une ligne, une station et un véhicule quelconque dans le SAI :

- Pour accéder à une ligne, l'utilisateur doit effectuer une manipulation sur l'agent *Etat_Trafic*. En effet, sur cet agent, il faut que l'utilisateur clique une seule fois sur un des « *checkboxes* » ou sur un des « *panels* » de cet agent pour accéder à la ligne correspondante parce que ces « *checkboxes* » ou « *panels* » correspondent aux lignes du trafic. Ces EVIUs sont identifiés respectivement par *eviu4,1-I* (*checkbox_ligne_click*) et *eviu1,1-I* (*Panel_Ligne_Click*). La figure 4.11 illustre ces deux manipulations sur l'agent *Etat_Trafic*. L'agent *Etat_Ligne* est affiché (comme l'illustre la figure 4.3) pour montrer la ligne concernée lorsque l'utilisateur effectue un de ces deux clics.
- Pour accéder à la fenêtre de propriétés d'une station ou d'un véhicule quelconque afin de lui envoyer un message, l'utilisateur doit cliquer (une seule fois) sur l'image de la station concernée ou sur l'image du véhicule concerné de l'agent *Etat_Ligne*. Ces EVIUs sont identifiés respectivement par *eviu1,3-I* (*Image_Station_Click*) et *eviu1,4-I* (*Image_Vehicule_Click*). La figure 4.12 illustre ces deux manipulations sur l'agent *Etat_Ligne*. Les fenêtres de propriétés d'une station et d'un véhicule sont ouvertes après que ses images soient cliquées ; elles ont été déjà illustrées respectivement (cf. figures 4.4, 4.5).

En réalité, plusieurs sujets ont effectué des manipulations inutiles avant de trouver les lignes ou stations concernées. Ces manipulations inutiles peuvent être observées sur les réseaux de pétri (RdPs) générés ; elles sont présentées dans le tableau 4.3.

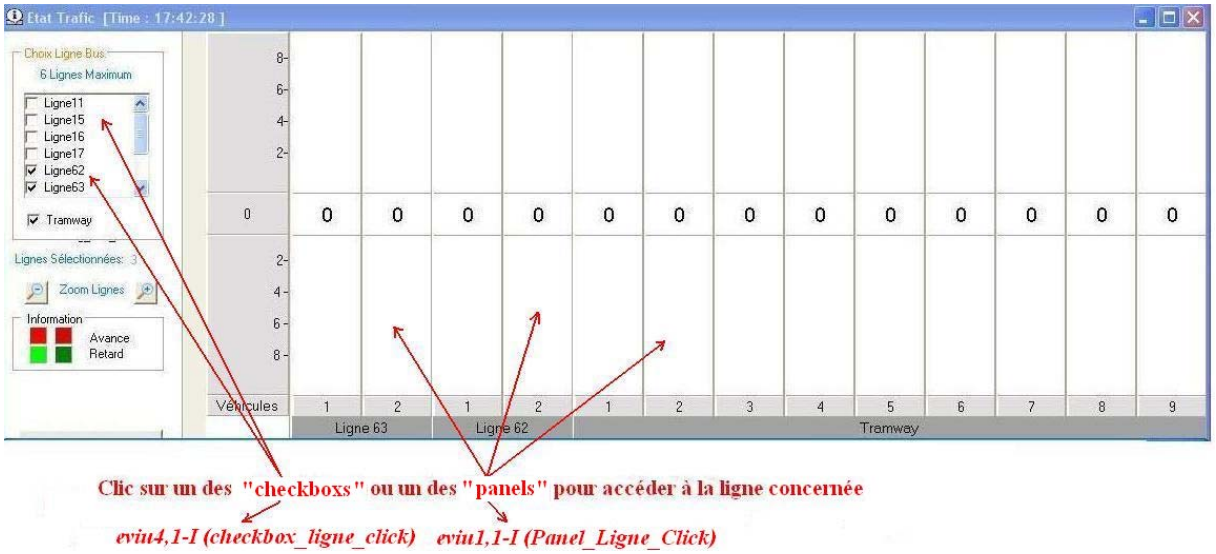

Figure 4.11 : Agent Etat_Trafic du SAI : comment accéder à une ligne

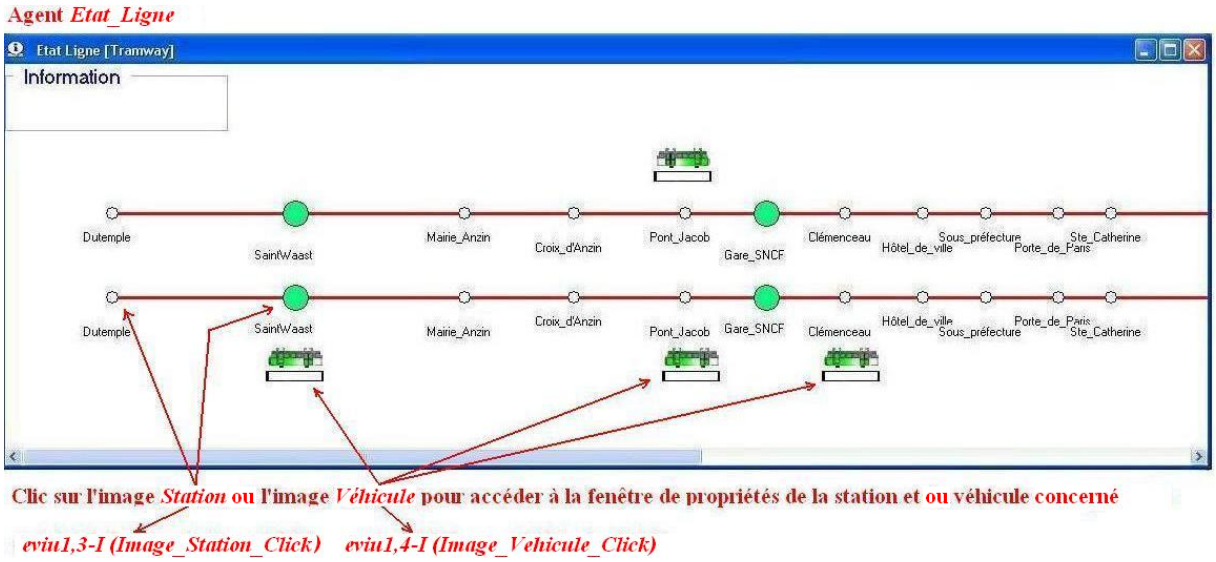

Figure 4.12 : Agent Etat_Ligne du SAI : comment accéder à une station et à un véhicule

En plus, l'évaluateur a pu remarquer que concernant ce critère : les sujets sont devenus plus rapides dans leur recherche de lignes, stations ou véhicules pendant l'expérimentation grâce très certainement au phénomène d'apprentissage. Il a remarqué aussi que, pour l'instant, dans la version courante du SAI, si un utilisateur veut chercher une station d'une ligne pour lui envoyer un message, il a une seule façon de le faire. En effet, il doit regarder toutes les stations de cette ligne et lire tous les noms des stations pour choisir la bonne. En conséquence, l'utilisateur peut prendre beaucoup de temps, notamment dans le cas des lignes comprenant un grand nombre de stations.

Tableau 4.3 : Manipulations inutiles des sujets pendant l'expérimentation avant de trouver une ligne, une station ou un véhicule concerné

Tâches ou traitements de perturbation	Sujets	Manipulations inutiles du ou des sujets
Tâches 2, 6 (envoi d'un message à un véhicule)	2	Ce sujet a cliqué plusieurs fois sur les différents « *panels* » de l'agent *Etat_Trafic* pour chercher la ligne concernée. *L'eviu1,1-I* (*Panel_Ligne_Click*) s'est produit plusieurs fois avant de trouver la ligne concernée. La figure 4.13 illustre une partie de la capture d'écran du RdP généré pour le sujet 2 pendant sa réalisation de la tâche 2
Traitement du retard d'un véhicule (envoi d'un message à ce véhicule)	7, 8	Ces deux sujets ont cliqué plusieurs fois sur les différents « *panels* » de l'agent *Etat_Trafic* pour chercher la ligne à laquelle ce véhicule en retard appartient. *L'eviu1,1-I* (*Panel_Ligne_Click*) s'est produit plusieurs fois avant de trouver la ligne concernée.
Tâche 5 (envoi d'un message à une station)	2, 9	Ces deux sujets ont cliqué plusieurs fois sur les différents « *checkboxes* » de l'agent *Etat_Trafic* pour chercher la ligne concernée. *L'eviu4,1-I* (*checkbox_ligne_click*) s'est produit plusieurs fois avant de trouver la ligne concernée.
Tâche 3 (envoi d'un message à une station)	2, 9	Ces deux sujets ont dû choisir exactement la bonne station pour lui envoyer un message mais ils se sont trompés. Ils ont sélectionné la mauvaise station et ouvert sa fenêtre de propriétés ; donc ils ont dû fermer cette fenêtre par un clic sur son bouton *Annuler* (*eviu9,3-I* (*bouton_Annuler_Click*)) pour rechoisir la bonne station. Le sujet 9 a aussi cliqué plusieurs fois sur les différents « *checkboxes* » de l'agent *Etat_Trafic*. La figure 4.14 illustre une partie de la capture d'écran du RdP généré pour le sujet 9 pendant sa réalisation de cette tâche 3.
Tâche 2 (envoi d'un message à un véhicule)	5	Ce sujet a dû choisir un véhicule pour lui envoyer un message mais il s'est trompé. Il a sélectionné le mauvais véhicule et ouvert sa fenêtre de propriétés, donc il a dû fermer cette fenêtre par un clic sur son bouton *Annuler* (*eviu16,4-I* (*bouton_Annuler_Click*)) pour rechoisir le bon véhicule.
Traitement du retard d'un véhicule (envoi d'un message à ce véhicule)	7	Idem précédemment
Tâche 7 (envoi d'un message à trois stations différentes)	7	Ce sujet a cliqué plusieurs fois sur les différents « *panels* » de l'agent *Etat_Trafic*. *L'eviu1,1-I* (*Panel_Ligne_Click*) s'est produit plusieurs fois avant de trouver la ligne concernée pour envoyer le message à la première station.
Tâche 7 (envoi d'un message à trois stations différentes)	9	Ce sujet a cliqué plusieurs fois sur les différents « *checkboxes* » de l'agent *Etat_Trafic*. *L'eviu4,1-I* (*checkbox_ligne_click*) s'est produit plusieurs fois avant de trouver la ligne concernée.
Traitement du retard d'un véhicule (envoi d'un message à la prochaine station du véhicule en retard pour informer les voyageurs de ce retard)	5, 6	Ces deux sujets ont cliqué sur l'image d'un véhicule (*l'eviu1,4-I* (*Image_Vehicule_Click*)) pour ouvrir sa fenêtre de propriétés. C'est une fausse action ; donc ils ont dû fermer cette fenêtre par un clic sur son bouton *Annuler* (*eviu16,4-I* (*bouton_Annuler_Click*)). Ensuite, ils ont cliqué sur l'image de la station concernée (*l'eviu1,3-I* (*Image_Station_Click*)) à laquelle ils devaient envoyer un message, pour ouvrir sa fenêtre de propriétés et ils ont continué à faire les autres manipulations pour l'envoi d'un message.

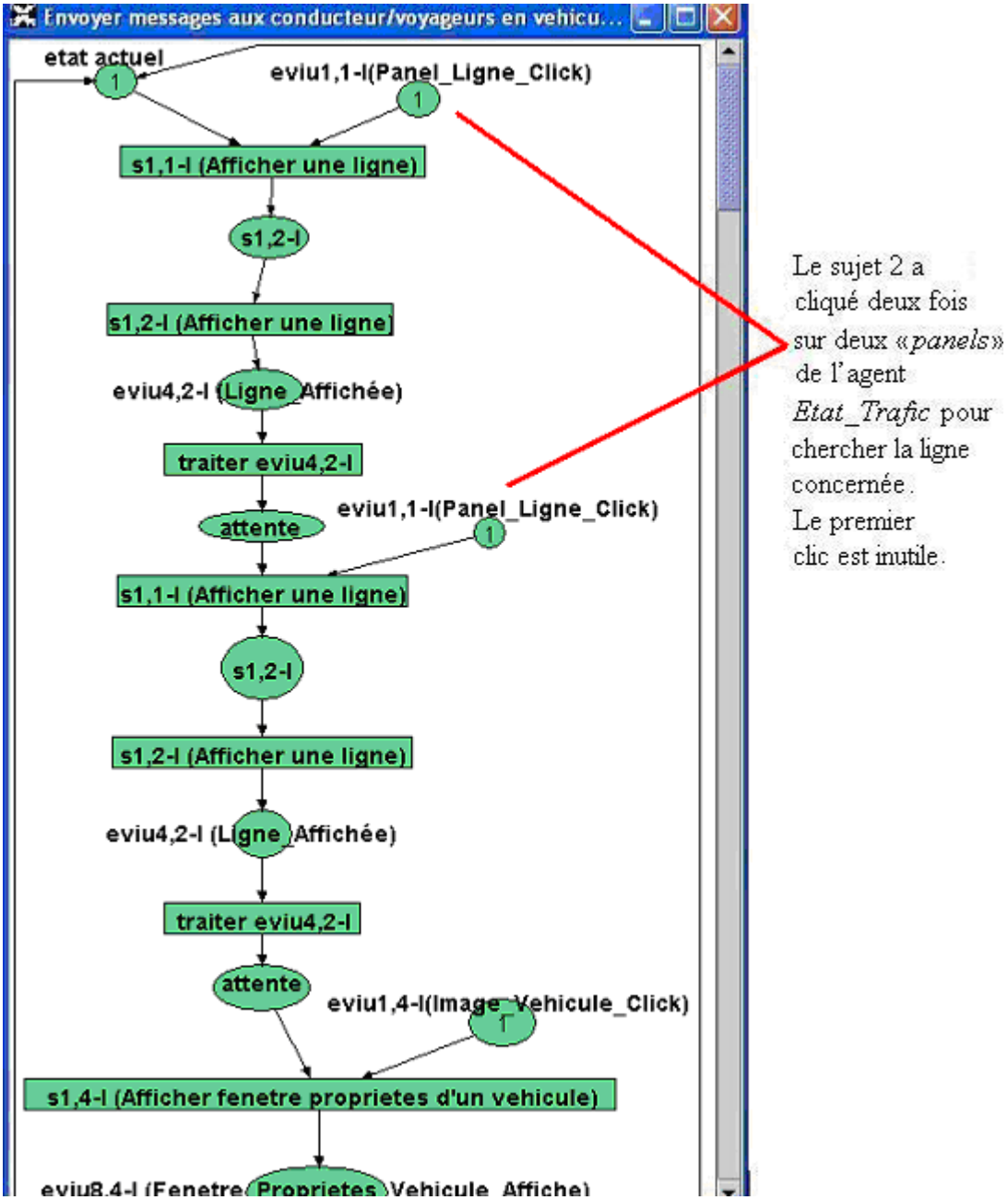

Figure 4.13 : Extrait de la capture d'écran du RdP généré pour le sujet 2 pendant sa réalisation de la tâche 2

B) Améliorations proposées

A partir des critiques présentées ci-dessus, l'évaluateur peut suggérer au concepteur d'améliorer l'agent *Etat_Ligne* du SAI comme l'illustre la figure 4.15. Il faut ajouter à cet agent deux boîtes de liste (Listbox) qui contiennent respectivement des stations et des véhicules de la ligne courante comme l'illustre la figure 4.15b. Ces deux boîtes de liste permettent aux utilisateurs de choisir rapidement et exactement la station ou le véhicule qu'ils visent. Il est suggéré au concepteur d'améliorer aussi l'agent *Etat_Trafic* du SAI pour faciliter la recherche de la ligne. Cette amélioration sera présentée ultérieurement.

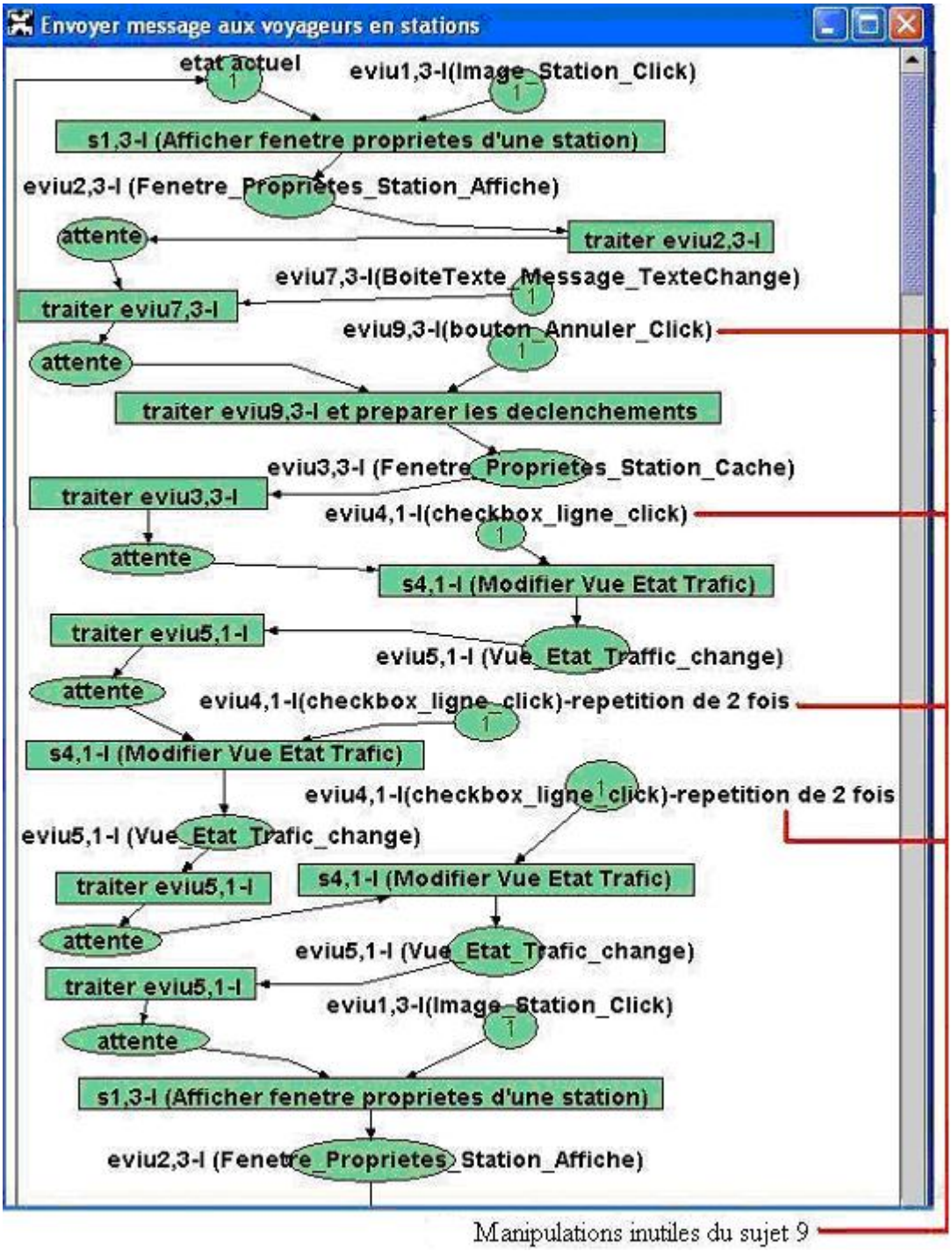

Figure 4.14 : Extrait de la capture d'écran du RdP généré pour le sujet 9 pendant sa réalisation de la tâche 3

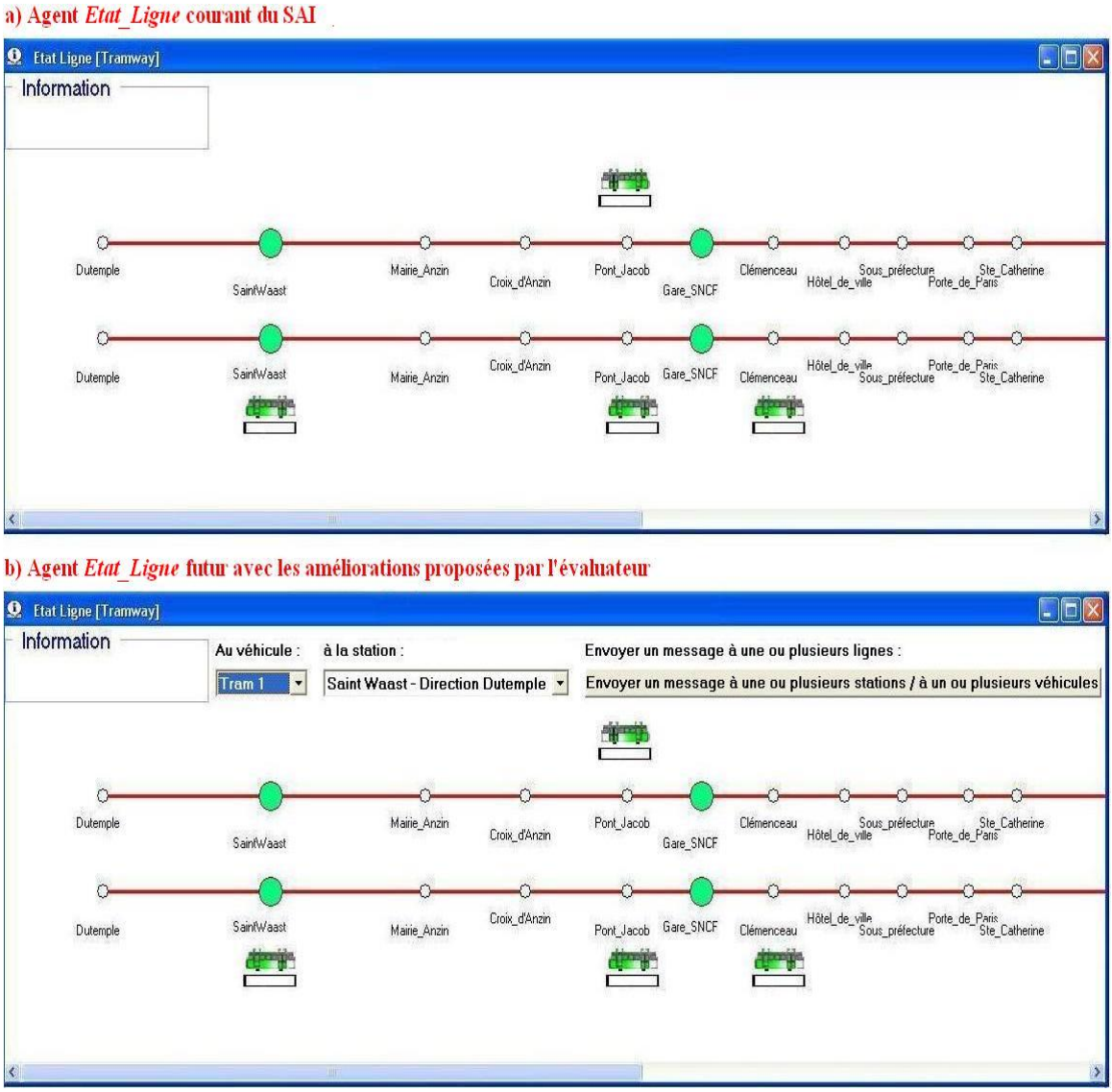

Figure 4.15 : Agent *Etat_Ligne* du système SAI. La partie a illustre l'agent *Etat_Ligne* courant. La partie b illustre l'agent *Etat_Ligne* après amélioration

4.4. Critère : Feedback immédiat

Le *Feedback Immédiat* concerne les réponses du système interactif consécutives aux actions des utilisateurs, lesquelles peuvent être un simple appui sur une touche ou la saisie d'une séquence de commandes. Dans tous les cas, l'ordinateur doit répondre, dans les plus brefs délais, avec un délai de réponse approprié et homogène selon les types des interactions. Dans tous les cas, une réponse aussi immédiate que possible doit être fournie à l'utilisateur, le renseignant sur l'action accomplie et sur son résultat [Bastien et Scapin, 1993].

La qualité et la rapidité du feedback sont deux facteurs importants pour l'établissement de la confiance et de la satisfaction des utilisateurs ainsi que pour leur compréhension du dialogue. Ces facteurs permettent aux utilisateurs de se faire une bonne représentation du fonctionnement du système. L'absence de feedback ou des délais trop importants entre les actions des utilisateurs et le feedback, peuvent déconcerter ces derniers, ce qui augmente la possibilité que les utilisateurs entreprennent des actions qui risquent d'entraver les opérations en cours [Bastien et Scapin, 1993].

4.4.1. Interprétation des résultats d'analyse pour une évaluation selon ce critère

L'évaluateur peut utiliser les RdPs générés par l'environnement d'évaluation (qui reconstituent les processus des activités réelles de l'utilisateur et du système lors de la réalisation des tâches) et les résultats d'analyse des EVIUs du module 3 pour évaluer le système selon ce critère. Si les EVIUs correspondant aux réponses du système et aux actions de l'utilisateur n'existent pas ou si les intervalles de temps entre les apparitions des EVIUs correspondant aux actions de l'utilisateur et les apparitions des EVIUs correspondant aux réponses du système sont longs, alors selon ce critère, le système n'est pas satisfaisant.

4.4.2. Critiques et améliorations proposées par l'évaluateur en se basant sur ce critère

Dans le cas du système SAI, tous les sujets reçoivent le feedback du système interactif lors de réalisation des tâches d'envoi d'un message aux stations ou aux véhicules. Dans le système SAI, quand l'utilisateur décide d'envoyer le message qu'il a déjà saisi, il doit :

- cliquer sur le bouton *OK* de l'agent *interface Véhicule* (*eviu15,4-I* (*boutonOK_Click*)) dans le cas d'envoi du message à un véhicule **(a)** ;
- cliquer sur le bouton *OK* de l'agent *interface Station* (*eviu8,3-I* (*boutonOK_Click*)) dans le cas d'envoi du message à une station **(b)** ;
- cliquer sur le bouton *Envoyer* de l'agent *interface Message* du SAI (*eviu13,5-I* (*boutonOK_Click*))) dans le cas d'envoi du message à plusieurs stations et/ou à plusieurs véhicules **(c)**.

Dans ces trois cas, après que l'utilisateur clique sur ces boutons d'envoi de message (cités en **(a)**, **(b)**, **(c)** ci-dessus), le message déjà saisi est envoyé et une fenêtre de feedback est retournée à l'utilisateur. Cet EVIU est identifié par *l'eviu15,5-I* (*BoiteResponse_FeedbackAuEnvoiMessage_Affiché*).

Il faut utiliser l'environnement d'évaluation pour vérifier si ces EVIUs (*eviu15,5-I*) sont toujours apparus après que les sujets aient cliqué sur ces boutons d'envoi de message (*eviu 15,4-I*, *eviu8,3-I*, *eviu13,5-I* cités en **(a)**, **(b)**, **(c)** ci-dessus).

Si c'est le cas, il faut examiner si les intervalles de temps entre les apparitions de ces EVIUs (*eviu15,5-I*) et celles des EVIUs correspondant aux clics sur ces boutons d'envoi du message (*eviu 15,4-I*, *eviu8,3-I*, *eviu13,5-I* cités à **(a)**, **(b)**, **(c)** ci-dessus) sont longs ou courts.

En utilisant les RdPs générés par le module 4 et les résultats d'analyse du module 3, l'évaluateur peut constater que les *eviu15,5-I* cités ci-dessus (qui correspondent au feedback du système) se sont toujours produits après que les sujets aient cliqué sur les boutons d'envoi d'un message (*eviu 15,4-I*, *eviu8,3-I*, *eviu13,5-I* cités à **(a)**, **(b)**, **(c)** ci-dessus) et avec des intervalles de temps entre apparitions courts, de l'ordre de 25-30 millisecondes.

D'après GNOME Human Interface Guidelines 2.2 [23], le temps de feedback acceptable relativement à une tâche de fond (background task) est de l'ordre d'une seconde. L'évaluateur peut donc conclure que ce critère *Feedback immédiat* est assuré dans le système SAI.

4.5. Critère : Incitation

Ce critère recouvre les moyens mis en oeuvre pour amener les utilisateurs à effectuer des actions spécifiques, qu'il s'agisse de saisie de données ou non. Ce critère englobe aussi tous les mécanismes ou moyens faisant connaître aux utilisateurs les alternatives, lorsque plusieurs

[23] GNOME Human Interface Guidelines 2.2, GNOME Documentation Library, http://library.gnome.org/devel/hig-book/stable/feedback-response-times.html.en

actions sont possibles, selon les états ou les contextes dans lesquels ils se trouvent. *L'Incitation* concerne également les informations permettant aux utilisateurs de savoir où ils en sont, d'identifier l'état ou contexte dans lequel ils se trouvent, de même que les outils d'aide et leur accessibilité [Bastien et Scapin, 1993].

Une bonne *incitation* guide les utilisateurs et leur évite par exemple d'avoir à apprendre une série de commandes. Elle permet aussi aux utilisateurs de savoir quel est le mode ou l'état en cours, où ils se trouvent dans le dialogue et ce qu'ils ont fait pour s'y trouver. Une bonne *incitation* facilite donc la navigation dans une application et permet d'éviter les erreurs [Bastien et Scapin, 1993].

4.5.1. Interprétation des résultats d'analyse pour une évaluation selon ce critère

L'évaluateur peut utiliser l'environnement d'évaluation pour évaluer ce critère :

- L'évaluateur peut utiliser le nombre d'apparitions des EVIUs correspondant aux affichages des messages et fenêtres d'erreurs lorsque l'utilisateur saisit les données ou commet des actions erronées, pour évaluer le système ce critère. Si le nombre d'apparitions de ces EVIUs est élevé, alors l'évaluateur peut interpréter que l'utilisateur a commis plusieurs erreurs pendant son interaction avec l'interface. Plus le nombre d'apparitions de ces EVIUs est élevé, plus le système, relativement à ce critère, peut être considéré comme mauvais.
- L'évaluateur peut aussi utiliser les RdPs générés et les résultats d'analyse des EVIUs du module 3 pour évaluer le système selon ce critère. A travers les navigations effectuées par l'utilisateur lors de la réalisation des tâches qui sont visualisées par les RdPs, l'évaluateur peut détecter les EVIUs correspondant aux manipulations inutiles, aux navigations et actions erronées de l'utilisateur. Si le nombre d'apparitions de ces EVIUs est élevé, alors l'évaluateur peut interpréter que *l'incitation* n'est pas bonne. S'il y a beaucoup de manipulations inutiles de l'utilisateur, alors, d'après [Lecerof et Paterno, 1998], l'évaluateur peut interpréter que l'interface n'est pas efficace et qu'elle est la cause de problèmes d'interprétation de l'utilisateur. Dans ce cas, l'évaluateur peut conseiller au concepteur d'y ajouter, par exemple, des étiquettes ou incitateurs divers plus explicites ou informatifs.
- Le nombre de tâches que l'utilisateur a pu réaliser est aussi utilisé pour évaluer le système selon ce critère. Si l'utilisateur n'a pas pu réaliser certaines tâches, alors une mauvaise *incitation* de l'interface peut être une des raisons possibles. En effet, l'évaluateur peut interpréter que l'utilisateur n'a pas pu trouver les chemins pour réaliser ces tâches. L'interface inefficace et inexplicite du système peut être la raison de ces échecs de l'utilisateur.
- Dans le cas où le système fournit à l'utilisateur plusieurs chemins possibles afin de réaliser une tâche quelconque, si le chemin optimal est rarement emprunté par les utilisateurs, alors l'évaluateur peut interpréter que *l'incitation* a des lacunes, et qu'en conséquence, les utilisateurs n'ont pas pu trouver le chemin optimal pour réaliser cette tâche.
- Le nombre d'apparitions des événements correspondant aux consultations de l'aide peut être utilisé pour évaluer ce critère. Si la fréquence d'apparition de ces événements est élevée, alors l'évaluateur peut interpréter que l'utilisateur a eu des difficultés dans son travail.
- Les intervalles de temps entre les apparitions des EVIUs qui correspondent aux actions de l'utilisateur pendant sa réalisation d'une tâche et les intervalles de temps entre les réalisations des tâches de l'utilisateur peuvent être utilisés pour évaluer ce critère.

- Quand l'utilisateur réalise une tâche quelconque, il doit effectuer plusieurs actions qui apparaissent sous forme d'EVIUs. Si les intervalles de temps entre les apparitions de ces EVIUs sont longs, alors l'évaluateur peut interpréter que l'utilisateur a pris beaucoup de temps à chercher une solution en termes de navigation dans l'application pour réaliser cette tâche. L'évaluateur peut supposer que la mauvaise *incitation* peut être une des raisons possibles pour cette lenteur de l'utilisateur.
- Similairement, si les intervalles de temps entre les réalisations des tâches sont longs, alors l'évaluateur peut supposer que la mauvaise *incitation* peut être une des raisons possibles pour cette lenteur. Le module 3 permet d'afficher les tâches réalisées dans l'ordre chronologique (cf. chapitre précédent). Donc, l'évaluateur peut en prendre connaissance, ils peuvent lui être utiles dans son évaluation.

Dans ces deux cas, une autre méthode d'évaluation, par exemple, l'utilisation d'un questionnaire peut compléter cet environnement pour avoir une évaluation plus complète et exacte relativement à ce critère.

4.5.2. Critiques et améliorations proposées par l'évaluateur en se basant sur ce critère

L'évaluateur a constaté que, en général, *l'incitation* de l'interface du SAI n'est pas très bonne en se basant sur les problèmes suivants détectés à l'aide de l'environnement :

A) Les sujets ont effectué plusieurs **manipulations inutiles** constatées sur les RdPs, relativement à la recherche de lignes, de station et de véhicules ; ils ont fait aussi d'autres manipulations inutiles présentées dans le tableau 4.4.

Tableau 4.4 : Autres manipulations inutiles des sujets pendant l'expérimentation

Tâches ou traitements de perturbation	Sujets	Manipulations inutiles
Tâche 1 (envoi d'un message à une station)	10	Après avoir ouvert la fenêtre de propriétés de la station pour lui envoyer un message, ce sujet a cliqué sur le bouton *Message_Prédéfini* de cette fenêtre comme l'illustre la figure 4.16b. Cette action inutile est identifiée par *l'eviu4,3-I* (*bouton_MessagePredefini_Click*). En conséquence, la fenêtre de l'agent *interface Message* du SAI a été ouverte (cf.figure 4.7). Cet EVIU est identifiée *par l'eviu1,5-I* (*Fenêtre_Messages_Affichée*). Ce sujet a dû fermer cette fenêtre par un clic sur son bouton *Annuler* (*l'eviu14,5-I* (*bouton_Annuler_click*). Ensuite, il a dû recommencer la réalisation de cette tâche. La figure 4.17 illustre une partie de la capture d'écran du RdP généré pour le sujet 10 pendant sa réalisation de la tâche 1.
Tâche 6 (envoi d'un message à un véhicule)	10	Ce sujet a cliqué sur l'image *Véhicule* de l'interface (*l'eviu1,4-I* (*Image_Vehicule_Click*)) pour ouvrir la fenêtre de propriétés de ce véhicule qui permet d'envoyer un message à ce véhicule (cf. figure 4.12). Mais, après, il a re-cliqué sur cette image *Véhicule*. C'est une action inutile.
Tâche 4 (envoi d'un message à un véhicule)	2	Ce sujet a commis la même erreur que le sujet 10 dans sa réalisation de la tâche 6 (cf. ci-dessus)
Tâche 7 (envoi d'un message à trois stations différentes)	6, 8	Le sujet 6 a cliqué sur l'image *Station* de l'interface (*l'eviu1,3-I* (*Image_Station_Click*)) pour ouvrir la fenêtre de propriétés de cette station (la figure 4.10). Mais après, il a dû fermer cette fenêtre par un clic sur son bouton *Annuler* (*eviu9,3-I* (*bouton_Annuler_Click*)). Ces manipulations inutiles ont été refaites encore une fois avant de réaliser cette tâche. Le sujet 8 a aussi effectué plusieurs actions inutiles avant de trouver le chemin pour réaliser cette tâche 7.

L'évaluateur peut interpréter que l'interface du SAI n'est pas très explicative, ne guide pas bien les sujets, d'où la raison possible de ces manipulations inutiles.

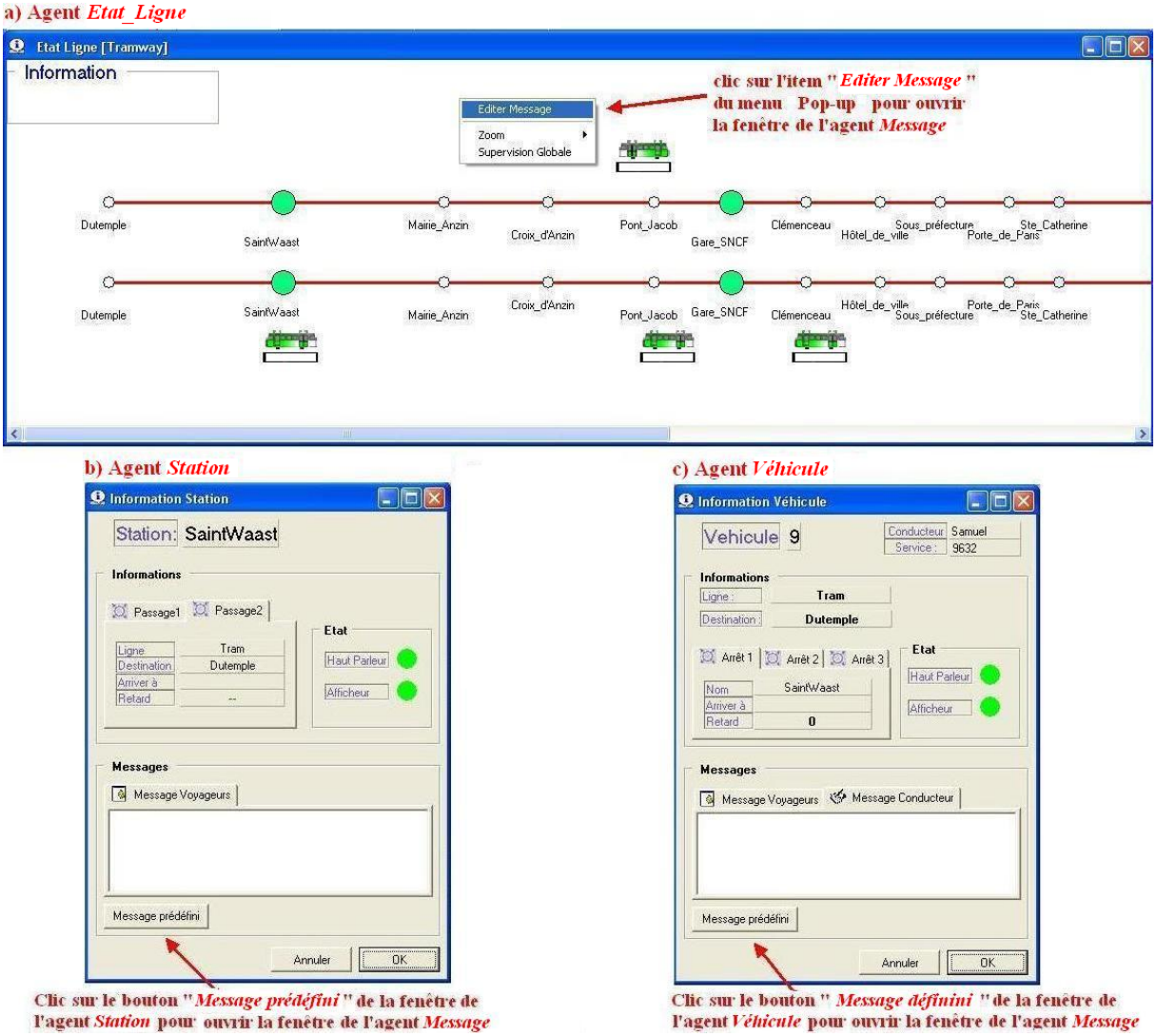

Figure 4.16 : Manière d'accéder à la fenêtre de propriétés de l'agent *interface Message* du SAI

B) L'évaluateur a pu constater aussi le problème concernant **le temps pris** par les sujets pendant l'expérimentation. Les sujets 3, 5, 6 ont eu besoin de 2 minutes environ pour commencer la première tâche ; donc, l'évaluateur peut interpréter que ces sujets ont pu éprouver des difficultés pour comprendre l'interface du SAI au début de leur session de travail. Tel problème devrait bien entendu ne plus être rencontré après la prise en main du système.

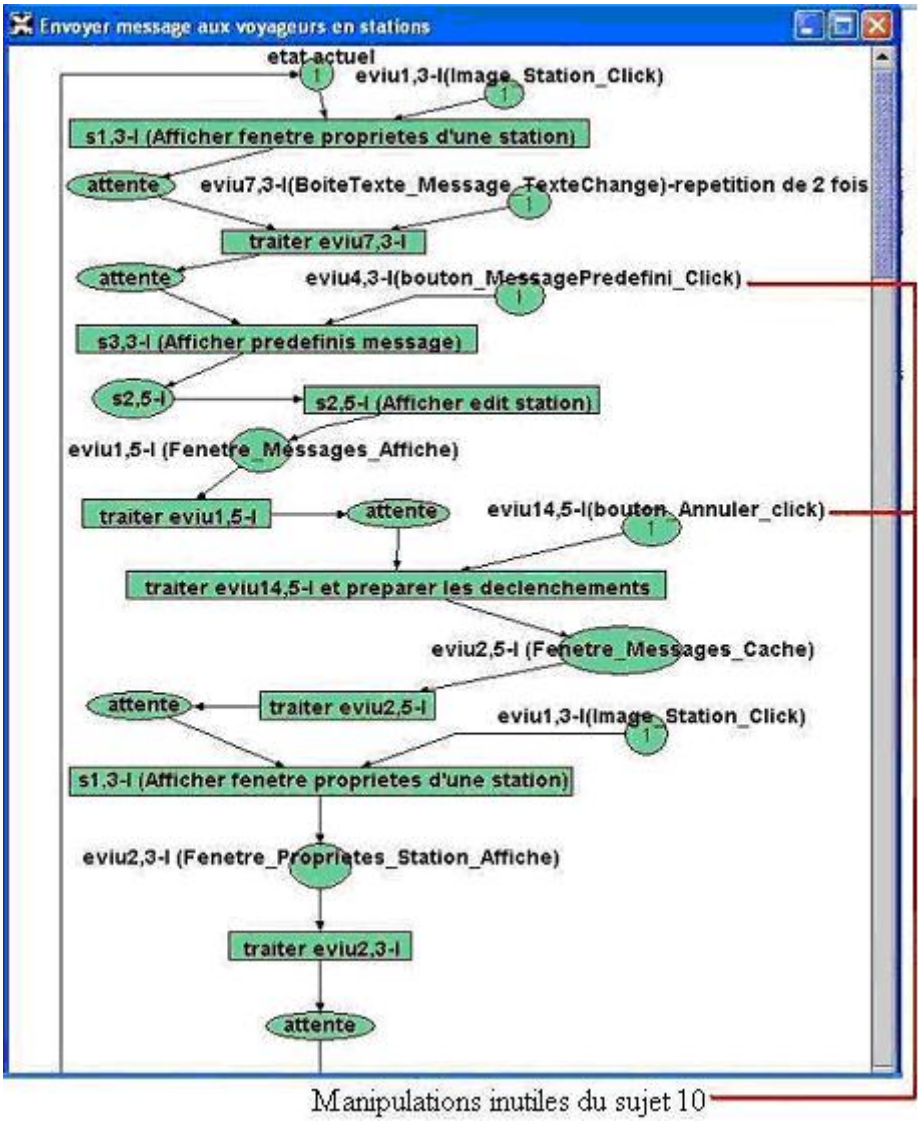

Figure 4.17 : Extrait de la capture d'écran du RdP généré pour le sujet 10 pendant sa réalisation de la tâche 1

C) Un autre problème réside dans **la réalisation des tâches** par les sujets pendant l'expérimentation. Ce problème concerne les réponses à ces questions : « *Quelles sont les tâches qui n'ont pas pu être réalisées ?* », « *Quelles sont celles qui ont pu l'être ?* » « *Les chemins choisis pour réaliser les tâches sont ils optimaux ?* »

Parmi les 9 tâches (cf. tableau 4.1) que les sujets ont dû réaliser pendant l'expérimentation, les 6 premières concernent l'envoi d'un message à un véhicule ou à une station et elles sont relativement simples. La tâche 7 concerne l'envoi d'un message à trois stations différentes d'une ligne et elle est plus complexe. La tâche 8 concerne l'envoi d'un message à toutes les stations de toutes les lignes et la 9 concerne l'envoi d'un message à tous les véhicules de toutes les lignes. Ces trois tâches sont les plus complexes et elles nécessitent l'ouverture de la fenêtre de propriétés de l'agent interface *Message* du SAI (cf.figure 4.7). Tous les sujets ont réussi à réaliser les 6 premières tâches, certains ont éprouvé des difficultés relativement aux trois dernières.

Avant de présenter les résultats à ce sujet, nous expliquons maintenant comment réaliser ces neuf tâches. Pour l'envoi d'un message aux stations, le concepteur a prévu trois chemins possibles. Il a aussi prévu trois chemins différents pour l'envoi d'un message aux véhicules. Il

est possible de choisir n'importe quel chemin parmi ceux possibles. Nous présentons maintenant ces trois chemins :

- **Le chemin (1)**. Pour envoyer un message à une station ou à un véhicule, on clique sur l'image de la *Station* concernée ou celle du *Véhicule* concerné de l'agent *Etat_Ligne* du SAI pour ouvrir la fenêtre de propriétés correspondante. Ces actions sont identifiées respectivement par *l'eviu1,3-I* (*Image_Station_Click*), *l'eviu1,4-I* (*Image_Vehicule_Click*) (déjà illustrées en figure 4.12). Sur la fenêtre de propriétés, on peut saisir le message puis cliquer sur le bouton *OK* pour envoyer celui-ci. C'est le chemin considéré comme optimal si on veut envoyer un message à une seule station ou à un seul véhicule.

- **Les chemins (2)** et **(3)**. Ils concernent l'ouverture de la fenêtre de l'agent *Message* du SAI (déjà illustrée en figure 4.7). Dans cette fenêtre, on peut saisir le message ou sélectionner un des messages disponibles. Ensuite, on sélectionne les stations ou/et les véhicules destinataires. Enfin, on clique sur le bouton *Envoyer* de cette fenêtre pour effectuer l'envoi. La diférence entre les chemins (2) et (3) concerne la manière d'ouvrir la fenêtre de l'agent *Message*.

    - **Dans le chemin (2)**, on ouvre celle-ci à travers la fenêtre de propriétés d'une station ou d'un véhicule quelconque. Comme l'illustrent les figures 4.16b et 4.16c, il faut seulement cliquer sur le bouton « *Message Prédéfini* » de la fenêtre en question, la fenêtre de l'agent *Message* s'ouvre en conséquence. Ces deux EVIUs sont identifiés respectivement par *l'eviu4,3-I* (*bouton_MessagePredefini_Click*) et *l'eviu10,4-I* (*bouton_MessagePredefini_Click*).

    - **Dans le chemin (3)**, on ouvre la fenêtre de l'agent *Message* à travers le menu Pop-up de l'agent *Etat_Ligne*. Comme l'illustre la figure 4.16a, il faut seulement cliquer sur l'item « *Editer Message* » du menu Pop-up, la fenêtre de l'agent *Message* du SAI s'ouvre en conséquence. Cet EVIU est identifié par *l'eviu3,2-I* (*PopUpMenuItem_Editer_Click*).

  On peut remarquer que le chemin 2 est plus long que le 3 parce que le 2 oblige à ouvrir la fenêtre de propriétés d'une station ou d'un véhicule quelconque avant. Si l'utilisateur veut envoyer un message à une seule station ou à un seul véhicule, alors le chemin 1 est optimal. Mais s'il veut envoyer un message à plusieurs stations et/ou à plusieurs véhicules, alors il faut choisir le chemin 2 ou le 3. Le chemin 3 est optimal dans ce cas parce qu'il permet d'ouvrir la fenêtre de l'agent *Message* plus rapidement.

Comme précisé ci-dessus, tous les sujets ont réussi à réaliser les 6 premières tâches. Bien qu'ils aient effectué des manipulations inutiles (déjà présentées ci-dessus dans les sections relatives aux deux critères : « *Incitation* » et « *Facilité de l'utilisateur à trouver les lignes, les stations ou les véhicules souhaités* »), ils ont réussi. Tous ces sujets ont choisi le chemin (1) pour réaliser ces six tâches. C'est le chemin optimal parce que ces six tâches concernent l'envoi d'un message à une seule station ou à un seul véhicule. Pourtant, quelques sujets n'ont pas pu réaliser les trois dernières tâche. Nous nous focalisons donc sur les réalisations des tâches 7, 8, et 9. Ces trois tâches concernent l'envoi de message à plusieurs stations et véhicules, donc le chemin (3) est le chemin optimal. Pendant les réalisations de ces trois tâches, les sujets ont effectué des manipulations inutiles déjà détaillées ci-dessus, donc nous ne les présentons pas dans le tableau 4.5 s'intéressant à tâche 7, ni dans le tableau 4.6 s'intéressant aux tâches 8 et 9.

Tableau 4.5 : Problèmes de réalisation de la tâche 7

Sujets	Réussite (R) ou Echec (E)	Problèmes détectés pendant la réalisation de la tâche 7 (l'envoi d'un message à trois stations d'une ligne)
1	R	Ce sujet a pu trouver le chemin optimal (3) pour réaliser cette tâche. Aucune manipulation inutile n'a été effectuée. La figure 4.18 illustre une partie de la capture d'écran du RdP généré pendant sa réalisation de cette tâche.
4	R	Ce sujet a pu aussi réaliser cette tâche mais il a suivi le chemin non optimal (2). En effet, il a ouvert la fenêtre de propriétés d'une station puis il a cliqué sur le bouton « *Message Prédéfini* » pour ouvrir la fenêtre permettant d'envoyer un message. Aucune manipulation inutile n'a été effectuée par ce sujet.
9	R	Ce sujet a pu aussi réaliser cette tâche 7 mais il a suivi le chemin (2) qui n'est pas optimal ; il a aussi effectué plusieurs actions inutiles avant de trouver ce chemin
6, 8	R	Ces sujets ont pu trouver le chemin optimal (3) pour réaliser cette tâche mais ils ont effectué plusieurs actions inutiles avant de trouver ce chemin.
2, 3, 5, 7, 10	R	Ces sujets n'ont pas pu trouver le chemin optimal (3) pour réaliser cette tâche. Ils ont suivi le chemin (1) qui leur permet d'envoyer un message à une seule station. En conséquence, ils ont dû réaliser ce chemin (1) trois fois consécutivement. Particulièrement, avant de choisir le chemin (1), les manipulations inutiles du sujet 7 montrent que celui-ci a été embarassé lors de la réalisation de cette tâche.

Tableau 4.6 : Problèmes de réalisation des tâches 8 et 9

Sujets	Réussite (R) ou Echec (E)	Problèmes détectés pendant la réalisation des tâches 8, 9 (l'envoi d'un message à toutes les stations et tous les véhicules de toutes les lignes)
3	T8 (R) et T9 (E)	Ce sujet a réussi la tâche 8 mais il a suivi le chemin non-optimal (2). En effet, il a ouvert la fenêtre de propriétés d'une station puis il a cliqué sur le bouton « *Message Prédéfini* » pour ouvrir la fenêtre de l'agent *Message* qui lui permet d'envoyer un message à toutes les stations de toutes les lignes. Aucune manipulation inutile n'a été effectuée. Mais il n'a pas pu réaliser la tâche 9. Il n'a pas pu re-ouvrir la fenêtre de l'agent *Message* pour réaliser cette tâche 9 bien qu'il ait pu le faire lors de la réalisation de la tâche 8. Ce qui est surprenant et doit être approfondi. La figure 4.19 illustre une partie de la capture d'écran du RdP généré pendant sa réalisation de la tâche 8
2, 5, 7, 10	E	Ces sujet n'ont pas pu trouver le chemin pour réaliser les tâches 8 et 9
1, 4, 6, 8	R	Ces sujets ont pu trouver le chemin optimal (3) pour réaliser les tâches 8, 9. Aucune manipulation inutile n'a été effectuée.
9	R	Ce sujet a pu aussi réaliser ces tâches mais il a suivi le chemin non-optimal (2). En effet, il a ouvert la fenêtre de propriétés d'une station puis il a cliqué sur le bouton « *Message Prédéfini* ». Aucune manipulation inutile n'a été effectuée.

Comme nous l'avons abordé ci-dessus, les deux chemins (2) et (3) concernent l'ouverture de la fenêtre de l'agent *interface Message* qui permet d'envoyer un message à plusieurs stations et/ou véhicules des lignes. A travers l'expérimentation, l'évaluateur a pu constater que plusieurs sujets n'ont pas pu trouver l'agent *Message* pour réaliser les tâches 7, 8, 9 et que quelques sujets ont pu le trouver en suivant le chemin non optimal (2). Donc, l'évaluateur a pu conclure que l'interface du SAI ne fournit aucun mécanisme ou aucun moyen faisant connaître aux sujets les différentes alternatives ou chemins pour réaliser les tâches complexes de type 7, 8, 9.

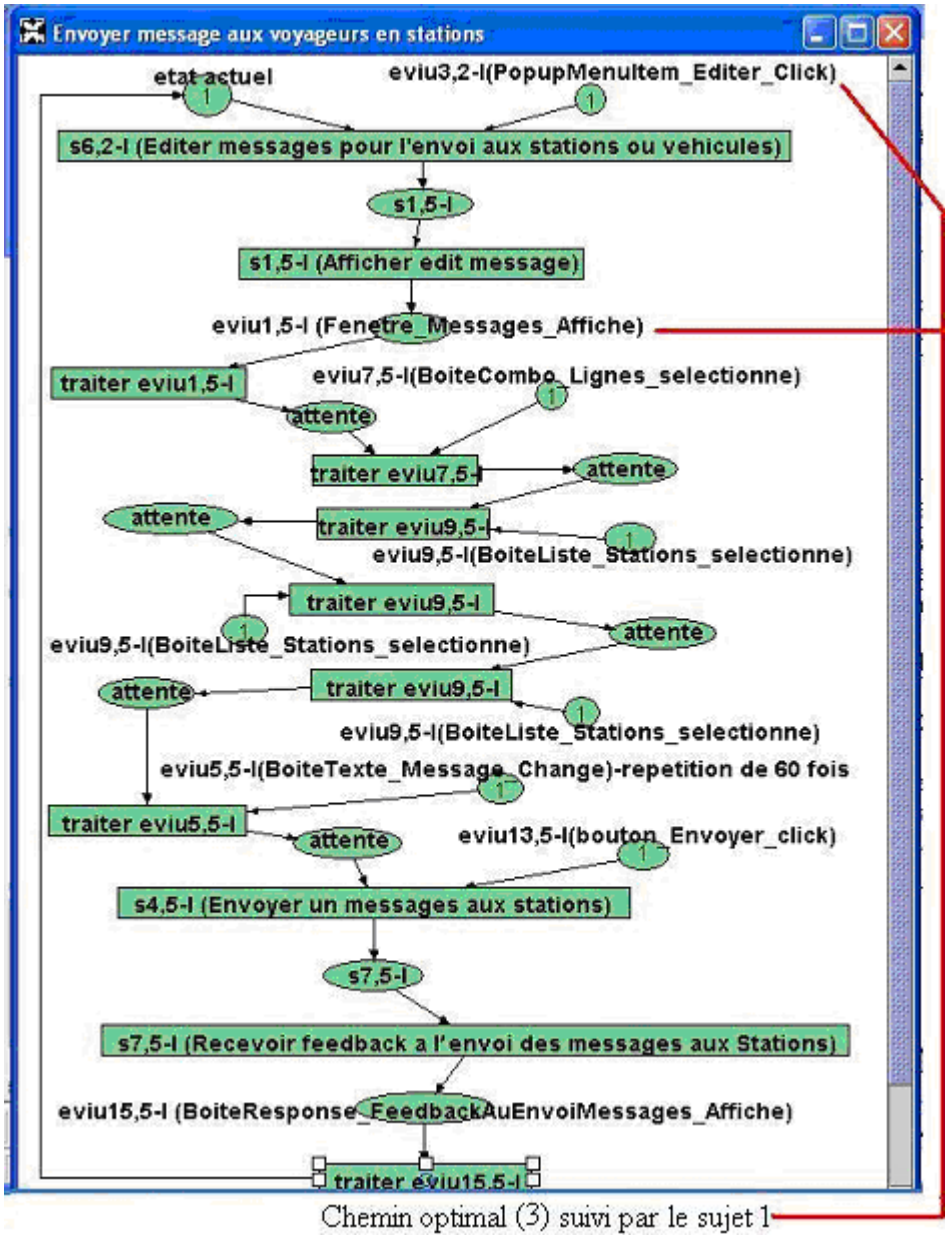

Figure 4.18 : Extrait de la capture d'écran du RdP généré pour le sujet 1 pendant sa réalisation de la tâche 7

A partir des critiques ci-dessus, l'évaluateur peut suggérer au concepteur d'améliorer l'agent *Etat_Ligne* du SAI comme l'illustre la figure 4.15. On peut, par exemple, ajouter à cet agent un bouton intitulé « *Envoyer un message aux stations et/ou aux véhicules* » comme l'illustre la figure 4.15b. Ce bouton permet à l'utilisateur d'ouvrir la fenêtre de l'agent *interface Message* après un clic sur ce bouton. Cette amélioration fournit une indication aidant à trouver facilement l'agent *Message* du SAI.

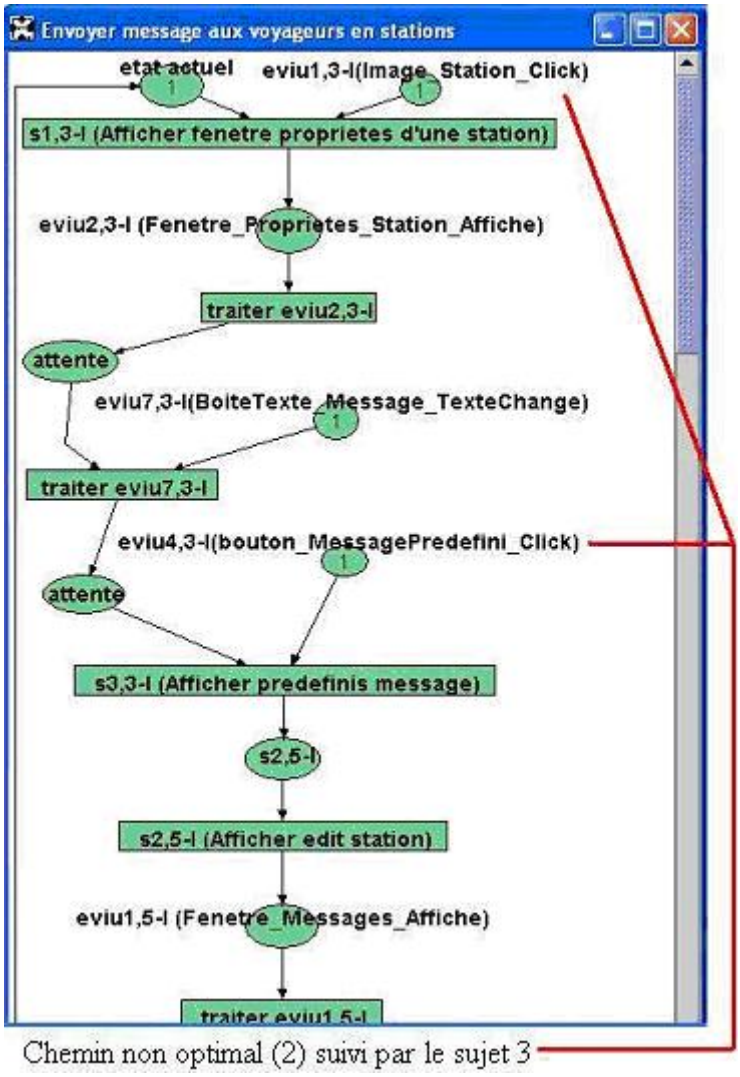

Figure 4.19 : Extrait de la capture d'écran du RdP généré pour le sujet 3 pendant sa réalisation de la tâche 8

D) L'évaluateur peut aussi constater un autre problème lié au ce critère *Incitation*. Il concerne **les messages disponibles** dans la fenêtre de l'agent *Message*. Il est possible de sélectionner un de ces messages pour l'envoyer aux stations et/ou aux véhicules. Il ne faut pas les saisir en utilisant le clavier. Par exemple, le message « *Joyeuses Fêtes* » que les sujets doivent envoyer lors de la réalisation de la tâche 9 est un des messages disponibles et sélectionnables dans la boîte de liste (Listbox) de la fenêtre de l'agent *interface Message*. Mais pendant l'expérimentation, aucun sujet capable de réaliser la tâche 9 (les sujets 1, 4, 6, 8, 9) ne l'a perçu à part le sujet 1. Ces quatre sujets (4, 6, 8, 9) ont re-saisi ce message "*Joyeuses Fêtes*". Donc, l'évaluateur peut conclure que l'interface de l'agent *interface Message* n'est pas assez explicite sur ce point. Cet inconvénient peut augmenter la durée de réalisation des tâches, notamment dans le cas de message long.

A partir des critiques ci-dessus, l'évaluateur peut suggérer au concepteur d'améliorer l'agent *interface Message* du SAI comme l'illustre la figure 4.20.

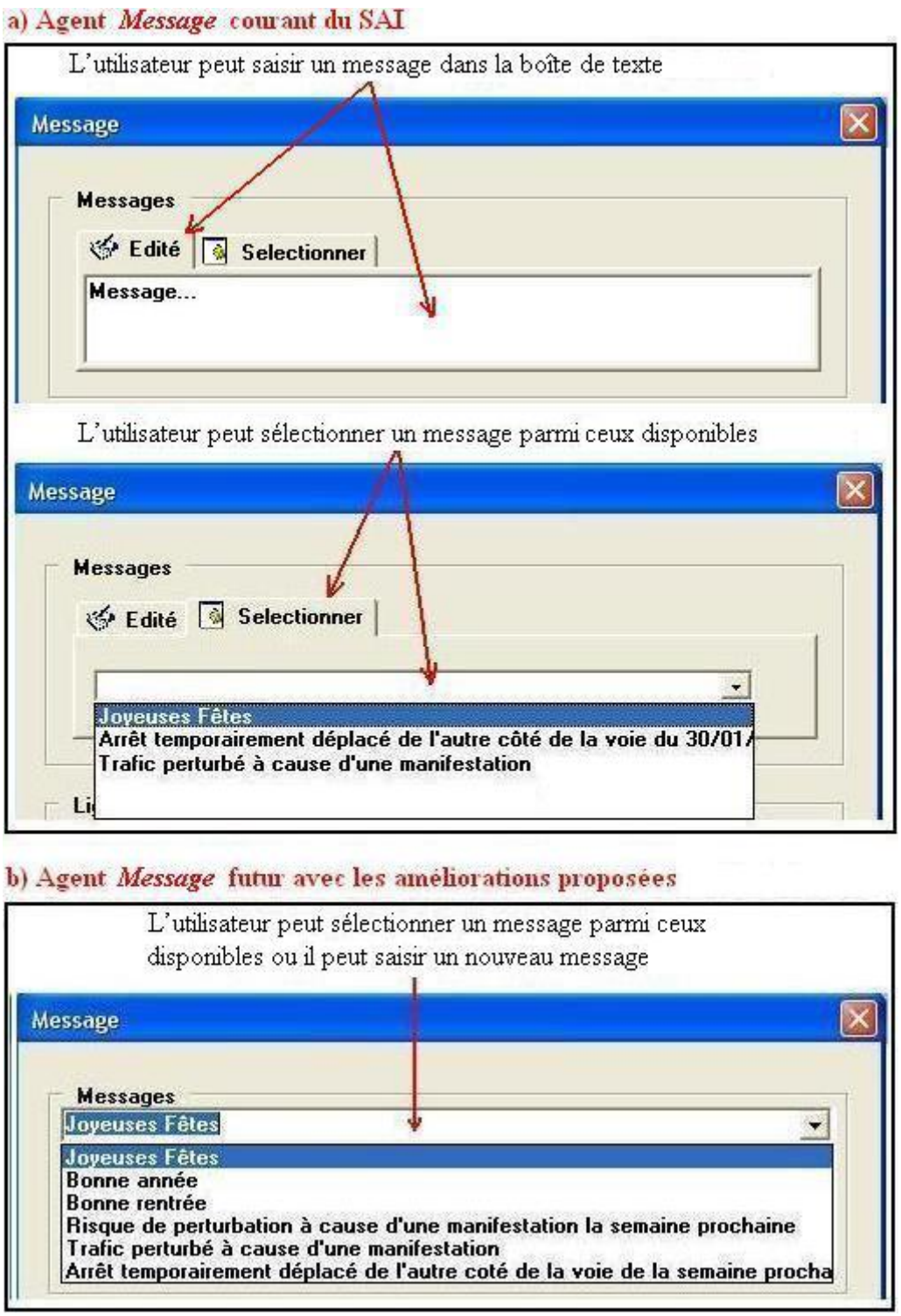

Figure 4.20 : Extrait de l'agent *Message*. La partie a illustre l'agent *Message* courant. La partie b illustre l'agent *Message* après amélioration.

La figure 4.20 montre une partie de la fenêtre de l'agent *interface Message* dans deux cas : sans amélioration et avec amélioration. Pour l'instant, l'agent *interface Message* utilise deux éléments interactifs : une boîte de texte de saisie d'un message et une boîte de liste des messages disponibles et sélectionnables. Il est possible de les remplacer par un seul élément interactif : une boîte de combo (ComboBox) contenant une liste de messages disponibles (cf. figure 4.20b) avec laquelle on peut sélectionner un message à envoyer ou en saisir un nouveau. Cet élement permet les deux actions. Il a aussi la fonctionnalité suivante : lorsque

l'utilisateur saisit les premiers caractères du message, les messages concernés sont affichés et deviennent sélectionnable. Dans ce cas, on ne doit plus continuer à saisir les caractères suivants s'il peut trouver le message attendu. Par exemple, quand on saisit le premier caractère « j » du message « *Joyeuses Fêtes* », alors tous les messages commençant par « j » comme « *Joyeux Noel* », « *Joyeuses fêtes de fin d'année* », etc., sont affichés et deviennent sélectionnables.

4.6. Critère : Complexité

Cet attribut de qualité est une propriété non fonctionnelle du système. Il est utilisé pour examiner le degré de complexité dans la conception du système [Lee et Hwang, 2004]. La complexité affecte la vitesse du système ; plus le système est complexe, plus le système risque de fonctionner lentement. En plus, un système dont la conception est complexe peut être plus difficile, pour les développeurs, à comprendre et gérer le code source.

4.6.1. Interprétation des résultats d'analyse pour une évaluation selon ce critère

L'évaluateur peut utiliser les mesures complémentaires calculées par le module 3 pour estimer ce critère. Le ratio entre *« nombre des paires de services interagissant »* et *« nombre de tâches réalisées" »* peut être utilisé pour évaluer le système selon ce critère. Ce ratio faire connaître à l'évaluateur le nombre moyen des interactions effectuées pour réaliser une tâche. S'il y a trop d'interactions, alors l'evaluateur peut considérer que la conception du système est complexe. Pour résumer, plus ce ratio est élevé, plus la conception du système peut être considéré comme complexe. Il faudrait que le concepteur du système diminue cette complexité en réorganisant les services des agents pour diminuer le nombre d'interactions entre eux dans la réalisation des tâches.

En cas de nécessité, afin d'évaluer la complexité de la réalisation de quelques tâches importantes, l'évaluateur peut aussi observer les RdPs générés de ces tâches pour connaître les interactions effectuées ; donc il peut évaluer leur complexité et proposer des améliorations éventuelles pour optimiser leur réalisation en diminuant le nombre d'interactions entre services.

4.6.2. Critiques et améliorations proposées par l'évaluateur en se basant sur ce critère

En étudiant ce ratio, l'évaluateur peut constater qu'une ou deux interactions en moyenne ont été effectuées pour réaliser une tâche (par exemple, dans le cas du sujet 1, seize interactions entre les services ont été effectuées pour réaliser les 9 tâches comme l'illustre la figure 4.21.). Ce ratio est acceptable.

L'évaluateur peut conclure que l'organisation des services du système SAI n' est pas très complexe puisque pour réaliser une tâche, le système ne doit pas effectuer beaucoup d'interactions entre les services des agents. Cette remarque n'est valable que dans le cas des tâches étudiées bien entendu.

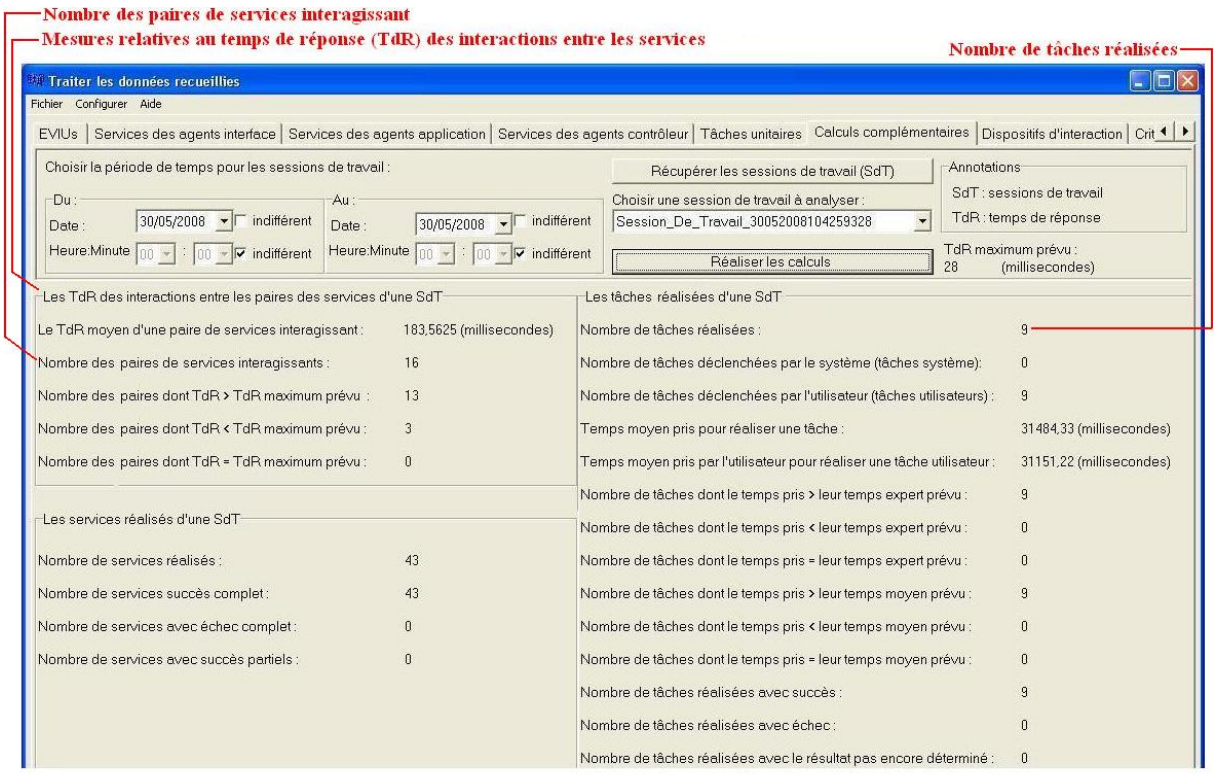

Figure 4.21 : Capture d'écran des calculs complémentaires pour le sujet 1

4.7. Critère : Fiabilité

Cet attribut de qualité est une propriété non fonctionnelle du système. Il y a plusieurs définitions de la fiabilité comme :

- "la capacité d'un système ou d'un composant à effectuer les fonctions requises sous les conditions déclarées pendant une période du temps spécifiée" [IEEE, 1990]
- "la capacité d'un équipement, d'une machine, ou d'un système d'exécuter systématiquement sa fonction ou mission (prévue ou requise), selon la demande et sans dégradation ou échec"[24]

4.7.1. Interprétation des résultats d'analyse pour une évaluation selon ce critère

Pour évaluer ce critère, l'évaluateur peut utiliser les mesures complémentaires du module 3 comme le ratio entre "*nombre des services réalisés avec succès*" et "*nombre des services réalisés*". Ce ratio est appelé le ratio de succès. Plus il est élevé, plus le fonctionnement du système peut être considéré comme fiable. Plus le nombre de services réalisés est élevé, plus l'exactitude de cette évaluation de fiabilité est élevée.

4.7.2. Critiques et améliorations proposées par l'évaluateur en se basant sur ce critère

Dans le cas du SAI, à travers l'expérimentation, le ratio du succès des services est maximal (100%). Par exemple, comme l'illustre la figure 4.21, tous les quarante trois services ont été réalisés avec succès pendant la réalisation des tâches du sujet 1. On peut en conclure que dans le cas des tâches étudiées, le fonctionnement du système SAI est fiable.

[24] Selon le Business Dictionary (http://www.businessdictionary.com/definition/reliability.html). Le texte anglais original est : " Ability of an equipment, machine, or system to consistently perform its intended or required function or mission, on demand and without degradation or failure ".

4.8. Critère : Temps de réponse

Cet attribut de qualité est une propriété non fonctionnelle du système. D'après [Lee et Hwang, 2004], un des attributs dynamiques les plus importants pour évaluer la performance d'un système multi-agents (SMA) est le temps de réponse. La performance réelle du système SMA doit être mesurée par le temps de réponse qui est le temps pour passer du service requis au service fourni.

4.8.1. Interprétation des résultats d'analyse pour une évaluation selon ce critère

Le module 3 peut effectuer quelques mesures concernant le temps de réponse (TdR) : *nombre d'interactions entre les services, temps de réponse (TdR) moyen des interactions entre les services, comparaison entre le TdR réel et le TdR théorique prévu par le concepteur, nombre d'interactions dont les temps de réponse sont plus/moins/aussi longs que les TdR théoriques*.

Afin d'évaluer ce critère, l'évaluateur peut utiliser ces mesures. Plus le nombre d'interactions entre les services est élevé, plus l'exactitude de cette évaluation est élevée.

4.8.2. Critiques et améliorations proposées par l'évaluateur en se basant sur ce critère :

Dans le cas du système SAI, à travers l'expérimentation, l'évaluateur a pu constater que le temps de réponse moyen entre les services interagissant était de 90-150 millisecondes, permettant ainsi de conclure que le SAI était rapide. Cela ne nous surprend pas dans la mesure où tous les agents de la version courante du SAI opèrent sur une même machine (une distribution sur différentes machines conduirait certainement à des résultats beaucoup moins bons).

Cependant, le temps de réponse de la plupart des services interagissant est supérieur à celui prévu par le concepteur qui est de 28 millisecondes ; par exemple, dans le cas du sujet 1, il n'y a que trois parmi seize interactions entres services inférieures au TdR prévu (cf. figure 4.21), c'est-à-dire que la vitesse du système n'a pas été aussi élevée que le concepteur l'avait souhaitée.

4.9. Performance des utilisateurs

L'évaluateur est en mesure de formuler des remarques sur la performance des utilisateurs lors de l'utilisation du système et de comparer celles-ci, par exemple dans le but d'organiser des séances d'entrainement.

4.9.1. Interprétation des résultats d'analyse pour une évaluation selon ce critère

L'évaluateur peut utiliser les RdPs générés et les résultats d'analyse du module 3 pour évaluer ce critère de la façon suivante :

- Le nombre des tâches réalisées peut être utilisé pour évaluer ce critère. Un utilisateur exprimenté a tendance à pouvoir réaliser plus de tâches, tandis qu'un utilisateur novice peut éprouver différentes difficultés pour réaliser les tâches.
- Les chemins choisis pour réaliser les tâches parmi plusieurs chemins possibles prévus par le concepteur peuvent être aussi utilisés. Un utilisateur expérimenté a tendance à choisir un chemin optimal parmi plusieurs chemins possibles pour réaliser une tâche.
- Les manipulations inutiles, les actions et navigations erronées sont aussi exploitable. Un utilisateur expérimenté a tendance à en effectuer de moins en moins.

- Les durées de réalisation des tâches peuvent être utiles aussi (cf. mesure « *Temps moyen que l'utilisateur a pris pour réaliser une tâche* »). Un utilisateur expérimenté a tendance à réaliser les tâches rapidement.
- Le nombre d'apparitions des événements correspondant aux consultations de l'aide peut être utilisé pour évaluer ce critère. Dans le cas d'un utilisateur expérimenté, ces événements peuvent se produire rarement.

4.9.2. Critiques et améliorations proposées par l'évaluateur en se basant sur ce critère

Comme déjà présenté dans la section relative au critère *Incitation* l'évaluateur a pu constater que tous les sujets ont pu réaliser les six premières tâches d'envoi de message à une seule station et à un seul véhicule (avec plusieurs manipulations inutiles, actions et navigations erronées), mais que quelques sujets n'ont pas pu réaliser les trois dernières tâches relatives à l'envoi du message à plusieurs stations et véhicules. Donc, la réalisation de ces tâches 7, 8 et 9 est bien le critère utilisé par l'évaluateur pour examiner la capacité du sujet. Ainsi, il peut classifier les sujets en deux groupes :

A) **Le groupe 1**. Ce groupe concerne les sujets qui ont pu réaliser toutes les tâches, y compris les tâches complexes 7 (envoi d'un message aux trois stations d'une ligne), 8 (envoi d'un message à toutes les stations de toutes les lignes), 9 (envoi d'un message à tous les véhicules de toutes les lignes). Ce groupe 1 contient les sujets 1, 4, 6, 8, 9. Il y a trois chemins possibles pour réaliser ces tâches complexes et le chemin (3) est le chemin optimal (déjà abordé dans la section du critère *Incitation*). Les chemins (1) et (2) ne sont pas optimaux. Le tableau 4.7 se focalise sur les sujets de ce groupe.

Tableau 4.7 : Classement des sujets dans le groupe 1

<table>
<tr><th>Sujet</th><th>Chemin suivi pour réaliser la tâche 7 (chemin optimal (3) ou non). Existence ou non d'actions inutiles</th><th>Chemin suivi pour réaliser les tâches 8, 9 (chemin optimal (3) ou non). Existence ou non d'actions inutiles</th><th>Message "Joyeuses Fête». Perçu ou non</th><th>Temps moyen de réalisation d'une tâche (en secondes)</th><th>Classement de ce sujet dans le groupe 1</th></tr>
<tr><td>4</td><td>Chemin non-optimal (2). Aucune action inutile</td><td>Chemin optimal (3). Aucune action inutile.</td><td rowspan="4">Non. Ces sujets ont dû les re-saisir lors de la réalisation de la tâche 9</td><td>60,1</td><td>4/5</td></tr>
<tr><td>8</td><td rowspan="2">Chemin optimal (3). Plusieurs actions inutiles (avant de trouver ce chemin)</td><td rowspan="2">Chemin optimal (3). Aucune action inutile</td><td>40,5</td><td>3/5</td></tr>
<tr><td>6</td><td>27,7</td><td>2/5</td></tr>
<tr><td>9</td><td>Chemin non-optimal (2). Plusieurs actions inutiles (avant de trouver ce chemin)</td><td>Chemin non-optimal (2). Aucune action inutile</td><td>38,8</td><td>5/5</td></tr>
<tr><td>1</td><td colspan="2">Chemin optimal (3). Aucune action inutile (cf. figure 4.18)</td><td>Oui. Il l'a sélectionné</td><td>31,1 (cf. figure 4.21)</td><td>1/5</td></tr>
</table>

B) **Le groupe 2**. Ce groupe concerne les sujets restants (2, 3, 5, 7,10) qui n'ont pas pu réaliser toutes les tâches 7, 8, 9. Les sujets du groupe 2 sont estimés moins performants que ceux du groupe 1. Dans ce groupe , un sujet a suivi le chemin non-optimal (2) pour réaliser la tâche 8 et les autres sujets ont suivi le chemin le plus mauvais (1) pour réaliser la tâche 7 qui concerne l'envoi d'un message à trois stations d'une ligne. Les réalisations des tâches de ce groupe sont décrites dans le tableau 4.8.

Tableau 4.8 : Classement des sujets dans le groupe 2

Sujet	Chemin suivi pour réaliser la tâche 7. Existence ou non d'actions inutiles.	Réalisation des tâches 8, 9. Existence ou non d'actions inutiles	Temps moyen de réalisation d'une tâche (secondes)	Classement de ce sujet dans le groupe 2
3	Chemin le plus long (1). Ils ont dû réaliser ce chemin (1) trois fois consécutivement. Particulièrement, avant de choisir ce chemin (1), le sujet 7 a effectué des actions inutiles.	Chemin non-optimal (2) pour réaliser la tâche 8 (cf. figure 4.19). Ce sujet n'a pas pu réaliser la tâche 9. Aucune action inutile n'a été faite	52,0	1/5 - le seul sujet qui a pu réaliser la tâche 8
2		Ces quatre sujets n'ont pas pu réaliser les tâches 8, 9	68,6	5/5
5			35,6	4/5
7			41,0	3/5
10			37,7	2/5

Nous présentons maintenant quelques remarques supplémentaires relativement au classement des sujets de ce groupe 2.

- Lors de la réalisation de la tâche 7, ces cinq sujets (2, 3, 5, 7, 10) ont suivi le chemin (1) qui leur permet d'envoyer un message à une seule station. Par conséquent, ils ont dû réaliser ce chemin (1) trois fois consécutivement. L'évaluateur a pu constater que les temps pris pour les deuxième et troisième fois sont bien plus courts que pour la première et que l'apparition de *l'eviu7,3-I (BoiteTexte_Message_Texte_changé)* a été plus rare. Cet EVIU correspond à la saisie du message des sujets. L'évaluateur peut deviner que ces sujets ont saisi le message la première fois mais ils ont utilisé la fonctionnalité copier/coller pour les deuxième et troisième fois pour éviter de re-saisir ce message.
- Le sujet 5 n'a pas traité une perturbation concernant un retard d'un véhicule. Deux perturbations sont apparues, la dexième est apparue 8 secondes plus tard que la première et ce sujet a omis le traitement de la première perturbation.

C) Conclusion sur le classement des sujets :
Le classement global des dix sujets est présenté dans le tableau 4.9 ci-dessous :

Tableau 4.9 : Classement global des sujets

Sujet	Classsement global
1	1/10
6	2/10
8	3/10
4	4/10
9	5/10
3	6/10 (le meilleur du groupe 2)
10	7/10
7	8/10
5	9/10
2	10/10

4.10. Fréquence d'apparition des événements

Ce critère vise à répondre aux questions : « *quels sont les événements qui sont apparus fréquemment ?* », « *quels sont ceux qui sont apparus rarement ?* », « *quels sont ceux qui ne sont jamais apparus ?* ». Les réponses à ces question sont utiles pour l'évaluation du système interactif.

4.10.1. Interprétation des résultats d'analyse pour une évaluation selon ce critère

L'évaluateur utilise les résultats d'analyse du module 3 pour connaître la fréquence d'apparition des événements (EVIUs, services, tâches). A partir de ces fréquences, l'evaluateur peut tirer des commentaires concernant le système et les utilisateurs et il peut donner des conseils utiles au concepteur pour l'amélioration du système.

- Fréquence d'exécutions des services : si quelques services sont souvent exécutés, alors il est conseillé au concepteur d'améliorer le temps d'exécution de ces services, contribuant ainsi à celui du système global. Il faut optimiser ces services (réduire le temps d'exécution, la mémoire consommée, optimiser le code d'exécution de ces services, etc.).
- Fréquence de réalisation des tâches. Dans le chapitre précédent, nous avons distingué deux types de tâches : les *tâches systèmes* (déclenchées par le système) et les *tâches utilisateurs* (déclenchées par l'utilisateur)
    - Si quelques *tâches systèmes* sont souvent réalisées, alors on suggère au concepteur de chercher à accélérer leur temps de réalisation. Pour atteindre cet objectif, il faut accélérer le temps d'exécution des services associés ainsi que les interactions entre ces services (temps de réponse).
    - Si quelques *tâches utilisateurs* sont souvent réalisées, alors on suggère au concepteur de faciliter leur réalisation. Pour atteindre cet objectif, le concepteur est invité à améliorer les vues, les fenêtres associées (par exemple, le concepteur peut ajouter différentes façons de réaliser ces tâches, à travers les raccourcis (shortcuts) et/ou les touches rapides (speed keys), il faut permettre aux utilisateurs de commencer à réaliser ces tâches n'importe où dans le système, etc.). En conséquence, la productivité ainsi que la satisfaction des utilisateurs du système peuvent en être augmentées.
- Fréquence d'apparition des EVIUs :
    - Si un EVIU est souvent déclenché par l'utilisateur, alors le concepteur peut être sollicité pour fournir à l'utilisateur plusieurs manières pour déclencher cet EVIU (souris, touches rapides).
    - La fréquence d'apparition des EVIUs peut faire savoir à l'évaluateur si l'apparence courante de l'interface est efficace.
        - Si quelques EVIUs d'une même fenêtre se produisent souvent, alors leurs widgets associés doivent être placés de telle façon qu'ils soient proches les uns des autres sur l'interface ou qu'ils soient regroupés dans un même groupe sur l'interface. Cette organisation permet de réduire les mouvements de la souris et l'interface devient plus efficace [Sears, 1995].
        - Si quelques EVIUs ne se produisent jamais ou s'ils se produisent rarement, alors leurs widgets associés pourraient être supprimés ou cachés. Par exemple, quand on veut imprimer un document, quelques options concernant les papiers, texte peuvent être cachées dans une boîte d'option séparée [Sears, 1995] parce que ces options sont rarement utilisées par l'utilisateur.
- Le nombre d'apparitions des événements correspondant aux consultations de l'aide peut être utilisé pour évaluer l'interface et même les performances de l'utilisateur.

4.10.2. Critiques et améliorations proposées par l'évaluateur en se basant sur ce critère

A travers l'expérimentation, l'évaluateur a pu percevoir que la fréquence d'apparition des EVIUs *Zoom avant*, *Zoom Arrière* de l' agent *interface Etat_trafic* sont nulles (cf. figure 4.10). Ces événements permettent aux utilisateurs de changer la taille de vues et ils ne se sont jamais produits pendant l'expérimentation.

En conséquence, l'évaluateur peut suggérer au concepteur d'éliminer deux boutons « *Zoom avant* », « *Zoom arrière* » de l'agent *interface Etat_Trafic* du SAI (Ces deux boutons sont les éléments interactifs associés à ces EVIUs). En conséquence, si l'utilisateur veut faire « *Zoom Avant* » ou « *Zoom Arrière* » à l'avenir, il lui reste une seule manière pour le faire : cliquer sur les items concernés dans le menu Pop-up de l'agent *interface Etat_Ligne*. Cette élimination permet au concepteur d'augmenter la taille du cadre de choix des « *checkboxes* » des lignes de l'agent *interface Etat_Trafic* et d'augmenter aussi la distance entre ces « *checkboxes* » des lignes de ce cadre comme l'illustre la figure 4.22.

Figure 4.22 : Agent *Etat_Trafic* du SAI. La partie a illustre l'agent *Etat_Trafic* courant. La partie b illustre l'agent *Etat_Trafic* après amélioration

Grâce à ces augmentations, l'utilisateur peut être certain de ne pas choisir une mauvaise ligne. Comme nous l'avons déjà abordé dans la section du critère d'évaluation « *Facilité de l'utilisateur à trouver les lignes, stations les véhicules souhaités* », quelques sujets ont choisi la mauvaise ligne ; par exemple, les sujets 2, 9 ont cliqué plusieurs fois sur les différents « *checkboxs* » de l'agent *Etat_Trafic* pour chercher la ligne concernée lorsqu'ils ont réalisé la tâche 5 (envoi d'un message à une station). En effet, *l'eviu4,1-I* (*checkbox_ligne_click*) s'est produit plusieurs fois avant de trouver la ligne concernée.

La raison de ce mauvais choix est dûe à l'espace très restreint entre les « *checkboxs* » des lignes de l'agent *interface Etat_Trafic* courant ; en conséquence, le clic de l'utilisateur sur le « *checkbox* » peut être erroné. Cette amélioration fournit aux utilisateurs une vue plus claire des « *checkboxs* » des lignes et diminue la possibilité de procéder à une mauvaise sélection.

4.11. Facilité et rapidité de l'utilisateur à traiter les perturbations reçues des véhicules

C'est un critère métier spécifique au système SAI. Selon ce critère, on évalue si le SAI permet à l'utilisateur de traiter rapidement les perturbations, notamment dans les cas où les intervalles de temps entre les apparitions des perturbations sont courts, c'est-à-dire que les perturbations apparaissent consécutivement ou presque simultanément.

4.11.1. Interprétation des résultats d'analyse pour une évaluation selon ce critère

L'évaluateur peut utiliser les résultats d'analyse du module 3 pour prendre en compte ce critère :

- L'évaluateur peut déterminer si l'utilisateur a omis le traitement des perturbations. Si le nombre des perturbations omises est élevé, alors des mesures sont à prendre.
- L'intervalle de temps entre l'apparition d'une perturbation et l'apparition d'un événement correspondant à son traitement peut aussi être utilisé pour évaluer ce critère. Si cet intervalle est long, il faut ainsi en tirer les conséquences.
- Si l'utilisateur a effectué plusieurs manipulations inutiles avant de traiter une perturbation, alors l'évaluateur peut interpréter que l'utilisateur a connu des difficultés pour la traiter.

4.11.2. Critiques et améliorations proposées par l'évaluateur en se basant sur ce critère

Le traitement des perturbations a impliqué l'envoi de messages nécessaires aux véhicules et/ou aux stations concernées. Pendant l'expérimentation, chaque sujet doit traiter plusieurs perturbations (de 3 à 6 perturbations). En général, les intervalles de temps entre les apparitions des perturbations sont relativement longs (d'une à trois minutes), donc les sujets n'ont omis aucune perturbation.

Il y a un seul cas où l'intervalle de temps entre deux apparitions de perturbations est court : pendant la session de l'expérimentation du sujet 5, cet intervalle est seulement de 8 secondes et ce sujet a omis le traitement de la première perturbation.

En conséquence, l'évaluateur peut supposer que si les perturbations étaient apparues consécutivement ou presque simultanément pendant l'expérimentation, alors elles auraient pu être omises par les sujets.

Ce critère n'est pas bien évalué dans le SAI courant parce que le SAI oblige l'utilisateur à mémoriser exactement tous les véhicules perturbés afin de chercher manuellement chaque véhicule perturbé sur son interface. Après l'avoir trouvé, il faut ouvrir manuellement la fenêtre des propriétés de ce véhicule pour lui envoyer un message.

A partir de ces critiques, l'évaluateur peut proposer l'amélioration suivante qui concerne la modification et l'ajout de nouvelles fonctionnalités au SAI pour faciliter aux utilisateurs le traitement des perturbations : dès que le SAI recoit une perturbation d'un véhicule, il sauvegarde cette perturbation dans une queue. S'il y a plusieurs pertubations qui arrivent au SAI en même temps, toutes ces perturbatitons sont aussi sauvegardées. Chaque perturbation sera retirée de la queue pour la traiter. Pour chaque perturbation retirée de la queue, le SAI la traite de cette manière :

- une fenêtre d'avertisement de cette perturbation va apparaître qui demande à l'utilisateur s'il veut envoyer un message au véhicule perturbé pour traiter cette perturbation.
- Si l'utilisateur veut lui envoyer un message, alors un service associé de l'agent *interface Véhicule* va être déclenché pour afficher la fenêtre des propriétés du véhicule perturbé et l'utilisateur peut envoyer le message à ce véhicule à travers cette fenêtre.

Cette proposition peut s'avérer très utile, notamment dans les cas où les perturbations apparaissent consécutivement ou presque simultanément. En effet , dans ces cas, il est souvent difficile, voire impossible de mémoriser exactement tous les véhicules perturbés afin de chercher manuellement chaque véhicule perturbé sur l'interface du SAI et d'ouvrir manuellement sa fenêtre de propriétés permettant de lui envoyer un message. Avec cette amélioration, il n'est plus possible d'omettre une perturbation (comme cela avait été le cas du sujet 5).

5. Conclusion

Dans ce chapitre, nous avons d'abord présenté la structure du projet SART (Système d'Aide à la Régulation du Trafic) qui s'est inscrit dans le cadre du programme technologies avancées pour les transports du Groupement Régional Nord-Pas de Calais pour la Recherche dans les Transports (G.R.R.T) et qui a pour objectif de réaliser un système aidant les régulateurs en salle de contrôle à mieux accomplir leur tâche en mode normal et mode dégradé de fonctionnement du réseau du trafic. Ensuite, le système interactif à base d'agents SAI, un des trois sous-systèmes du projet concerné a été détaillé. Notons que ce thème s'est poursuivi dans le cadre du projet CISIT ISART.

Une expérimentation qui a été réalisée au sein de notre laboratoire LAMIH avec dix sujets a été présentée dans ce chapitre. Pendant cette expérimentation, l'environnement d'évaluation EISEval (Environment for Interactive System Evaluation) a été appliqué pour évaluer le système SAI. La démarche d'évaluation d'un système interactif en utilisant cet environnement a été présentée et illustrée. Enfin, nous avons détaillé et illustré les résultats d'évaluation (qui concernent les critiques du SAI et améliorations proposées) en se basant sur les critères du module 6. Dans le cas de cette expérimentation, nous avons joué le rôle de l'évaluateur. L'évaluateur peut utiliser l'environnement d'évaluation EISEval pour produire un document imprimable contenant ces critiques et améliorations par un simple clic sur le bouton « *Voir sous forme textuelle* » de l'interface du module 6 (cf. figure 3.24, section 3.5, chapitre 3).

Cette expérimentation nous a aidés à détecter les problèmes du SAI et à tester EISEval. Ainsi, EISEval possède les avantages présentés dans le chapitre 3 et il a aidé à identifier les problèmes de chaque agent du SAI afin que l'évaluateur puisse proposer des améliorations associées. Cependant, EISEval possède aussi des inconvénients, et nécessite d'être amélioré. Dans le chapitre suivant, nous allons présenter ces inconvénients et proposer de futures améliorations prenant la forme de perspectives de recherche.

Chapitre 5 : Perspectives de recherche

Sommaire

1. Introduction

Dans les chapitres précédents de ce mémoire, après avoir dressé un état de l'art des modèles d'architecture et des méthodes d'évaluation des systèmes interactifs, l'environnement d'évaluation EISEVal (Environment for Interactive System Evaluation) proposé et dévelopé a été présenté.

Bien que cet environnement d'évaluation puisse étendre les possibilités des mouchards électroniques traditionnels et fournir quelques fonctionnalités originales par rapport à ces mouchards traditionnels, il présente encore un ensemble de défauts et inconvénients. Dans ce chapitre, nous déterminons quelques pistes de recherche futures qui visent à perfectionner l'environnement d'évaluation courant, augmenter son automatisation et l'intégrer dans un environnement plus global, qu'on pourrait qualifier d'intégré, d'aide à l'évaluation des systèmes interactifs.

2. Perspectives relatives au système SAI

Comme perspectives concernant le SAI, quelques futurs travaux sont proposés :

- Il s'agirait d'abord d'améliorer le SAI en appliquant les suggestions proposées suite au traitement des résultats de l'analyse des données de l'expérimentation (cf. chapitre 4). Ces propositions ont été déjà présentées dans le chapitre précédent. L'amélioration de l'agent *interface Etat_Ligne* aurait pour objectif de fournir une indication aidant les utilisateurs à trouver facilement l'agent *Message* afin de réaliser les tâches complexes d'envoi de message à plusieurs véhicules et/ou stations à traver le chemin optimal. Cette amélioration permettrait aussi aux utilisateurs de trouver plus rapidement des lignes, des stations ou des véhicules nécessaires. L'amélioration de l'agent *interface Message* aiderait les utilisateurs à percevoir les messages disponibles, donc ils pourraient les sélectionner au lieu de les re-saisir, ce que les sujets ont effectué pendant l'expérimentation. L'agent *interface Etat_Trafic* devrait être amélioré pour obtenir une vue plus claire qui pourrait diminuer la possibilité de sélectionner une mauvaise ligne. Le mécanisme de traitement des perturbations reçues serait modifié pour faciliter aux utilisateurs les tâches à ce sujet. Cette amélioration pourrait assurer qu'aucune perturbation ne soit omise.
- Pour l'instant, les dix sujets ayant participé à l'expérimentation sont des ingénieurs et étudiants doctorants en Informatique ou Automatique. Bien qu'ils soient habitués à des logiciels informatiques, ce ne sont pas les professionnels en régulation de réseau de transport public. Dans l'avenir, il faudrait effectuer une deuxième expérimentation avec des utilisateurs expérimentés en régulation. Nous pensons qu'il serait ainsi possible de détecter plus de problèmes relatifs au SAI. En effet, si le SAI est utilisé par de tels utilisateurs dans leur contexte habituel de travail, alors il serait possible de découvrir des fonctionnalités manquantes et/ou des inconvénients liées aux fonctionnalités courantes. Par exemple, pendant l'expérimentation, une perturbation n'a pas été perçue (cf. 4.11, chapitre 4). Avec de vrais opérateurs, il serait possible d'analyser finement de tels problèmes et d'envisager avec eux des améliorations à apporter.

3. Perspectives relatives aux modules de l'environnement EISEval

En termes de perspectives concernant l'environnement EISEval, les travaux suivants pourraient être envisagés :

- Le module 1 de l'environnement d'évaluation EISEval est chargé de capturer les données correspondant aux événements se produisant dans le système interactif à évaluer. Il est

développé comme un système complètement indépendant des autres modules. Le lien entre le module 1 et les autres modules est une base de données communes. Il faudrait améliorer ce module selon les deux pistes suivantes :

- Pour l'instant, le lien entre le système interactif à évaluer et le module 1 est réalisé à travers le mécanisme de socket. Le module 1 capture chaque événement dès son apparition dans le système interactif. Cette méthode de capture des événements fonctionne efficacement dans l'environnement de réseau local de grand débit. Cependant, si le module 1 et le système interactif communiquent à travers un grand réseau de faible débit, par exemple, Internet, alors cette méthode de capture des événements n'est pas bonne. En effet, dans un tel environnement, le transfert continu (ou presque continu) de grandes quantités de données (provenant du système interactif et acquises par le module 1) peut ralentir le réseau ; le réseau peut même être embouteillé. Donc, il faudrait améliorer ce module afin que sa méthode de capture des événements puisse fonctionner correctement dans les cas où le débit du réseau n'est pas important.
- Pour l'instant, ce module vise à capturer les événements des applications fonctionnant sur des ordinateurs de type PC. Si on veut appliquer cet environnement pour évaluer d'autres types d'applications (applications Web, Mobile, etc.), alors il faut améliorer ce module afin qu'il puisse capturer les événements et stocker les événements capturés dans la base de données qui est exploitée par les autres modules.

- Les modules 4-5 de l'environnement d'évaluation EISEval génèrent les RdPs qui reconstituent les processus des activités réelles de l'utilisateur et du système pour réaliser les tâches. Suite à l'expérimentation, nous avons constaté que ces RdPs générés sont souvent très complexes parce que la plupart des sujets ont procédé à plusieurs manipulations inutiles et/ou actions erronées pendant l'expérimentation. La figure 5.1 illustre une partie du RdP complexe du sujet 6. Ce RdP correspond à un envoi d'un message à une station.

  Le sujet 6 a voulu envoyer un message à une station mais il a effectué des actions erronées. A travers le RdP illustré en figure 5.1, l'évaluateur peut savoir que ce sujet a cliqué sur l'image d'un véhicule au lieu de la station concernée (*l'eviu1,4-I (Image_Vehicule_Click)*) pour ouvrir sa fenêtre de propriétés. En conséquence, il a dû fermer la fenêtre affichée de ce véhicule en cliquant sur son bouton *Annuler* (*eviu16,4-I (bouton_Annuler_Click)*). Enfin, il a cliqué sur l'image de la station concernée (*l'eviu1,3-I (Image_Station_Click)*) à laquelle il devait envoyer un message, pour ouvrir sa fenêtre de propriétés et ils ont continué à effectuer les autres manipulations pour l'envoi d'un message.

  Pour l'instant, suite à la génération des RdPs du module 4, le module 5 permet d'afficher les RdPs générés et/ou les RdPs prévus par le concepteur afin que l'évaluateur puisse les confronter de lui-même. Par conséquent, l'évaluateur a été surchargé par ces RdPs (environ 130 RdPs générés pour l'expérimentation réalisée). Dans l'avenir, il faudrait améliorer le module 5 pour aider l'évaluateur à analyser ces RdPs générés. Cette amélioration a pour objectif de détecter les actions erronées, les manipulations utiles provenant de l'utilisateur (sous forme des transitions et des états redondants). Une telle amélioration faciliterait à l'évaluateur la réalisation de ses tâches. La figure 5.2 illustre notre vision sur le futur module 5.

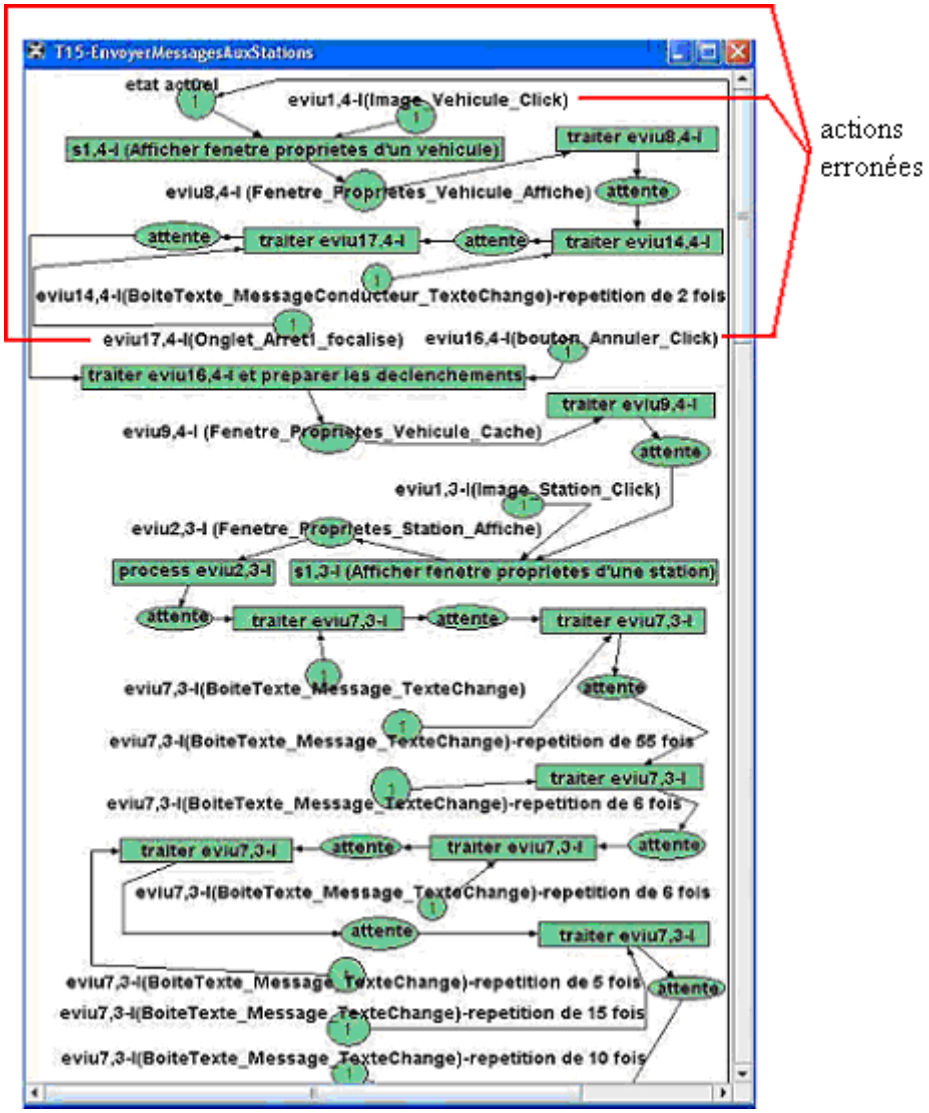

Figure 5.1 : Exemple de RdP généré (du sujet 6) qui est très complexe

Par exemple, suite à l'analyse du RdP généré du sujet 6 pour effectuer un envoi de message à une station (cf. figure 5.1), le futur module 5 produirait une liste contenant les EVIUs : *eviu1,4-I (Image_Vehicule_Click)*, *eviu17,4-I (Onglet_Arret1_Focalise)*, *eviu16,4-I (bouton_Annuler_Click)*. Ces EVIUs correspondraient aux actions erronées de l'utilisateur.

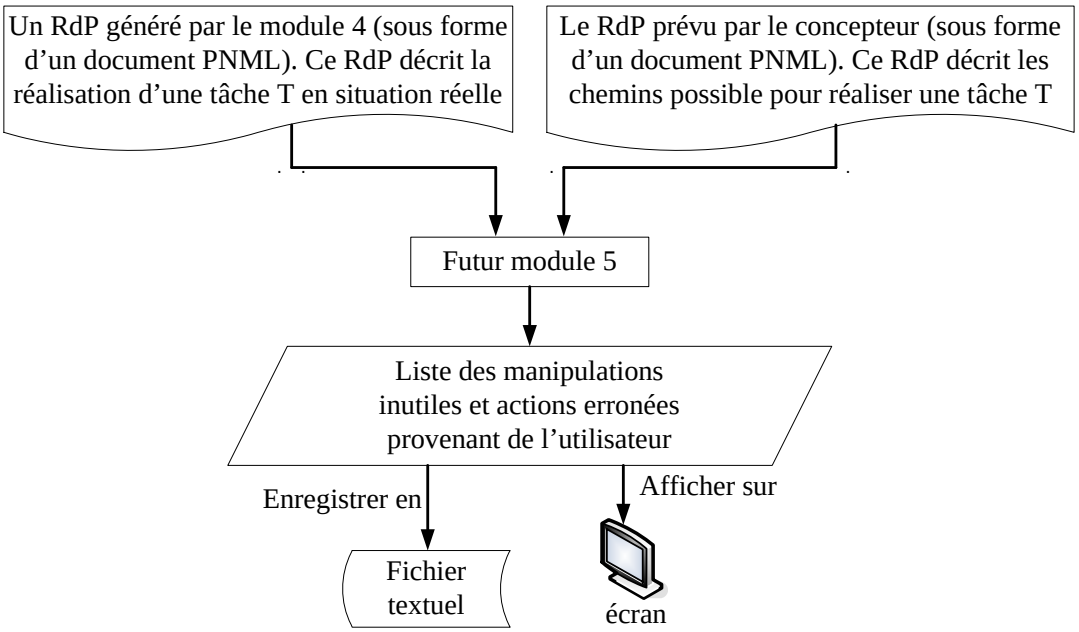

Figure 5.2 : Fonctionnement du futur module 5

- Pour l'instant, les associations entre les critères d'évaluation du module 6 et les résultats d'analyse des autres modules (3, 4, 5) ne sont pas encore formalisées. En conséquence, le module 6 travaille seulement comme un indicateur pour l'évaluateur. En effet, ce module 6 se limite à un rôle de fournisseur des indications nécessaires à l'évaluateur pour l'aider à interpréter les résultats d'analyse afin de critiquer le système et proposer des améliorations au concepteur. Dans l'avenir, il faudrait formaliser ces associations le plus possible. Cette formalisation servirait à augmenter l'automatisation du module 6. La nouvelle démarche d'évaluation d'un système interactif en utilisant l'environnement EISEval serait encore similaire à celle présentée et illustrée dans le chapitre 4 (cf. figure 4.9, section 3.3) mais l'évaluateur ne devrait plus saisir tous ses critiques en se basant sur l'indication d'un critère quelconque fourni par le module 6 pour critiquer le système selon ce critère. Le module 6 effectue de telles critiques et l'évaluateur peut modifier et/ou ajouter ses propres critiques si nécessaires. La figure 5.3 illustre la démarche d'un futur module 6.

  Afin d'évaluer les systèmes adaptatifs qui peuvent changer leurs comportements et interfaces en fonction du contexte d'utilisation, il faut prendre en compte le contexte lors de l'évaluation. Pour l'instant, c'est bien l'évaluateur qui doit prendre en compte le contexte lors de l'interprétation des résultats d'analyse de l'EISEval (en se basant sur les critères du module 6). Dans ce cas, pour évaluer les systèmes adaptatifs, on peut utiliser cet environnement pour évaluer le fonctionnement, l'interface du système dans les différents contextes. En interprétant ces résultats d'analyse, on peut savoir dans quels contextes les erreurs et problèmes sont apparus le moins et le plus et on peut suggérer au concepteur de revoir le fonctionnement, l'interface du système dans les contextes dans lesquels les erreurs et problèmes sont apparus le plus. En conséquence, on pourrait certainement évaluer la qualité de l'adaptation du système en se basant sur les données objectives. Dans cette perspective, il faudrait prendre en compte le contexte relativement à cette formalisation des associations entre les critères du module 6 et les résultats d'analyse issus des autres modules.

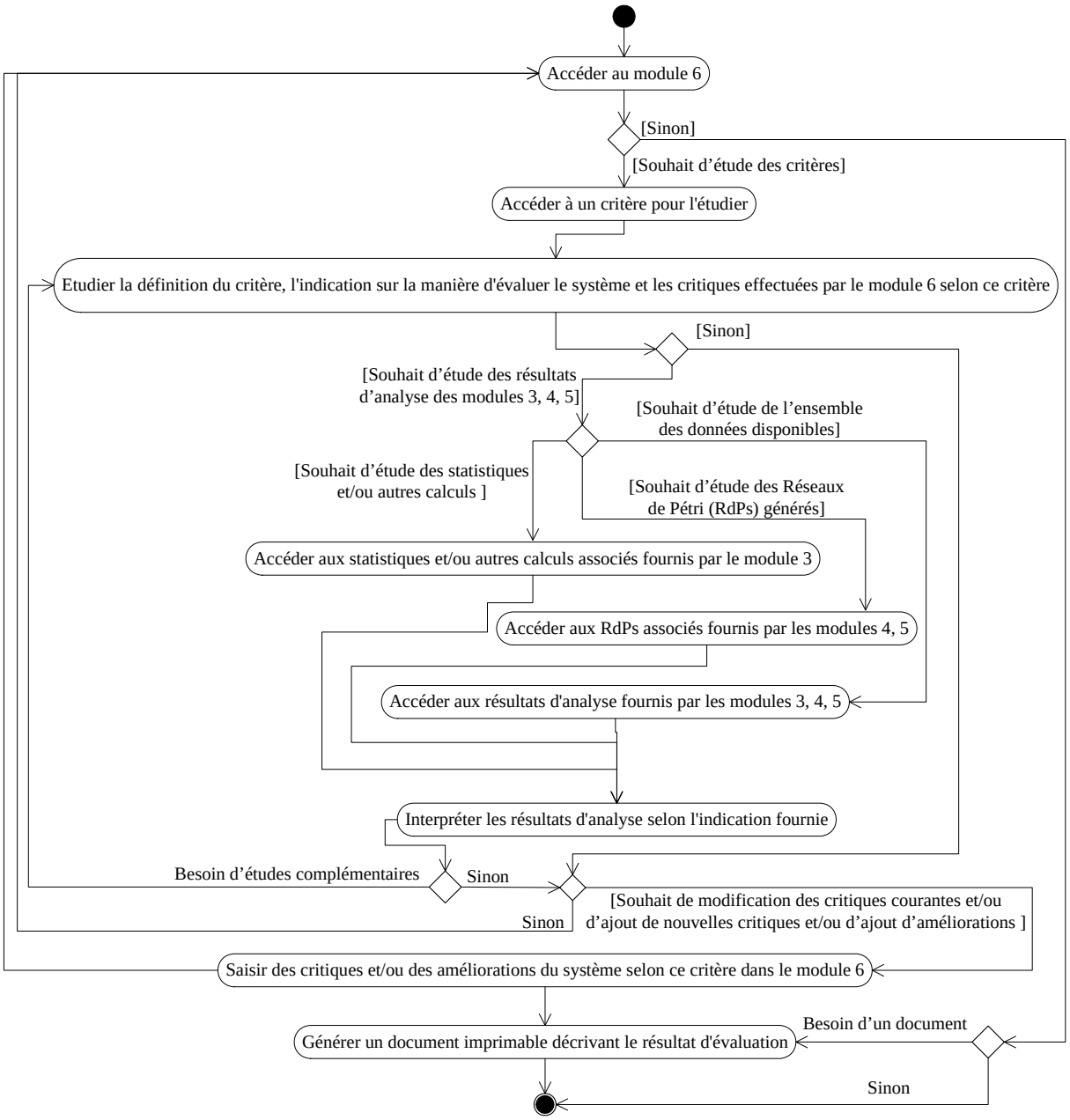

Figure 5.3 : Démarche d'évaluation d'un système interactif en utilisant l'environnement EISEval avec le futur module 6

4. Perspectives relatives à un futur environnement d'évaluation intégré

La figure 5.4 illustre un environnement intégré plus global que nous visons.

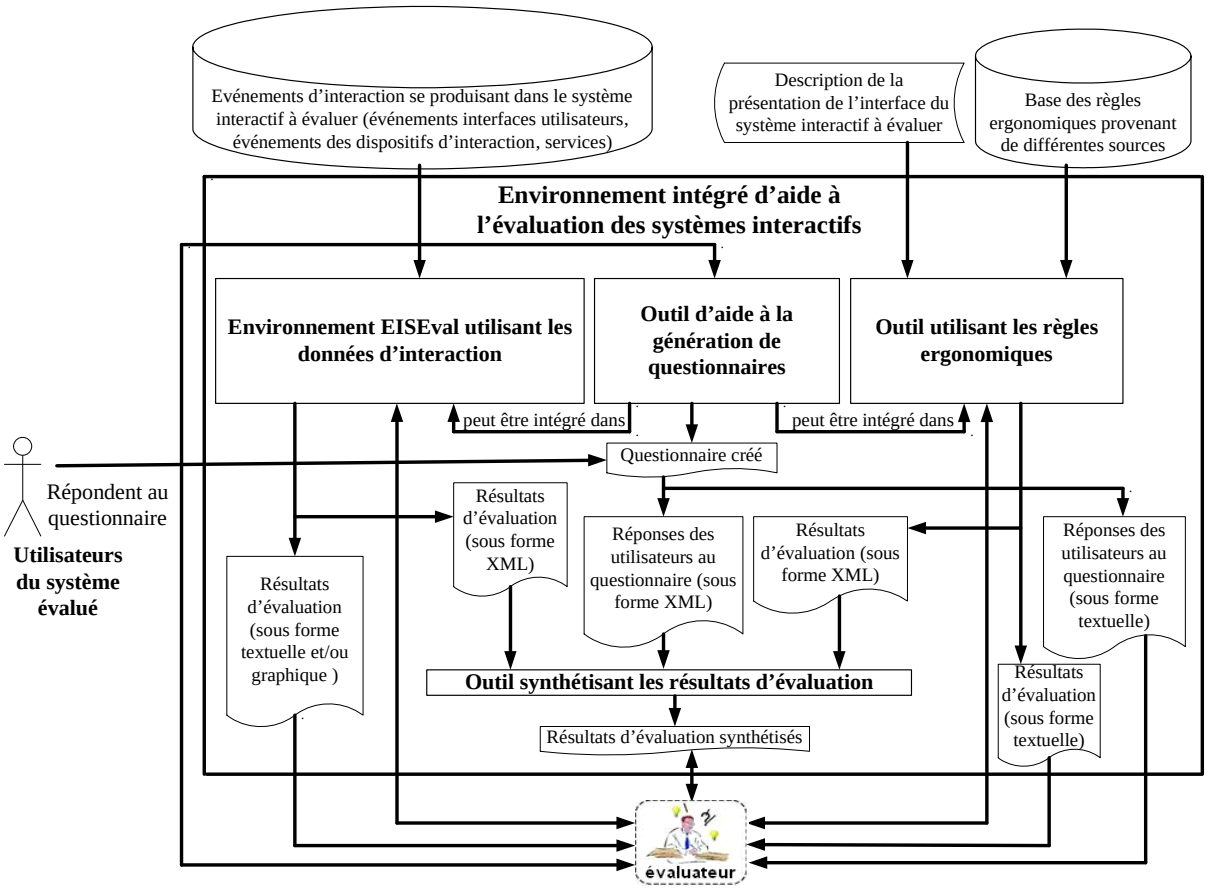

Figure 5.4 : Proposition d'un environnement intégré IEISEval pour évaluer les systèmes interactifs

Pour obtenir une évaluation plus pertinente et complète, il faudrait combiner différentes méthodes d'évaluation. Donc, nous visons un environnement intégrant différentes méthodes. La structure de cet environnement est spécifiée en figure 5.4. Pour l'instant, cet environnement est nommé IEISEval (Integrated Environment for Interactive System Evaluation)

Cet environnement IEISEval serait par exemple composé de trois systèmes correspondant à trois méthodes d'évaluation. L'environnement d'évaluation proposé et développé serait un de ces trois systèmes. Les deux systèmes restants seraient : un outil utilisant les règles ergonomiques (appelé temporairement EvalTUGL – Evaluation Tool Using Guidelines) et un outil d'aide à la génération de questionnaires. Comme l'illustre la figure 5.1, on peut constater que :

- L'évaluateur pourrait interagir avec ces trois systèmes et chaque système pourrait produire ses propres résultats d'évaluation. Nous proposons que chaque système puisse produire les résultats d'évaluation sous deux formes : une forme textuelle et/ou graphique utilisée pour l'évaluateur et une forme XML utilisée par un outil chargé de synthétiser les résultats d'évaluation de ces trois outils afin de produire un résultat d'évaluation final. Nous choisissons le format XML en raison de plusieurs de ses avantages[25] : les documents XML peuvent être manipulés par plusieurs outils disponibles ou même par n'importe quel éditeur de texte, il y a aussi plusieurs outils disponibles qui peuvent valider les documents XML en exploitant le principe de DTD (Document Type Definition), etc.

[25] http://www.w3.org/XML/

- La base des règles ergonomiques ne ferait pas partie de l'outil EvalTUGL et elle ne ferait pas partie de cet environnement intégré non plus parce que nous voulons assurer la séparation entre les règles ergonomiques et l'outil EvalTUGL qui contiendrait le moteur d'évaluation de l'interface (selon ces règles). Cette séparation permettrait d'ajouter et/ou de mettre à jour cette base de règles ergonomiques (selon l'évolution rapide des technologies en IHM et les nouveaux résultats de recherche dans le domaine ergonomique) sans affecter le moteur d'évaluation (et vice versa). En conséquence, il nous faudrait proposer une description formelle des règles ergonomiques afin que ces règles puissent être évaluées le plus automatiquement possible par l'outil EvalTUGL. Notons qu'un tel principe est expliqué dans [Jasselette *et al*., 2006].
- Dans cet environnement, l'outil EvalTUGL récupérerait les paramètres des éléments de l'interface (polices, couleurs, tailles, positions, etc.) à travers la description de l'interface du système interactif à évaluer. Cette description ne ferait pas partie de l'environnement. Le système interactif à évaluer pourrait être développé à l'aide de différents langages et environnements de programmation. Afin que cet outil EvalTUGL ne dépende pas d'un langage spécifique, il faudrait proposer une description de l'interface qui serait indépendante d'un langage spécifique. Après proposition d'une telle description, il y aurait besoin de développer des outils permettant de transformer des interfaces, développées dans différents langages, en cette description générique. Les personnes souhaitant approfondir cette piste peuvent consulter des documents sur les langages de description de l'interface utilisateur (UIDL - User Interface Description Language) comme UIML[26] (User Interface Markup Language) ; UsiXML[27] (USer Interface eXtensible Markup Language); XiML[28] (eXtensible Interface Markup Language); XUL[29] (XML-based User interface Language) ; etc. Ces langages UIDL permettent de décrire des interfaces en se basant sur XML. Un document UIDL décrit l'interface d'une manière générique pour qu'il puisse être transformé (rendering) en langage spécique (Html, WML, VoiceXML, Java, etc.) par les moteurs correspondants. Un document UIDL contient les informations relatives à la structure d'une interface, les propriétés des éléments d'interface (position, couleur, etc.) ; par exemple, un document décrivant l'interface selon la version 4.0 de UIML [Helm *et al*., 2009] fournit les informations relatives aux parties composant une interface (par exemple, des labels, des boîtes de texte, des boîtes de liste, etc.) ; au style de présentation, au contenu et aux comportements de chaque composant (par exemple, aux position, couleur et police d'un label, au contenu d'un boîte de texte, à la couleur de fond la fenêtre, aux événements susceptibles de se produire sur chaque composant, etc.). De telles informations sont nécessaires pour l'outil EvalTUGL.

Pour l'instant, nous proposons que la combinaison de ces méthodes se base sur une liste de critères ergonomiques. Par exemple, afin d'évaluer la *lisibilité* de l'interface, l'évaluateur pourrait se baser sur la fréquence d'apparition des EVIUs qui permettent à l'utilisateur de percevoir l'interface plus clairement comme *Zoom Avant*, *Zoom Arrière*, etc. (cf. 4.4, chapitre 4). La fréquence de ces EVIUs est fournie par l'environnement d'évaluation mais l'évaluateur pourrait aussi utiliser un outil utilisant les règles érgonomiques pour évaluer l'interface selon ce critère de *lisibilité*. En effet, cet outil pourrait récupérer les paramètres de l'interface comme les polices, les tailles, les couleurs, les positions des textes et d'autres éléments interactifs, etc., pour évaluer les règles ergonomiques qui respectent le critère de *lisibilité*. Un

[26] http://www.uiml.org/
[27] http://www.usixml.org/
[28] http://www.ximl.org/
[29] http://xulfr.org/; http://www.xul.fr/xml-xul.html

questionnaire peut être aussi utilisé pour savoir si les utilisateurs ont lu facilement les informations sur l'interface.

Intégré dans les méthodes du futur environnement, EvalTUGL viserait l'évaluation de l'aspect statique de l'IHM, EISEval viserait l'évaluation de l'aspect dynamique de l'IHM, et la méthode de questionnaire permettrait de recueillir des avis subjectifs provenant de l'utilisateur. Pour une perspective plus lointaine, nous souhaiterions un environnement EISEval étendu (Extended EISEval) qui remplacerait EISEval. Il combinerait des données d'interaction, de vidéo et d'oculométrie pour évaluer le système interactif. La vidéo permettrait à l'évaluateur de voir directement la situation réelle de travail de l'utilisateur. L'oculomètre aiderait l'évaluateur à comprendre la stratégie de navigation de l'utilisateur, d'identifier ses zones d'intérêt sur l'interface et les données quantitatives associées (nombre de fixations, durées de fixation, etc.). EISEval étendu combinerait ces différentes données en se basant sur les périodes de temps où les tâches sont réalisées, et produirait les résultats. L'architecture de l'environnement intégré IEISEval étendu est illustré en figure 5.5.

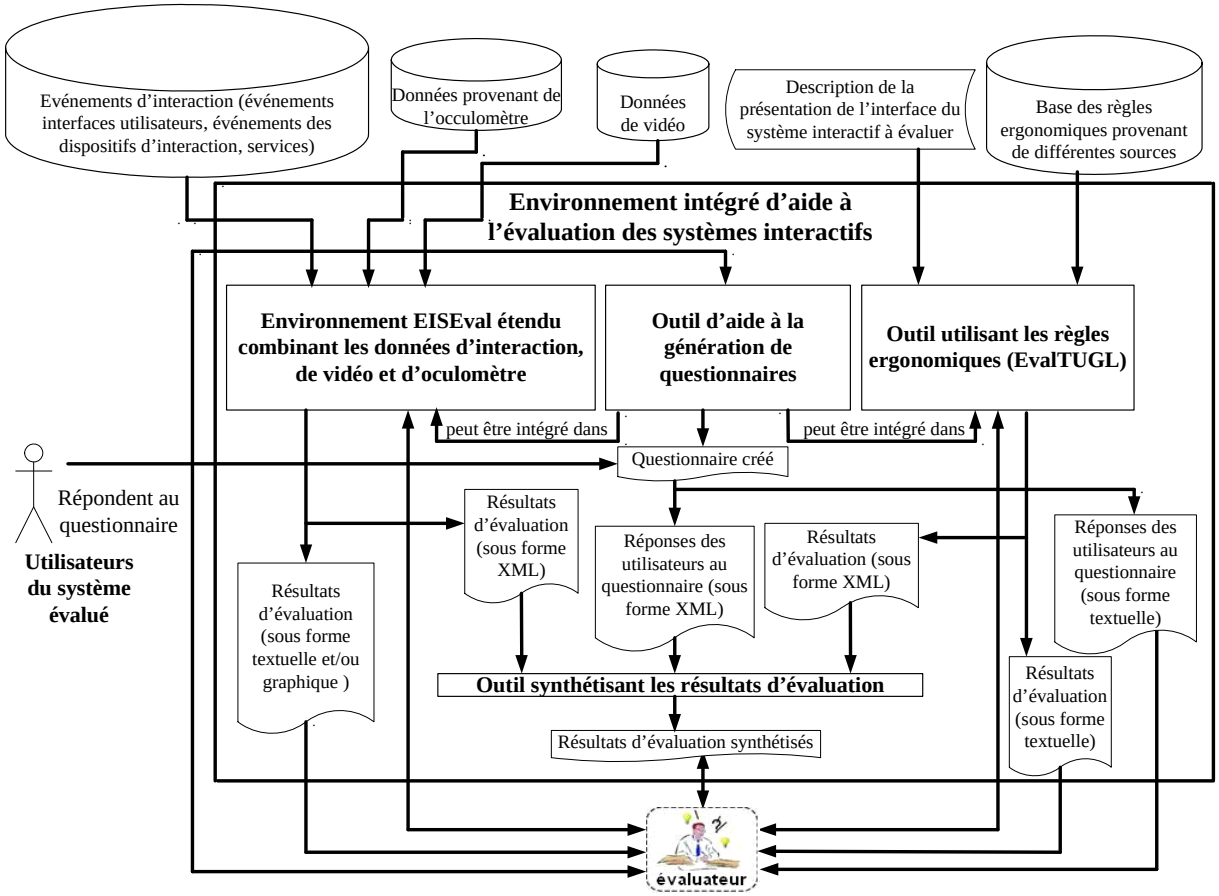

Figure 5.5 : Proposition d'un environnement intégré et étendu pour évaluer les systèmes interactifs

Nous pouvons déterminer quelques étapes à effectuer pour combiner les méthodes d'évaluation :

- Il faudrait déterminer la liste des critères ergonomiques communs qui sont à la base de combinaisons de méthodes d'évaluation.
- Ensuite, chaque critère serait associé à une échelle pour classer le système (par exemple, une échelle de 1 à 5 utilisée pour évaluer la *lisibilité* de l'interface).

- Puis, il faudrait proposer la manière d'associer l'échelle de chaque critère aux résultats d'évaluation de chaque méthode afin que cette méthode puisse classer le système selon ce critère. Il faut formaliser ces associations le plus possible en prenant en compte le contexte.
- Enfin, il faudrait déterminer une formule pour combiner les classements de toutes les méthodes selon chaque critère pour obtenir un classement final du système selon ce critère.

Pour chaque système à évaluer, on pourrait associer à chaque méthode du module 6 un poids pour distinguer son degré d'importance par rapport à l'évaluation de ce système.

Les méthodes d'évaluation considéréées dans l'environnement intégré IEISEval seraient associées à des systèmes indépendants. Chaque méthode pourrait produire ses propres résultats d'évaluation qui seraient combinés par un outil de génération de synthèse (cf. figures 5.4 et 5.5) pour donner des résultats d'evaluation finaux. Donc, la structure de cet environnement serait modulaire et ouverte. On pourrait ajouter d'autres méthodes dans une perspective plus lointaine.

5. Conclusion

Dans ce chapitre, nous avons proposé différentes perspectives de recherche nécessitant un ensemble de travaux à effectuer à l'avenir. Il faudrait appliquer les propositions suite à l'expérimentation pour améliorer quesques agents *interface* (*Etat_Trafic*, *Etat_Ligne* et *Message*) et modifier le mécanisme de traitement des perturbations du SAI. Il serait aussi nécessaire d'effectuer une deuxième expérimentation avec des utilisateurs professionnels en régulation de transport public pour détecter avec plus de profondeur les problèmes du SAI. L'environnement EISEval possède encore un ensemble de limitations relatives aux modules 1, 5 et 6. Il faudrait améliorer le module 1 pour assurer une capture efficace des données dans un environnement de réseau de faible débit. L'amélioration des modules 5 et 6 pourrait faciliter à l'évaluateur la réalisation de ses tâches et augmenter l'automatisation d'EISEval. L'environnement EISEval pourrait lui-même être amélioré et étendu ; il combinerait des données d'interaction, de vidéo et d'oculométrie pour évaluer les systèmes interactifs. Enfin, un environnement intégré IEISEval pourrait être proposé : il permettrait de combiner des résultats d'évaluation provenant des différentes méthodes (questionnaire, EISEval ou EISEval étendu et EvalTUGL utilisant des règles ergonomiques) pour l'aide à l'évaluation des systèmes interactifs. Cet environnement serait modulaire et ouvert ; donc on pourrait viser une combinaison d'autres méthodes dans une perspective plus lointaine.

Conclusion générale

Les travaux de recherche présentés dans ce mémoire se sont situés dans le domaine de l'Interaction Homme-Machine et plus particulièrement dans les thématiques de la conception et de l'évaluation des systèmes interactifs. Ils ont visé la proposition et le développement d'un environnement générique et configurable dit EISEval (Environment for Interactive System Evaluation) pouvant prendre en compte les spécificités d'architectures à base d'agents des systèmes interactifs, étendre les possibilités effectuées par les mouchards électroniques traditionnels et fournir quelques fonctionnalités originales par rapport à ceux-ci. Après la capture des interactions d'un système interactif à base d'agents (entre les utilisateurs et le système et entre ses agents constitutifs eux-mêmes), cet environnement est capable d'effectuer des analyses comme des classifications, des statistiques, des calculs liés à différentes mesures et d'afficher les résultats d'analyse sous forme textuelle ou graphique. Des Réseaux de Pétri (RdPs) sont générés pour reconstituer visuellement les processus d'activité réelle des utilisateurs et du système pour réaliser les tâches. La génération de RdPs et la comparaison entre RdPs sont utiles à l'évaluateur pour étudier le système. Les indications de l'environnement EISEval peuvent aider l'évaluateur à interpréter les résultats d'analyse des données capturées pour évaluer différents aspects d'un système interactif à base d'agents (interface, quelques propriétés non fonctionnelles du système, propriétés en lien avec l'utilisateur).

Ainsi, dans le premier chapitre, nous avons présenté quelques modèles d'architecture représentatifs parmi les plus connus des deux grands types de modèles (les modèles *fonctionnels* et *structurels*) comme Arch, Seeheim, PAC, MVC, AMF et quelques modèles d'architectures hybrides comme H4 ou PAC-Amodeus. Les modèles d'architecture fonctionnels visent à décomposer un système interactif en plusieurs composants fonctionnels et indépendants pour assurer une séparation nette entre la partie *Application* et la partie interface (*Présentation*). Donc, on peut modifier une partie sans en affecter une autre.

Ces modèles fournissent aux concepteur un cadre de pensée pour analyser un grand système interactif en le décomposant en modules différents. Cependant, la granularité de ces modèles ne peut pas offrir de solution précise pour concevoir des systèmes interactifs complexes, par exemple, les systèmes de supervision dans le contexte industriel. Les modèles d'architectures structurels visent une décomposition avec un niveau de granularité plus fin. Le principe de base de ces modèles réside dans la décomposition d'un système interactif en plusieurs entités autonomes et coopératives (dits agents). La structure et la communication entre ces entités sont aussi décrites (alors que de telles decriptions sont inexistantes dans les modèles fonctionnels). La décomposition en plusieurs entités autonomes et coopératives pourrait accélérer l'interaction entre le système interactif et l'utilisateur. Cette caractéristique est nécesssaire pour la conception des systèmes hautement interactifs (par exemple, les systèmes de supervision). Cependant, les modèles hybrides possèdent des limitations. En effet, le rôle des agents n'est pas clairement précisé ; en plus, le concepteur peut avoir des difficultés pour concevoir l'interface globale du système parce que les facettes Présentation sont réparties sur de nombreux agents. Les modèles fonctionnels peuvent remédier à ce problème parce qu'ils ont un seul composant Présentation. Les qualités et les défauts de chaque type de modèles (fonctionnels et structurels) ont donné naissance à des architectures hybrides qui visent à tirer profit des avantages de ces deux types de modèles. Ces modèles structurent certains composants des modèles fonctionnels d'une façon structurelle et plus précise comme H^4, PAC-Amodeus que nous avons présentés dans le premier chapitre. Le modèle d'architecture à base d'agents que nous considérons dans cette thèse se situe dans les

modèles hybrides. Ce modèle à base d'agents, qui emprunte les principes de ces deux types de modèles d'architecture pour exploiter leurs avantages, a été globalement présenté. Les modèles hybrides ont déjà été proposés dans la littérature, mais le modèle d'architecture à base d'agents sur lequel nous nous appuyons dans cette thèse, vise principalement la conception de systèmes interactifs complexes de supervision dans le domaine industriel. Ce modèle s'appuie sur une combinaison entre le modèle fonctionnel Seeheim (en séparant un système interactif en trois composants fonctionnels : *Interface*, *Contrôleur* et *Application*) et la notion d'agents (comme dans les modèles structurels). Chaque composant peut être vu comme trois systèmes composés d'agents sans qu'il ne soit nécessaire de se baser sur une arborescence d'agents comme dans le modèle structurel PAC. Ces trois composants sont prévus pour fonctionner en parallèle. Avec ce modèle, les difficultés à traiter par les concepteurs lorsqu'ils l'appliquent pour concevoir un système interactif, sont l'organisation des agents ainsi que la distribution des services entre les agents du système. Une discussion sur ce modèle a terminé le premier chapitre de ce mémoire.

Le deuxième chapitre a concerné les méthodes et outils d'évaluation des systèmes interactifs. Après avoir introduit le principe de base de l'évaluation (visant à estimer deux critères globaux : l'utilité et l'utilisabilité de l'IHM dans des buts d'amélioration et/ou de validation) et différentes classifications des méthodes d'évaluation (provenant de notre équipe et aussi d'autres auteurs), un état de l'art des outils d'aide à l'évaluation a été présenté. Ces outils fournissent différents niveaux d'automatisation, et ont pour objectif de faciliter à l'évaluateur la réalisation de ses tâches. Ils offrent potiellement plusieurs avantages : réduction du coût et du temps de l'évaluation, facilité de détection des erreurs, etc. Parmi les différents types d'outils d'aide à l'évaluation, nous nous sommes focalisés sur deux types d'outils : les outils utilisant les règles ergonomiques (ergonomic guidelines) et les mouchards électroniques. Les outils utilisant des règles pour l'évaluation visent à examiner le code source ou une description de l'interface pour déterminer si l'interface viole les règles (provenant d'une ou plusieurs sources). Ces règles sont spécifiées de façons différentes dans les approches étudiées et stockées dans des bases pour être utilisées. En général, on peut constater que ces outils permettent d'évaluer l'aspect statique de l'IHM ; ils ne prennent souvent pas en compte les interactions entre l'utilisateur et l'interface. Les mouchards électroniques visent justement à capturer ces interactions pour les analyser ultérieurement. Les résultats d'analyse (statistiques, classifications, recherche de séquences d'interaction, etc.) sont affichés sous des formes visant à aider l'évaluateur à accomplir ses tâches. Nous avons examiné, sans souci d'exhaustivité, quelques méthodologies et outils parus à ce sujet depuis une dizaine d'années, comme WebTango, DESTINE, Sherlock, WebRemUSINE, MultiDevice RemUSINE, IBOT, WET, WebQuilt, etc. Le mouchard MESIA proposé lors de travaux précédents au laboratoire LAMIH a été aussi positionné dans cette section. Enfin, une synthèse de ces méthodologies et outils, se basant sur différentes dimensions, a terminé ce deuxième chapitre.

Dans le troisième chapitre, après avoir critiqué le mouchard MESIA et d'autres mouchards traditionnels, les principes de l'environnement EISEval (Environment for Interactive System Evaluation), proposé dans le cadre de cette thèse, ont été présentés. Le respect de ces principes permet à EISEval de remédier à certaines les limitations de MESIA et des mouchards traditionnels. En effet, EISEval peut prendre en compte les spécificités de l'architecture à base d'agent des systèmes interactifs et fournir aussi quelques fonctionnalités originales par rapport aux mouchards traditionnels. Il peut aussi aller plus loin que ces mouchards traditionnels pour évaluer différents aspects d'un système interactif . Pour proposer EISEval – un environnement générique et configurable d'aide à l'évaluation des systèmes interactifs à base d'agents, il était nécessaire de décrire formellement cette architecture à base d'agents. En fait, une description formelle de cette architecture avait été

déjà présentée dans [Trabelsi, 2006] mais elle n'est pas suffisamment exploitable vis-à-vis de notre but. Donc, nous avons proposé une autre description formelle de l'architecture à base d'agents des systèmes interactifs. C'était notre première tâche dans le cadre de cette thèse. Enfin, les sept modules indépendants qui composent cet environnement ont été présentés. Le principe de couplage entre l'environnement et le système interactif à évaluer a été abordé dans la section concernant le module 1 chargé de capturer les événements d'interaction et de les stocker dans des bases afin que les autres modules puissent les analyser ultérieurement. Ensuite, le module 2 permettant de déterminer les tâches que l'utilisateur a réalisées et le module 3 effectuant des analyses des événements capturés ont été sucessivement décrits. Les deux modules 4-5 visant à générer les RdPs (afin de reconstituer le processus des activités réelles de l'utilisateur et du système pendant les réalisations des tâches) et les confronter ont été aussi présentés. Enfin, la description du module 6 fournissant à l'évaluateur des indications nécessaires pour l'aider à interpréter des résultats d'analyse a terminé ce troisième chapitre. Afin d'illustrer les activités de ces modules, des données issues d'une expérimentation ont été utilisées.

Le quatrième chapitre a eu pour objectif de présenter la manière d'utiliser l'environnement EISEval pour évaluer un système interactif. Ce chapitre a décrit une expérimentation réalisée au sein de notre laboratoire LAMIH avec différents sujets. Pendant cette expérimentation, l'environnement d'évaluation a été appliqué pour évaluer un prototype de système interactif appelé SAI. L'introduction des trois sous-systèmes liés au projet SART (visant à aider les régulateurs en salle de contrôle à accomplir leurs tâches en mode normal et en mode dégradé de fonctionnement du réseau du trafic) a commencé ce chapitre. Ensuite, un de ces trois sous-systèmes, le système interactif à base d'agents SAI (Système d'Aide à l'Information) – un système d'aide à l'information voyageurs du transport public terrestre Valenciennois a été détaillé. La configuration de déploiement, le scénario de l'expérimentation et la démarche d'évaluation d'un système interactif en utilisant l'environnement EISEval ont été successivement expliqués et illustrés. Enfin, les résultats d'évaluation ont été détaillés. L'évaluateur a effectué ses tâches en utilisant les résultats d'analyse, ceci à l'aide du module 6. En effet, l'évaluateur s'est basé sur les critères d'évaluation fournis (qui sont associés aux résultats d'analyse des autres modules de l'environnement d'évaluation) pour critiquer le système SAI et pour proposer des améliorations au concepteur.

Enfin, dans le dernier chapitre, quelques perspectives de recherche ont été présentées. Ces perspectives ont concerné le système SAI, les modules de l'environnement EISEval et aussi un environnement intégré plus global nommé IEISEval (Integrated Environment for Interactive System Evaluation) qui pourrait être visé. Concernant les perspectives relatives au SAI, il s'agirait d'appliquer les améliorations proposées dans le quatrième chapitre et de prévoir une deuxième expérimentation avec des utilisateurs professionnels en régulation de transport public. Les perspectives relatives à EISEval viseraient à améliorer les modules 1, 5 et 6 afin d'améliorer la méthode de capture des événements, faciliter le travail de l'évaluation et augmenter l'automatisation d'EISEval. Les perspectives relatives à un environnement intégré et plus global (appelé IEISEval) concerneraient la proposition d'un modèle d'un environnement permettant de combiner les résultats d'évaluation de différentes méthodes pour obtenir une évaluation plus pertinente et complète. Cet environnement intégré IEISEval serait composé d'une méthode de génération de questionnaires (pour recueillir des données subjectives de l'utilisateur), d'un outil appelé EvalTUGL (Evaluation Tool Using Guidelines) utilisant des règles ergonomiques (pour évaluer l'aspect statique de l'IHM) et EISEval (ou EISEval étendu avec prise en compte de données vidéo et occulométrique) (pour évaluer l'aspect dynamique de l'IHM). L'environnement IEISEval serait modulaire et ouvert ; donc

une intégration d’autres méthodes d’évaluation serait très prometteuse dans une perspective plus lointaine.

Bibliographie

- [Abran *et al*., 2003] A. Abran, A. Khelifi, W. Suryn, A. Seffah. Consolidating the ISO Usability Models. In: Proc. of 11th Int. Software Quality Management Conference, Springer, Glasgow, Scotland, UK, 23–25 April 2003.
- [Adam, 2000] E. Adam. Modèle d'organisation multi-agent pour l'aide au travail coopératif dans les processus d'entreprise application aux systèmes administratifs complexes. Thèse de l'Université de Valenciennes et du Hainaut-Cambrésis, janvier 2000.
- [Adina *et al*., 2002] F. Adina, D. Kayser, S. Pentiuc et A. El-Fallah Segrounichi. Agents intelligents, cours WEB interactf. Politechnica University of Bucharest, Accessible à http://turing.cs.pub.ro/auf2/.
- [Allen et Garlan, 1997] R. Allen, D. Garlan. A Formal Basis for Architectural Connection., ACM Transactions on Software Engineering, 7 (3), July 1997.
- [Aknine, 2000] S. Aknine. Models and Methods for Multi-agent coordination. Thèse de l'Université Paris-Dauphine, 2000.
- [Alty et Bergan, 1992] J.L. Alty, M. Bergan. The design of multimedia interfaces in process control. In: H. G. Stassen (Ed.), Analysis, Design and Evaluation of Man-Machine Systems (Preprints, The Hague), Pergamon press, Oxford, 1992.
- [Alty, 1995] J.L. Alty. Multi-media interfaces in process control – a methodological approach. Proceeding IEEE Conference on Systems, Man and Cybernetics, Le Touquet, France, volume 1, pp. 361-366, October 17-20, 1995.
- [Al-Qaimari *et al*.,1999] Al-Qaimari, Ghassan, and D. McRostie. KALDI: A computer-aided usability engineering tool for supporting testing and analysis of human-computer interaction. In: Proceedings of the 3rd International Conference on Computer-Aided Design of User Interfaces, ed.by J. Vanderdonckt and A. Puerta, Louvain-la-Neuve, Belgium. Dordrecht, The Netherlands: Kluwer Academic Publishers, 1999.
- [Apple, 1992] Apple Computers. Macintosh human interface guidelines. Reading, Massachusetts: Addison Wesley, 1992.
- [Aubry, 1992] J.F. Aubry. La supervision : nécessité, offre, guide de choix. Acte de la journée « La supervision », Centre des technologies nouvelles, Caen, 19 Mars 1992.
- [Balbo, 1994] S. Balbo. Evaluation ergonomique des interfaces utilisateur : un pas vers l'automatisation. Thèse de doctorat, université Joseph Fourier, Grenoble I, septembre 1994.
- [Bastien et Scapin, 1993] J.M. Bastien, D.L. Scapin. Ergonomic Criteria for the evaluation of human-computer interfaces, Rapport Technique n°156, INRIA, Rocquencourt, 1993.
- [Bastien et Scapin, 1994] J.M. Bastien, D.L. Scapin. Evaluating a user interface with ergonomic criteria, Rapport de recherche de l'INRIA – Rocquencourt, 1994.
- [Bastien *et al*.,1999], J.M.C. Bastien, D.L. Scapin, C. Leulier. The ergonomic criteria and the ISO/DIS 9241-10 dialogue principles: a pilot comparison in an evaluation task. INRIA, Domaine de Voluceau, Rocquencourt-B.P. 105, 78153 Le Chesnay Cedex, France, 1999.
- [Bastien et Scapin, 2001 J.M.C Bastien, D.L. Scapin. Evaluation des systèmes d'information et critères ergonomiques. In : Kolski C. (Ed.), Environnement évolués et évaluation de l'IHM. Interaction Homme Machine pour les SI, Volume 2, pp. 53-80. Paris : Hermès, 2001.
- [Bastien et Scapin, 2002] J.M.C. Bastien, D.L. Scapin. Les méthodes ergonomiques : de l'analyse à la conception et à l'évaluation. In: Ergonomie et Informatique Avancée, Ergo-IA'2002, A. Drouin, J. Robert (Eds), I.D.L.S., Biarritz, 8-10 octobre 2002.

- [Beirekdar, 2004] A. Beirekdar. A methodology for automating web usability and accessibility evaluation by guideline, Thèse de Doctorat, Facultés Universitaires Notre-Dame de la Paix, Namur, Août 2004.
- [Bass *et al*.,1991] L. Bass, R. Little, R. Pellegrino, S. Reed. The Arch Model: Seeheim revisited. Proceedings of User Interface Developpers'Workshop, Seeheim, 1991.
- [Baccino *et* al., 2005] T. Baccino, C. Bellino, T. Colombi. Mesure de l'utilisabilité des interfaces. TIC et Sciences Cognitives. Hermes. 2005. ISBN : 2-7462-1026-6.
- [Bernonville *et al*., 2006] S. Bernonville, N. Leroy, C. Kolski, M. Beuscart-Zéphir. Explicit combination between Petri Nets and ergonomic criteria: basic principles of the ErgoPNets method, Proceedings of the 25th Edition of European Annual Conference on Human Decision-Making and Manual Control, Valenciennes, France, 27-29 September 2006.
- [Bernonville, 2008] S. Bernonville. Exploitation des techniques de modélisation du GL et de l'IHM pour la création de supports communs entre intervenants de projet de développement de systèmes interactifs et pour la modélisation des situations de travail complexes. Thèse de l'Université de Valenciennes et du Hainaut-Cambrésis, Décembre 2008.
- [Billington *et al*., 2003], J. Billington, S. Christensen, K.V. Hee, E. Kindler, O. Kummer, L. Petrucci, R. Post, C. Stehno, M. Weber. The Petri Net Markup Language: Concepts, Technology, and Tools, March 2003, preliminary version of a submission to the conference proceedings of the ICATPN 2003, Eindhoven, Netherlands, June 2003.
- [Bond et Gasser, 1988] A. Bond, L. Gasser. Readings in distributed artificial intelligence. Morgan Kaufman, San Mateo, Cam 1988.
- [Brajnik, 2000] G. Brajnik. Automatic web usability evaluation: Where is the limit? In Proceedings of the 6th Conference on Human Factors & the Web, Austin, TX. Accesible à : http://www.tri.sbc.com/hfweb/brajnik/hfweb-brajnik.html, 2000.
- [Bradshaw, 1997] J.M. Bradshaw. Software agents, Menlo Park, AAAI Press & MIT Press, 1997.
- [Calvary *et al*.,1996] G. Calvary, J. Coutaz, L. Nigay, D. Salber. PAC+3, un modèle d'architecture générique pour systèmes multi-utilisateurs. In: Proc. IHM'96, 8èmes Journées de l'Ingénierie de l'Interaction Homme-Machine, pp. 125-130, Grenoble, Septembre 1996.
- [Calvary *et al*., 1997] G. Calvary, J. Coutaz, L. Nigay. From Single-User Architectural Design to PAC*: a Generic Software Architecture Model for CSCW. Proceedings of CHI 97, ACM publ., pp. 242-249, 1997.
- [Calvary *et al*., 2001] G. Calvary, J. Coutaz, D. Thevenin, A unifying reference framework for the Development of Plastic User Interfaces, IFIP Working Conference, EHCI May 2001, Toronto.
- [Castelfranchi, 1995] Castelfranchi. Guarantees for autonomy in cognitive agent architecture. In: Wooldridge, M. and Jennings, N. R., editors, Intelligent Agents: Theories, Architectures, and Languages (LNAI Volume 890), pp. 56–70 Springer-Verlag, Heidelberg, Germany, 1995.
- [Chaib-Draa *et al*., 2001] B. Chaib-Draa, B. Moulin, I. Jarras. Agent et Systèmes Multiagents. In: Principes et architecture des systèmes multi-agents. JP. Briot et Y Demazeau (eds). Hermes, Lavoisier, 2001.
- [Chalon, 2004] R. Chalon. Réalité Mixte et Travail Collaboratif : IRVO, un modèle de l'Interaction Homme–Machine. Thèse de Doctorat, Ecole Centrale de Lyon, Décembre 2004.
- [Chi *et al*., 2001] Ed H. Chi, P. Pirolli, K. Chen, J. Pitkow. Using information scent to model user information needs and actions on the web. In: Proceedings of the Conference on Human Factors in Computing Systems, volume 1, pp. 490-497, Seattle, WA. New York, NY: ACM Press, 2001

- [Coutaz, 1987] J. Coutaz. PAC, an Object-Oriented Model for Dialog Design, In Bullinger, Hans-Jorg, Shackel, Brian (ed.), INTERACT 87, 2nd IFIP International Conference on Human-Computer Interaction, Stuttgart, Germany, September 1-4, 1987, p.431-436.
- [Coutaz, 1990] J. Coutaz. Interface homme-ordinateur, conception et réalisation. Dunod, Paris,1990.
- [Coutaz et Nigay, 2001] J. Coutaz, L. Nigay. Architecture logicielle conceptuelle des systèmes interactifs, chapter 7 de « Analyse et Conception de l'Interaction Homme-Machine dans les systèmes d'information ». Dans Kolski (Ed.), pp. 207-246, Paris: Éditions Hermes, 2001.
- [CPN, 2001] CPN Tools. Accessible à : http://www.daimi.au.dk/CPNtools. 2001/09/11.
- [Cugini et Scholtz, 1999] J. Cugini, J. Scholtz. VISVIP: 3D visualization of paths through web sites. In: Proceedings of the International Workshop on Web-Based Information Visualization, 259-263, Florence, Italy. Institute of Electrical and Electronics Engineers, 1999.
- [Dance *et al*., 1987] J.R. Dance, T.E. Granor, R.D. Hill, S.E. Hudson, J. Meadas, B.A. Myers, A. Sdhulert. The run-time structure of UIMS-Supported application. Computer graphics, 21 (2), avril 1987.
- [Daassi *et al*., 2003] O. Daassi, G. Calvary, J. Coutaz, A. Demeure. Comet. Une nouvelle génération de widget pour la plasticité des interfaces. IHM03, ACM Press, pp. 64-71, 2003.
- [David et Alla, 1992] R. David, H. Alla. Du Grafcet aux réseaux de Petri. Editions Hermès, Paris, 1992.
- [Denley et Long, 1997] I. Denley et J. Long. A planning aid for human factors evaluation practice. Behaviour & Information Technology, vol.16, n°415, pp. 203-219, 1997.
- [Depaulis, 2002] F. Depaulis. Vers un environnement générique d'aide au développement d'applications interactives de simulations de métamorphoses. Thèse de Doctorat, Université de Potiers, novembre 2002.
- [Design/CPN, 2001] Design/CPN. Acessible à : http://www.daimi.au.dk/designCPN/, 2001/09/21.
- [Dix *et al*., 1993] A. Dix, J. Finlay, G. Abowd, R. Beale. Human-Computer Interaction, Prentice-Hall, New Jersey, 1993.
- [Doniec *et al*., 2006] A. Doniec, R. Mandiau, S. Piechowiak, S. Espié, L'anticipation comme modèle d'interaction : application à la coordination multi-agent en simulation. Actes de RFIA 2006, 15ème congrès francophone Reconnaissance des formes et Intelligence Artificielle, Presses Universitaires François-Rabelais, Tours, 25-27 janvier 2006, ISBN 2-86906-216-8.
- [Dragicevic, 2004] P. Dragicevic. Un modèle d'interaction en entrée pour des systèmes interactifs multi-dispositifs hautement configurables. Thèse de Doctorat, Université de Nantes, Mars 2004.
- [Duval et Nigay, 1999] T. Duval, L. Nigay. Implémentation d'une application de simulation selon le modèle PAC-Amodeus. Actes de IHM1999, pp. 86-93, Montpellier, Cepadues Publ. 22-26 nov. 1999.
- [Eason, 1984] K. D. Eason. Towards the Experimental Study of Usability. Behaviour and Information Technology 3, pp. 133-143, 1984.
- [Eisenstein *et al*., 2001] J. Eisenstein, J. Vanderdonckt, A. Puerta. Applying Model-Based Techniques to the Development of UIs for Mobile Computers. In IUI01: 2001 International Conference on Intelligent User Interfaces. Santa Fe, NW, pp. 69-76, 2001.
- [Enembreck et Barthès 2005] F. Enembreck, J-P. A. Barthès. ELA - A new Approach for Learning Agents. Journal of Autonomous Agents and Multi-Agent Systems, 3(10): 215-248, 2005.

- [Etgen et Cantor, 1999] M. Etgen, J. Cantor. What does getting WET (Web Event-logging Tool) Mean for Web Usability?. User Experience Engineering Division AT&T Labs Middletown, NJ, USA, 1999.
- [Ezzedine et Abed, 1997] H. Ezzedine, M. Abed. Une méthode d'évaluation d'Interface Homme Machine de supervision d'un procédé industriel. Journal Européen des Systèmes Automatisés (RAIRO-APII-JESA), 1997, 7, pp. 1078-1110
- [Ezzedine, 2002], H. EZZEDINE. Méthodes et Modèles de Spécification et d'Evaluation des Interfaces Homme-Machine dans les systèmes industriels complexes. Mémoire d'HDR, Université de Valenciennes et du Hainaut-Cambrésis, décembre 2002
- [Ezzedine *et al*., 2003] H. Ezzedine, A. Trabelsi, C. Kolski. Modelling of Agent oriented Interaction using Petri Nets, application to HMI design for transport system supervision. In: P. Borne, E.Craye, N. Dangourmeau (Ed.), CESA2003 IMACS Multiconference Computational Engineering in Systems Applications, Ecole Centrale Lille, Villeneuve D'Ascq, pp. 1-8, Lille, France, July 9-11, 2003.
- [Ezzedine *et al*., 2005] H. Ezzedine, C. Kolski, A. Péninou. Agent-oriented design of human–computer interface:application to supervision of an urban transport network. Engineering Applications of Artificial Intelligence, 18, pp. 255-270, 2005.
- [Ezzedine et Trabelsi, 2005] H. Ezzedine, A. Trabelsi. From the design to the evaluation of an agentbased human-machine interface. Application to supervision for urban transport system. In: P.Borne, M. Benrejeb, N. Dangoumeau, L. Lorimier (Ed.), IMACS World Congress "Scientific Computation, Applied Mathematics and Simulation" (July 11-15, Paris), ECL, pp. 717-725, juillet 2005.
- [Ezzedine *et al*., 2006] H. Ezzedine, A. Trabelsi, C. Kolski. 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, 2006.
- [Farenc, 1997] C. Farenc. ERGOVAL: une méthode de structuration des règles ergonomiques permettant l'évaluation automatique d'interfaces graphiques. Thèse de Doctorat, Université Toulouse 1, janvier 1997.
- [Fayech, 2003] B. Fayech. Régulation des réseaux de transport multimodal : Systèmes multi agents et algorithmes évolutionnistes. Thèse de doctorat, Université des Sciences et Technologies de Lille. Octobre 2003.
- [Fekete, 1996] J.D. Fekete. Un modèle multicouche pour la construction d'applications graphiques interactives. Thèse de Doctorat, Université de Paris XI Orsay, Janvier 1996.
- [Ferber, 1991] J. Ferber. Conception et programmation par objets. Hermes, 1991.
- [Ferber, 1995] Ferber, J.: Les systèmes multi-agents. InterEditions, 1995.
- [Flanders et Willis, 1998] V. Flanders, M. Willis. Web Pages That Suck: Learn Good Design by Looking at Bad Design. San Francisco, CA: SYBEX, 1998.
- [Folmer et Bosch, 2003] E. Folmer, J. Bosch. Case studies on Analyzing Software Architectures for Usability. 31st EUROMICRO Conference on Software Engineering and Advanced Applications, pp. 206-213, 2005.
- [Franklin and Graesser, 1996] S. Franklin, A. Graesser. Is it an agent or just a program?: a taxonomy for autonomous agents. In: Intelligent Agents III : Agents Theories, Architectures and Languages, ECAI'96 Workshop (ATAL), Springer-Verlag, 1996.
- [Galiz,1988] W.O. Galitz. Handbook of Screen Format Design. 3ème éd., Q.E.D. Information Sciences, Wellesley (Massachusetts), 1988.

- [Gilmore *et al*., 1989] W.E. Gilmore, D.I. Gertman, H.S. Blackman. User–Computer Interface in Process Control, a Human Factors Engineering Handbook. Academic Press, New York, 1989
- [Gasser, 1989] L. Gasser, M.N. Huhns. Distributed artificial intelligence. Vol. 2, Pitman, London, 1989.
- [Genesereth et Ketchpel, 1994] M.R. Genesereth, S.P. Ketchpel. Software agents. Communications of the ACM, 37(7):48–53, 1994.
- [Girard et Ribette, 2001] D. Girard, J.N. Ribette. OpenSourceJava-Troisième partie : MVC2 avec Struts. Avril 2001. Accessible à : http://www.applicationservers.com/articles/pdf/opensourcejava-struts.pdf
- [Golden *et al*., 2005] E. Golden, B.E. John, L.Bass. Quality vs. quantity: Comparing evaluation methods in a usability-focused software architecture modification task. Proceedings of the 4th International Symposium on Empirical Software Engineering, Noosa Heads, Australia, November 17-18, 2005.
- [Gomes, 2003] T. Gomes. Les terminaux mobiles vont-ils révolutionner les interfaces homme-machine ? Mesures, 758, pp. 44-48, 2003. Accessible à : http://www.mesures.com/archives/Terminaux%20mobiles.pdf
- [Goschnick et Steding, 2001] S. Goschnick, L. Steding. ShadowBoard: an Agent-orientated Model-View-Controller (AoMVC) Architecture for a Digital Self. The 2001 International Workshop on Agent Technologies over Internet Applications, Proceedings of the Seventh International Conference on Distributed Multimedia Systems, 24 - 29, Knowledge System Institute (Illinois), 2001.
- [Grislin, 1995] M. Grislin. Définition d'un cadre pour l'évaluation a priori des interfaces homme-machine dans les systèmes industriels de supervision. Mémoire de Doctorat, Université de Valenciennes et du Hainaut-Cambrésis, janvier 1995.
- [Grislin et Kolski, 1996] M. Grislin, C. Kolski. Evaluation des interfaces homme-machine lors du développement de système interactif. Technique et Science Informatiques (TSI), 3, pp. 265-296, 1996.
- [Grislin-Le Strugeon et Péninou, 1998] E. Grislin-Le-Strugeon, A. Péninou. Interaction Homme-SMA : réflexion et problématiques de conception. In Barthès J.P., Chevrier V. & Brassac C. (Eds.), Sytèmes multi-agents, de l'interaction à la socialité. Actes des JFIADSMA'98. pp. 133-146. Paris: Editions Hermès, 1998.
- [Grislin-Le Strugeon *et al*., 2001] E. Grislin-Le Strugeon, E. Adam, C. Kolski. Agents intelligents en interaction homme-machine dans les systèmes d'information. In: Kolski C. (Ed.), Environnements évolués et évaluation de l'IHM, pp. 207-248, Paris: Éditions Hermes, 2001.
- [Goldberg, 1983] A. Goldberg. Smaltalk-80, the interactive programming environnement. Addison-Wesley, 1983
- [Grammenos *et al*., 2000a] D. Grammenos, D. Akoumianakis, C. Stephanidis. 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, 2000.
- [Grammenos *et al*., 2000b] D. Grammenos, D. Akoumianakis, C. Stephanidis. Sherlock: A Tool Towards Computer-Aided Usability Inspection. In: J. Vanderdonckt & C. Farenc (Eds), *Proceedings of the Scientific Workshop on "Tools for Working with Guidelines" (TFWWG 2000)*, Biarritz, France, 7-8 October (pp. 87-97). London: Springer-Verlag, 2000.
- [Guittet, 1995] L. Guittet. Contribution à l'Ingénierie des IHM - Théorie des Interacteurs et Architecture H4 dans le système NODAOO. Thèse de Doctorat, Université de Poitiers, 1995.

- [Hayat, 2001] S. Hayat. Amélioration de la qualité des correspondances dans les réseaux de transports urbains. Rapport final du projet SART, 2001.
- [Helm *et al*., 2009] J. Helms, R. Schaefer, K. Luyten, J. Vermeulen, M. Abrams, J. Vanderdonckt, J. Vermeulen, M.Abrams. User Interface Markup Language (UIML) Version 4.0, Committee Specification 01, 01 May 2009. Accessible à : http://docs.oasis-open.org/uiml/v4.0/cs01/uiml-4.0-cs01.html
- [Henninger *et al*., 1995] S. Henninger, K. Heynes, M. Reith. A Framework for Developing Experience-Based Usability Guidelines, Conference Proceedings of DIS'95, University of Michigan, ACM Press, pp. 43-53, August 23-25 1995.
- [Henninger *et al*., 1997] S. Henninger, C. Lu, C. Faith. Using organisational learning techniques to develop context-specific usability Guidelines. In: Gerrit Van der Veer, Austin Henderson, Susan Coles, Conference Proceedings of Designing Interactive Systems: Processes, practices, methods and techniques (DIS '97), Amsterdam 18-20 August, NY, ACM Press, pp. 129-136, 1997.
- [Hilbert et Redmiles, 1998] D.M. Hilbert, D.F. Redmiles. Agents for collecting application usage data over the Internet. In Proceedings of Autonomous Agents '98, Proceedings of the second international conference on Autonomous agents, pp. 149-156, Minneapolis, Minnesota, United States, ISBN:0-89791-983-1, 1998.
- [Hilbert et Redmiles, 2000] D.M. Hilbert, D.F. Redmiles. Extracting usability information from user interface events. ACM Computing Surveys (CSUR), v.32, n.4, p.384-421, 2000.
- [HP, 1988] Common User Interface Behaviour Guide. Hewlett-Packard Company, Etats-Unis, septembre 1988.
- [Holyer, 1993] A. Holyer. Methods for evaluating User Interfaces. School of Cognitive and Computing Sciences. University of Sussex, Brighton, 1993.
- [Hong *et al*., 2001] J.I. Hong, J. Heer, S.Waterson, J.A. Landay. WebQuilt: A Proxy-based Approach to Remote Web Usability Testing. In: *ACM Transactions on Information Systems*, 19 (3), pp. 263-385, ISSN:1046-8188, July 2001.
- [Hochheiser et Shneiderman, 2001] H. Hochheiser, B. Shneiderman. Using interactive visualizations of WWW log data to characterize access patterns and inform site design. Journal of the American Society for Information Science and Technology, 52, 331-343, 2001.
- [Hudson, 1987] S.E. Hudson. UIMS Support for Direct Manipulation Interfaces. Computer Graphics, Avril 1987, Vol. 21, N 2, pp. 120-124.
- [Hulzebosch et Jameson, 1996] R. Hulzebosch et A. Jameson. FACE: A Rapid Method for Evaluation of User Interfaces. In: P. W. Jordan, B. Thomas, B. A. Weerdmeester, & I. L. McClelland (Eds.), Usability evaluation in industry, pp. 195-204, Publisher Taylor&Francis, London, 1996.
- [IBM, 1993] IBM. Object-Oriented Interface Design: IBM's Common User Access Guidelines. Publisher Que Pub, March 1993.
- [IEEE, 1990] Institute of Electrical and Electronics Engineers. IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. New York, NY: 1990.
- [ISO 9241, 1998] ISO 9241 : Ergonomic Requirements for Office Work with Visual Display Terminals. 1998
- [ISO/IEC 9126-1, 2001] ISO/IEC 9126-1: Software engineering -- Product quality -- Part 1: Quality model. 2001 (JIS X 0129-1: 2003)
- [ISO/IEC 9126-2, 2001] ISO/IEC 9126-2: Software engineering -- Product quality -- Part 2: External metrics, 2003.

- [ISO/IEC 9126-3, 2001] ISO/IEC 9126-3: Software engineering -- Product quality -- Part 3: Internal metrics, 2003.
- [ISO/IEC 9126-4, 2001] ISO/IEC 9126-4: Software engineering -- Product quality -- Part 4: Quality in use metrics, 2004.
- [Ivory, 2001] M. Ivory. An Empirical Foundation for Automated Web Interface Evaluation, PhD Thesis, University of California at Berkeley, 2001.
- [Ivory et Hearst, 2001] M. Ivory, M. Hearst. The State of the Art in Automated Usability Evaluation of User Interfaces, ACM Computing Surveys, 33 (4) , pp. 173-197, 2001.
- [Jasselette *et al*., 2006] A. Jasselette, M. Keita, M. Noirhomme-Fraiture, F. Randolet, J. Vanderdonckt, C.V. Brussel, D. Grolaux, Automated repair tool for usability and accessibility of web sites. In Proceedings CADUI 2006, Bucharest, Springer-Verlag, 2006.
- [Jensen, 1992] J. Jensen, Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use.Volume 1, Analysis Methods. Monographs in Theoretical Computer Science – Jensen – 1992.
- [Johannsen, 1995] G. Johannsen. Conceptual design of multi-human machine interfaces. Proceedings 6th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design and Evaluation of Man-Machine Systems, MIT, Cambridge USA, June 27-29, October 1995.
- [John et Kieras, 1996] B.E. John, D.E. Kieras. 1996. The GOMS family of user interface analysis techniques: Comparison and contrast. ACM Transactions on Computer-Human Interaction, 3, pp. 320-351, 1996.
- [Karsenty et Weikart, 1991] S. Karsenty, C. Weikart. Une extension de l'interface d'application dans le modèle de Seeheim. Communication aux journées IHM'1991, Dourdan, Décembre 1991.
- [Kinder, 2004] E. Kindler. 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.), Ekkart Kinder (ed.), University of Paderborn, November 8, 2004.
- [Kinder, 2005] E. Kindler, 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, 1997] C. Kolski. Interfaces homme–machine, application aux systèmes industriels complexes. Editions Hermes, Paris, 1997.
- [Lecerof et Paterno, 1998] A. Lecerof, F. Paterno. Automatic support for usability evaluation. IEEE Trans. on Software Engineering. 24 (10), 1998.
- [Lee et Hwang, 2004] S.K. Lee, C.S. Hwang. Architecture modeling and evaluation for design of agent-based system, The Journal of Systems and Software 72, pp. 195–208, 2004.
- [Le Parc et al., 2004] P. Le Parc, J. Vareille, L. Marcé. E-productique ou contrôle et supervision distante de systèmes mécaniques sur Internet. JESA, 38 (5), pp. 525-558, 2004.
- [Logan, 1998] B. Logan. Classifying Agent Systems Software Tools for Developing Agents. papers from the 1998 Workshop, J. Baxter and B. Logan (Eds.), Technical Report WS-98-10, AAAI Press, pp. 11-21,1998.
- [Luck et d'Inverno, 2001] M. Luck, M. d'Inverno. A Conceptual Framework for Agent Definition and Development. The Computer Journal, 44 (1), pp. 1-20, 2001.
- [Mandiau, 2000] R. Mandiau.: Modélisation et évaluation d'organisations multi-agents. Mémoire d'Habilitation à Diriger des Recherches, Université de Valenciennes et du Hainaut-Cambrésis, Décembre 2000.

- [Mandiau et Grislin-Le Stugeon, 2001] R. Mandiau, E. Grislin-Le Strugeon. Systèmes multi-agents. In: TI (Ed.), S 7216, pp. 1-17, Paris: Les Techniques de l'Ingénieur, 2001.
- [Marcus, 1991] A. Marcus. Graphic Design for Electronic Documents and User Interfaces. ACM Press Books Tutorial Series, New York, 1991.
- [Mariage, 2005] C. Mariage. MetroWeb : logiciel de support à l'évaluation de la qualité ergonomique des sites web. Thèse de Doctorat, UCL, Louvain-la-Neuve, 2005.
- [Mick *et al*., 2005] M. Philippon, A. Mille, G. Caplat. Aide à l'utilisateur : savoir quand intervenir. Proc. IHM'05 : 17ème conférence francophone sur l'interaction homme-machine, ACM ed., Toulouse, France, 2005.
- [Microsoft, 1995] Microsoft. The WindowsTM interface guidelines for software design. Redmond, Washington: Microsoft Press, 1995.
- [Millot, 1996] P. Millot. De la surveillance à la supervision : intégration des opérateurs humains. École d'été d'Automatique sur la surveillance des systèmes continus, Tome 1, Grenoble, France, 1996.
- [Moray, 1997] N. Moray. Human factors in process control. In: Salvendy, G. (Ed.), Handbook of Human Factors and Ergonomics. Wiley, New York, 1997, 1944–1971.
- [Moha *et al*., 2005] N. Moha, Q. Li, A. Gaffar, A. Seffah. Enquête sur les pratiques de tests d'utilisabilité. IHM'05, 17ème Conférence Francophone sur l'Interaction Homme-Machine, pp. 115-122, Toulouse, France, September 27-30, 2005.
- [Murch, 1987] G.M. Murch. Colour Graphics - Blessing or Ballyhoo?, Chapter 7: The Visual Channel. In: R.M. Baecker & W.A.S. Buxton (eds.), Readings in Human-Computer Interaction - A Multidisciplinary Approach, Morgan Kaufmann Publishers, Los Altos, pp. 333–341, 1987.
- [Nendjo Ella, 1999] A. N. Ella. Vers un outils d'aide à la décision en évaluation des systèmes interactifs et la prise en compte conjointe de critères techniques et socio-culturels. Mémoire de Doctorat, Université de Valenciennes et du Hainaut-Cambrésis, Valenciennes, janvier, 1999.
- [Nielsen, 1993] J. Nielsen. Usability Engineering. Boston, MA: Academic Press, 1993.
- [Nielsen, 1994] J. Nielsen. Usability Engineering, AP Professional, Cambridge, 1994.
- [Nielsen, 2000] J. Nielsen. Designing Web Usability: The Practice of Simplicity. Indianapolis, IN: New Riders Publishing, 2000.
- [Nigay, 1994] L. Nigay. Conception et modélisation logicielles des systèmes interactifs : application aux interfaces multimodales. Thèse de Doctorat, Université Joseph Fourier, Grenoble, janvier 1994.
- [Nigay et Coutaz, 1993], L. Nigay, J. Coutaz. A design space for multimodal systems: Concurrent processing and data fusion. In Proceedings of ACM Interchi'93 Conference on Human Factors in Computing Systems, Voices and Faces, pp. 172-178,1993.
- [Nigay et Coutaz, 1995] L. Nigay, J. Coutaz. A Generic Platform for Addressing the Multimodal Challenge. CHI'95, ACM Press, Denver, Colorado, United States, 1995.
- [OSF, 1993] Open Software Foundation. OSF/Motif Style Guide, Revision 1.2. London: Prentice-Hall, 1993.
- [Ouadou, 1994] Ouadou K., AMF : Un modèle d'architecture multi-agents multi-facettes pour Interfaces Homme-Machine et les outils associés. Thèse de Doctorat, Ecole Centrale de Lyon, 1994.
- [Parush, 2000] A. Parush. A Database Approach to Building and Using Online Human Computer Interaction Guidelines. In: J. Vanderdonckt and C. Farenc (editors), Tools for Working with Guidelines, pp,. 77-84, Springer-Verlag, London, 2000

- [Paganelli and Paternò 2002-2] L. Paganelli, F. Paternò. Intelligent Analysis of User Interactions with Web Applications. Proceedings of ACM IUI 2002, San Francisco, CA, January 2002.
- [Paganelli and Paternò, 2003] L. Paganelli, F. Paternò. Tools for remote usability evaluation of Web applications through browser logs and task models. Behavior Research Methods, Instruments, and Computers, 35 (3), pp. 369–378, 2003.
- [Paterno *et al*., 1997] F. Paterno, C.Mancini, S.Meniconi. ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models. Proceedings Interact'97, pp. 362-369, Sydney, Chapman&Hall, 1997.
- [Paterno et Ballardin, 1999] F. Paterno, G. Ballardin. Model-aided remote usability evaluation. In: Proceedings of the IFIP TC13 Seventh International Conference on Human-Computer Interaction, ed. by A. Sasse and C. Johnson, Edinburgh, Scotland. Amsterdam, The Netherlands: IOS Press, 1999.
- [Paterno et Ballardin, 2000 F. Paterno, G. Ballardin. RemUSINE: a bridge between empirical and model-based evaluation when evaluators and users are distant. Interacting with Computers, 13(2), pp. 229–251, 2000.
- [Paterno *et al*., 2006] F. Paterno, A. Piruzza, C. Santoro. Remote Web usability evaluation exploiting multimodal information on user behavior. In Proceedings CADUI 2006, Bucharest, Springer-Verlag, 2006.
- [Paterno et al., 2007] F. Paternò, A. Russino, C. Santoro. Remote evaluation of Mobile Applications. Task Models and Diagrams for User Interface Design 6th International Workshop, TAMODIA 2007, Toulouse, France, November 7-9, 2007, Lecture Notes in Computer Science , Vol. 4849, Winckler, Marco; Johnson, Hilary; Palanque, Philippe (Eds.) ISBN: 978-3-540-77221-7.
- [Patry, 1999] G. Patry. Contribution à la conception du dialogue Homme Machinedans les applications graphiques interactives de conception technique: le système GIPSE. Thèse de Doctorat, Université de Poitiers, 1999.
- [PEP, 2002] The PEP Tool. Accessible à : http://parsys.informatik.uni-oldenburg.de/~pep/. 2002/07/29.
- [Petri, 1962] C. A. Petri. Kommunikation mit Automaten. Schriften des IIM 3, 1-128, Universität Bonn, Institut für Instrumentelle Mathematik, Bonn, 1962.
- [Plaff, 1985] Pfaff, GE.: User interface management system. Springer-Verlag, 1985.
- [Rafla et al., 2007] T.Rafla, P.N. Robillard, M. Desmarais. A method to elicit architecturally sensitive usability requirements: its integration into a software development process. Software Quality Journal, 15(2), pp. 117-133, 2007.
- [Rasmussen, 1986] J. Rasmussen. Information Processing and Human–Machine Interaction, an Approach to Cognitive Engineering, Elsevier Science Publishing, Amsterdam, 1986.
- [Ratner et al.,1996] J. Ratner, Eric M. Grose, C.Forsythe. 1996. Characterization and assessment of HTML style guides. In: Proceedings of the Conference on Human Factors in Computing Systems, volume 2, pp. 115-116, Vancouver, Canada. New York, NY: ACM Press, 1996.
- [Ravden et Johnson, 1989] S.J. Radven, G.I. Johnson. Evaluating usability of human-computer interfaces : a practical method. Ellis Horwood, Chicester, 1989.
- [Reiman *et al*., 1991] J. Reiman, S. Davies, C. Hair, M. Esemplare, E. Polson, and C. Lewis. An automated cognitive walkthrough. In: Robertson, S.E, Olsen, G.M., and Olsen, J.S. (Eds.), Human Factors in Computing Systems, CHI 91 Conference Proceedings (1991). ACM Press, New York.
- [Renew, 2002] Renew: The Reference Net Workshop. Accessible à : http://www.renew.de. 2002/03/04.

- [Robert, 2003] J.M. Robert. Que faut- il savoir sur l'utilisateur pour concevoir des interfaces de qualité ? Dans Boy, G.A. (Ed.). L'ingénierie cognitive : IHM et Cognition. Hermès, Paris, 2003.
- [Russell et Norvig, 1995] S. Russell, P. Norvig. Artificial Intelligence: a Modern Approach. Prentice Hall Series in Artificial Intelligence. Englewood Cliffs, New Jersey, 1995.
- [SART, 2007] Système d'Aide à la Régulation de Trafic du réseau de transport valenciennois et de ses pôles d'échanges. Rapport final, projet coopératif SART, INRETS, France, Déc. 2007
- [Scapin, 1986] D.L. Scapin. Guide ergonomique de conception des interfaces homme-ordinateur. Rapport INRIA N°77, Institut National de Recherche en Informatique et en Automatique, Le Chesnay, octobre 1986.
- [Scapin, 1989] D.L. Scapin. Guidelines for User Interface Design : Knowledge Collection and Organization. Rapport ITHACA.INRIA.89.D12.03, Institut National de Recherche en Informatique et en Automatique, Rocquencourt, 30 décembre 1989.
- [Scholtz et Laskowski, 1998] J. Scholtz, S. Laskowski. 1998. Developing usability tools and techniques for designing and testing web sites. In: Proceedings of the 4th Conference on Human Factors & the Web, Basking Ridge, NJ. Accessible à : http://www.research.att.com/conf/hfweb/proceedings/scholtz/index.html.
- [Salber, 1995] D. Salber. De l'interaction homme-machine individuelle aux systèmes multi-utilisateurs. L'exemple de la communication homme-homme médiatisée. Thèse de Doctorat, Université Joseph Fourier - Grenoble 1, Septembre 1995.
- [Sears, 1995] A. Sears. AIDE: A Step Toward Metric-Based Interface Development Tools. Proc. UIST'95, New York: CM Press, 1995.
- [Seffah et al.2001] A.Seffah, M.Donyaee and R.Kline. Usability And Quality in Use Measurement And Metrics: An Integrative Model. 2001, Accessible à : http://hci.cs.concordia.ca/www/hcse/projects/humancase/QUIM.pdf
- [Seffah, et al.2006] A. Seffah, M. Donyaee, R.B. Kline, H.K. Padda. Usability measurement and metrics: A consolidated model. Software Quality Journal, 14 (2), June 2006.
- [Sheridan, 1984] T.B. Sheridan. Supervisory control of remote manipulators, vehicules and dynamic processes: Experiment I command and display aiding. Advances in Man-machine System research, vol. 1, 1984.
- [Sheridan, 1985] T.B. Sheridan. Forty-five years of man-machine system: history and trends. 2nd IFAC Congress: Analysis, design and Evaluation of Man-Machine Systems, Varese, September 1985.
- [Sheridan, 1988] T.B. Sheridan. Task allocation and supervisory control. In: Helander, M. (Ed.), Handbook of Human–Computer Interaction. Elsevier Science Publishers B.V., North-Holland, Amsterdam, 1988
- [Seffah *et al*., 2008] A. Seffah, M. Taleb, H.H. Mammar, A. Abran. Reconciling usability and interactive s*ystem architecture using patterns. Journal of Systems and Software, 81(11), pp 1845-1852, 2008.*
- [Senach, 1990] B. Senach. Evaluation ergonomique des IHM : Une revue de la littérature. Rapport INRIA n°1180, mars, 1990.
- [Sidi, 2006] Mohamed Mahmoud Ould Sidi, Contribution à l'amélioration des systèmes d'aide à la décision dans le domaine du transport, Thèse de Doctorat, Ecole Centrale de Lille, 2006
- [Simonin, 2001] O. Simonin. Le modèle satisfaction-altruisme : coopération et résolution de conflits entre agents situés réactifs, application à la robotique. Thèse de Doctorat, Université de Montpellier 2, Décembre 2001.

- [Smith et Moisier, 1986] S.L. Smith, J.N. Moisier. Guidelines for designing user interface software. Report No. MTR-10090, ESD-TR-86-278, Bedford, MA:MITRE Corp,1986.
- [Tarpin-Bernard, 1997a] F. Tarpin-Bernard. Travail coopératif synchrone assisté par ordinateur : Approche AMF-C. Thèse de Doctorat, Ecole Centrale de Lyon, juillet 1997.
- [Tarpin-Bernard, 1997b] F. Tarpin-Bernard, B. David. AMF : a new design pattern for complex interactive software? Proceedings of International HCI'97, San Francisco, 351-354, 1997.
- [Tarpin-Bernard, 1997c] F. Tarpin-Bernard, B. David. Expression and Recognition of Design Intentions - The contributions of multimodality. Integrated Design and Manufacturing in Mechanical Engineering, Eds Chedmail & al., Kluwers, pp. 113-120, 1997.
- [Tarpin-Bernard et David, 1999] F. Tarpin-Bernard, B. David. AMF : un modèle d'architecture multi-agents multi-facettes, TSI, 18 (5) , pp. 555-586, 1999.
- [Taylor et Coutaz, 1994] R.M. Taylor, J. Coutaz. Workshop on Software Engineering and Human-Computer Interaction: Joint Research Issues. In: Proceedings of the International Conference on Software Engineering '94, Sorrento, Italy, May 1994.
- [Texier, 2000] G. Texier. Contribution à l'ingénierie des systèmes interactifs : Un environnement de conception graphique d'applications spécialisées de conception. Thèse de Doctorat, Université de Poitiers, novembre 2000.
- [The International Academy of Arts and Sciences, 2000] The International Academy of Arts and Sciences, 2000. The webby awards 2000 judging criteria. Accessible à http://www.webbyawards.com/judging/criteria.html.
- [Thevenin et Coutaz, 1999] D. Thevenin, J. Coutaz. Plasticity of User Interfaces: Framework and Research Agenda, In Proceedings of INTERACT'99, pp. 110-117, 1999.
- [Thevenin, 2001] Thevenin D. Adaptation en Interaction Homme-Machine : Le cas de la Plasticité. Thèse de Doctorat, Université Joseph Fourier, Décembre 2001.
- [Trabelsi *et al*., 2004] A. Trabelsi,, H. Ezzedine, C. Kolski. Architecture modelling and evaluation of agent-based interactive systems, In: Proc. IEEE SMC 2004, The Hague, October, 2005, 5159-5164.
- [Trabelsi, 2006] A. Trabelsi. Contribution à l'évaluation des systèmes interactifs orientés agents, application à un poste de supervision de transport urbain. Thèse de Doctorat, Université de Valenciennes et du Hainaut-Cambrésis, Septembre 2006.
- [TRAN *et al*., 2007] C.D. TRAN, H. Ezzedine, C. Kolski. Towards a generic and configurable model of an electronic informer to assist the evaluation of agent-based interactive systems. J. Cardoso, J. Cordeiro, J. Filipe (Ed.), ICEIS'2007, Ninth International Conference on Enterprise Information Systems (Funchal, Portugal, June 12-16, 2007), Proceedings Human-Computer Interaction, INSTICC, pp. 290-293, juin, ISBN 978-972-8865-92-4.
- [TRAN *et al*., 2008a] C.D. TRAN, H. Ezzedine, C. Kolski. A generic and configurable electronic informer to assist the evaluation of agent-based interactive systems. 7th international conference on Computer-Aided Design of User Interfaces, CADUI'2008, Albacete, Spania, juin 2008.
- [TRAN *et al*., 2008b] C.D. TRAN, H. Ezzedine, C. Kolski. Evaluation of agent-based interactive systems: proposal of an electronic informer using Petri Nets. Journal of Universal Computer Science, 14 (19), pp.3202-3216, 2008.
- [UIMS, 1992] UIMS. The UIMS Workshop Tool Developers. A metamodel for the runtime architecture of an interactive system. SIGCHI Bulletin, 24(1), pp32–37, 1992.
- [Vanderdonckt, 1994] J.Vanderdonckt. Guide ergonomique des interfaces homme-machine. Facultés universitaires Notre-Dame de la Paix à Namur (Belgique), Presses Universitaires de Namur, Namur, 1994.

- [W3C, 1999] W3C Web Content Accessibility Guidelines 1.0. Accessible à : http://www.w3c.org/TR/WCAG10.
- [Waterson, 2002] S.Waterson. WebQuilt: A Visual Analysis Tool for Understanding Web Usability Clickstream Data. Masters Report, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA. May 2002.
- [Whitefield *et al*., 1991] A. Whitefield, F. Wilson, J. Dowell. A framework for human factors evaluation. Behaviour & Information Technology, 10 (1), pp. 65-79, 1991.
- [Wood et Barbacci, 1999] S.G. Woods, M.R. Barbacci. Architectural evaluation of collaborative agent-based systems. Technical Report CMU/SEI-99-TR-025, CMU, 1999.
- [Wooldridge et Jennings, 1995] M. Wooldridge, N.R. Jennings. Intelligent Agents: Theory and Practice, The Knowledge Engineering Review, 10 (2), pp. 115-152, 1995.
- [Wooldridge et Jennings, 1999] M. Wooldridge, N.R. Jennings. Software engineering with agents: pitfalls and pratfalls, IEEE Internet Comput, pp. 20–27, May–June 1999.
- [Zettlemoyer et al.1999] L.S. Zettlemoyer., R.S. Amant, M.S. Dulberg. IBOTS: Agent control through the user interface. In: Proceedings of the International Conference on Intelligent User Interfaces, 31-37, Redondo Beach, CA. New York, NY: ACM Press, 1999.

Annexe 1 : PNML (Petri Nets Markup Language)

Sommaire

1. Modélisation

PNML (Petri Net Markup Language) est un langage au format compatible XML pour décrire les Réseaux de Pétri (RdPs). PNML évolue régulièrement. La version la plus récente est la version 2009. Cette annexe a pour objectif d'introduire une vue d'ensemble et les concepts de base du PNML. Les lecteurs intéressés peuvent consulter [Billington *et al*., 2003 ; Kinder, 2004 ; 2005 ; http://www.pnml.org] pour les détails.

La figure A1.1 illustre la structure de PNML. Les termes standard sont en anglais, donc nous gardons ces termes anglais sous chaque dénomination en français sur cette figure. La structure de PNML est composée de trois niveaux. Le niveau le plus haut utilise les concepts définis dans le niveau le plus bas. Notre environnement d'évaluation génère les RdPs sous forme de documents PNML au niveau le plus haut.

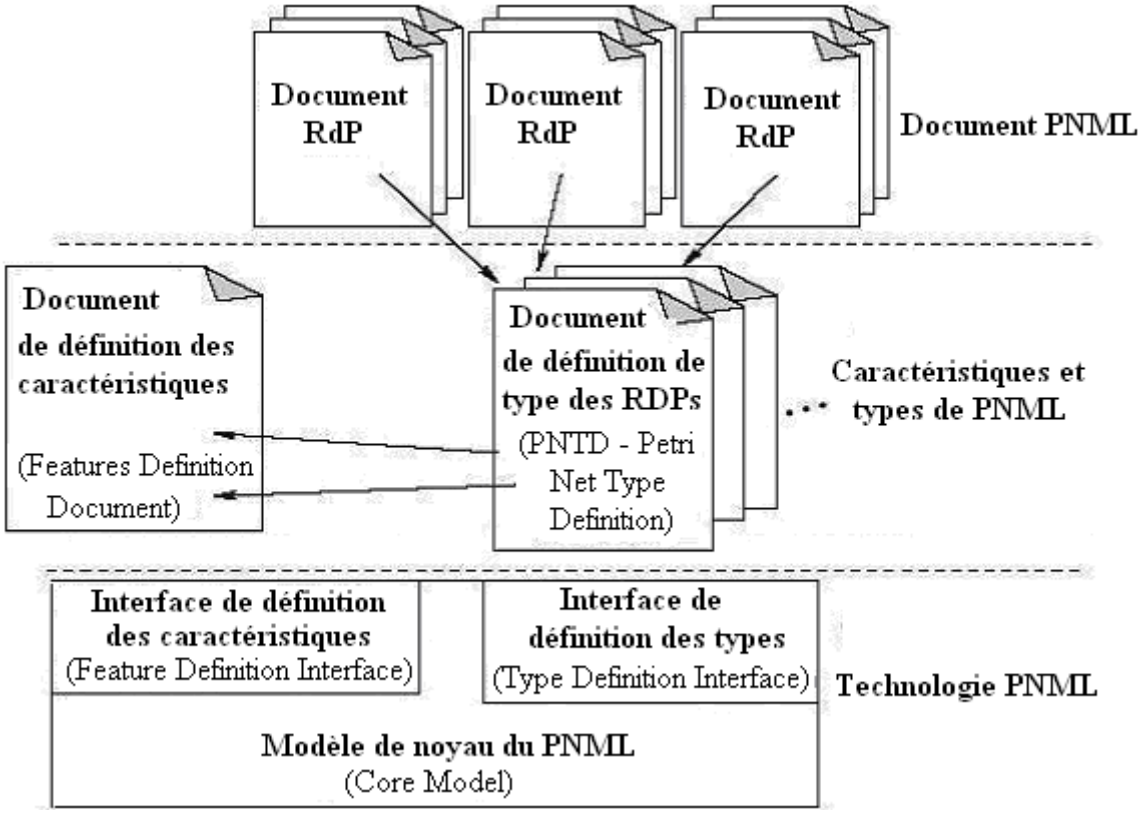

Figure A1.1 : Structure de la technologie PNML [Kinder, 2004]

PNML doit prendre en compte les différentes variantes et versions des RdPs (y compris les variantes et les versions futures). Afin d'obtenir cette flexibilité, PNML considère un RdP comme un graphe orienté étiqueté (labelled directed graph). Les nœuds de ce graphe sont les transitions ou les places du RdP, les arcs entre les nœuds de ce graphe correspondent aux arcs du RdP. Dans ce graphe, on trouve les labels qui sont attachés aux objets du graphe (nœuds, arcs, etc.) ou au graphe lui-même. Chaque label peut être associé à un nœud ou à un arc du réseau ou au réseau lui-même. Toutes les informations spécifiques qui concernent le type des RdPs sont représentées par ces labels. Cette structure basique est définie dans le modèle de noyau de PNML (le niveau le plus bas sur la figure A1.1 ci-dessus). Dans cette annexe, nous présentons seulement ce modèle de noyau de PNML parce qu'il concerne la structure du graphe des concepts basiques de PNML qui sont communs pour tous les types des RdPs.

La figure A1.2 montre le modèle de noyau de PNML sous forme d'un diagramme de classes UML. Dans les documents PNML, tous les éléments sont rédigés en anglais ; donc nous gardons ces mots anglais dans la suite de cette description.

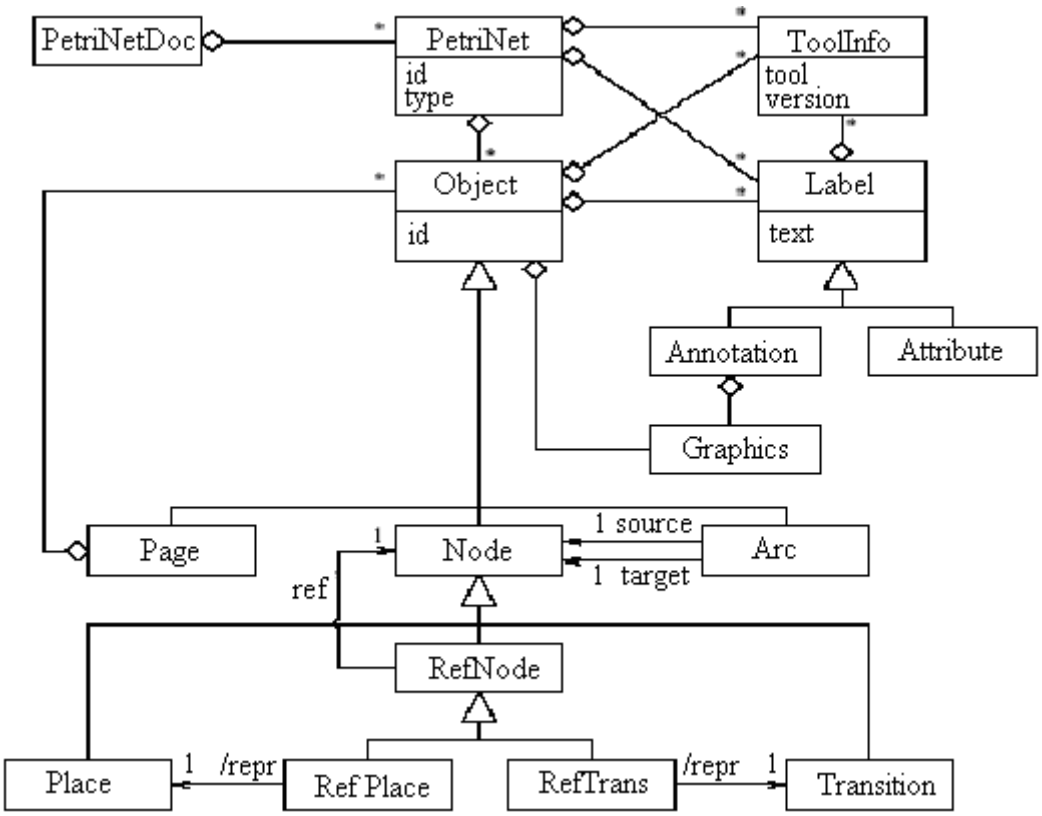

Figure A1.2 : Modèle de noyau (core model) de la technologie PNML sous forme d'un diagramme de classes UML [Kinder, 2004]

Le modèle de noyau définit la structure générale d'un graphe (RdP) et tous les *labels* qui sont susceptible d'être attachés aux objets du graphe ou au graphe lui-même. Nous expliquons maintenant quelques concepts de base (représentés sous forme de classes du diagramme UML de la figure A1.2) de ce modèle :

- *PetriNet* : cette classe (cf. figure A1.2) représente un RdP.
- *PetriNetDoc* : cette classe représente un document contenant un ou plusieurs RdPs
- *Object :* cette classe *Object* représente un objet qui peut être une *place*, une *transition* ou un *arc* d'un RdP. Un *arc* fait la liaison entre deux objets (une source, un destinataire). Plusieurs objets composent un RdP. Comme l'illustre la figure A1.2, chaque objet est identifié par un identificateur (*id*) et contient des informations graphiques (classe *graphics*) concernant la position, la taille, la couleur, etc., de cet objet. Notons que des objets comme *Page*, *RefPlace*, etc., concernent une version de PNML appelée PNML structuré, pour décrire un graphe d'un grand RdP selon plusieurs parties plus petites. Nous nous intéressons seulement aux PNML basiques ; donc ces objets ne sont pas présentés ici.
- *Label* : Comme nous l'avons abordé ci-dessus, PNML considère un RdP comme un graphe orienté étiqueté (labelled directed graph) où les *labels* sont attachés aux objets du graphe ou au graphe lui-même. Comme l'illustre la figure A1.2, un objet d'un RdP

(représenté par la classe *object*) ou un RdP lui-même (représenté par la classe *PetriNet*) peut être associé à un ou plusieurs *labels*. Les *labels* sont associés à un objet du graphe (ou au graphe lui-même) pour associer une signification à cet objet (ou au graphe). Par exemple, les déclarations d'un graphe de RdPs de haut niveau sont représentés sous forme de *labels*. Toutes les informations spécifiques qui concernent le type des RdPs sont représentées par les *labels*. Un *label* d'un objet du graphe peut être le nom de cet objet (par exemple, le nom d'une *place*, d'une *transition*, etc.) ou le « *transition guard* » d'une *transition* (qui est une expression de type Boolean attachée à cette *transition*) dans un RdP coloré, l'information temporelle dans un RdP temporisé, etc.

En guise d'illustration, nous présentons maintenant un RdP simple qui contient seulement une place, une transition et un arc comme la figure A1.3. Cet exemple est extrait de [Billinton *et al*., 2003 ; Kinder, 2004].

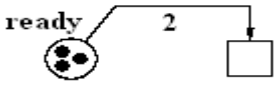

Figure A1.3 : Exemple d'un RdP simple [Billinton *et al*., 2003 ; Kinder, 2004]

La description PNML de ce RdP est la suivante :

```
<pnml xmlns="http://www.example.org/pnml">
    <net id="n1" type="http://www.example.org/pnml/PTNet">

        <name> <text> An example P/T-net </text> </name>

        <place id="p1">
            <graphics> <position x="20" y="20"/> </graphics>
            <name>
                <text> ready </text>
               <graphics>
                   <offset x="-10" y="-8"/>
               </graphics>
             </name>
            <initialMarking>  <text> 3 </text>  </initialMarking>
        </place>

        <transition id="t1">
            <graphics>   <position x="60" y="20"/>  </graphics>
           <toolspecific tool="PN4all" version="0.1"><hidden/></toolspecific>
        </transition>

        <arc id="a1" source="p1" target="t1">
            <graphics>
                <position x="30" y="5"/>
                <position x="60" y="5"/>
            </graphics>
           <inscription>
               <text> 2 </text>
              <graphics> <offset x="15" y="-2"/> </graphics>
           </inscription>
        </arc>

    </net>
</pnml>
```

Les éléments XML de la description PNML ci-dessus correspondent aux concepts du modèle de noyau de PNML. Le tableau A1.1 illustre cette correspondance :

Concepts du modèle de noyau	Éléments XML correspondants
PetriNetDoc	<pnml>
PetriNet	<net>
Place	<place>
Transition	<transition>
Arc	<arc>
Graphics	<graphics>
Label	<name>,<text>,<initialMarking>,<inscription>, etc.

Tableau A1.1 : Correspondance entre les concepts du modèle de noyau et les éléments XML

On peut constater que les *labels* sont associés aux objets du RdP ou au RdP lui-même. Dans cet exemple :

- Un *label* sous forme de l'élément XML ***<name>*** est associé à la seule *place* du RdP. Ce *label* contient un autre *label* sous forme de l'élément XML ***<text>*** pour représenter le nom « *Ready* » de cette *place*.
- Un *label* sous forme de l'élément XML ***< initialMarking >*** est associé à la seule *place* du RdP. Ce *label* contient un autre *label* sous forme de l'élément XML ***<text>*** pour représenter le nombre de jetons dans cette *place*.
- Un *label* sous forme de l'élément XML ***< name >*** est associé au RdP lui-même. Ce label contient un autre *label* sous forme de l'élément XML ***<text>*** pour représenter le nom « *An example P/T-net* » de ce RdP.
- Un *label* sous forme de l'élément XML ***<inscription>*** est associé au seul *arc* du RdP. Ce *label* contient un autre *label* sous forme de l'élément XML ***<text>*** pour représenter le poids « 2 » de cet *arc*. Les attributs « ***Source*** » et « ***Target*** » de *l'arc* du RdP prennent les valeurs « *p1* » et « *t1* » qui sont les identificateurs de la seule *place* et de la seule *transition* de ce RdP.

On peut se poser plusieurs questions : « *D'où viennent les labels ?* » « *Qui a défini ces labels ?* ». En fait, il n'y a aucune restriction sur les *labels* dans le modèle de noyau dans le but de représenter n'importe quel type de RdPs. De plus : « *Dans un type de RdP quelconque, quels sont les labels légauxs et comment combiner des labels d'une manière légale ?* ».

Les deux interfaces au niveau le plus bas sur la figure A1.1 (*l'interface de définition des caractéristiques* (*Feature Definition Interface*) et *l'interface de définition des types* (*Type Definition Interface*)) peuvent apporter des réponses à ces questions.

a) Interface de définition des caractéristiques (Feature Definition Interface) :

Cette interface permet de définir les *labels*. En effet, elle décrit la manière de définir la structure d'un *label* et elle définit la structure d'un document contenant les définitions des *labels*. Le *Document de définition des caractéristiques (features definition document)* au niveau intermédiaire (cf. figure A1.1) est celui contenant plusieurs définitions des *labels*. Par exemple, le *label* sous forme de l'élément XML <*initialMarking*> (associé à la seule *place* du RdP de la figure A1.3, afin de représenter le nombre de jetons de cette *place*) est défini comme suit, l'exemple étant tiré de [Kinder, 2004] :

```
<define name="PTMarking"
xmlns:pnml="http://www.informatik.hu-berlin.de/top/pnml">
    <element name="initialMarking">
        <interleave>
            <element name="text">
                <data type="nonNegativeInteger"
                datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"/>
```

```
            </element>
            <ref name="pnml:StandardAnnotationContent"/>
        </interleave>
    </element>
</define>
```

D'après cette definition, ce *label* s'appelle "***PTMarking***" et il doit être représenté par l'élément XML ***<initialMarking>*** qui peut contenir l'élément XML ***<text>***. Cet élément ***<text>*** doit contenir un nombre entier positif. Notons que ce *label* a déjà été utilisé dans l'exemple du RdP de la figure A1.3.

Ce *label* "***PTMarking***" est un *label* standard. Le *Document de définition des caractéristiques (features definition document)* au niveau intermédiaire contient plusieurs définitions similaires des *labels* standards pour garantir la compatibilité entre les différents types des RdPs [Billington *et al*., 2003]. On peut mentionner quelques *labels* standards comme ***HLMarking***, ***HLSort***, ***name*** (cf. celui-ci en figure A1.3), etc. Ce *Document de définition des caractéristiques* contient les définitions de ces *labels* standards, il est donc appelé aussi *document des conventions* (*conventions document*). Ce document évolue de temps en temps. On peut y ajouter les nouvelles définitions des *labels* ou modifier ou supprimer les définitions existantes.

b) Interface de définition des types (Type Definition Interface) :

En réalité, il y a plusieurs variantes et versions des RdPs mais ils ont la même structure fondamentale. Les spécificités d'un type de RdPs sont représentées par les *labels* qui sont attachés aux objets du graphe. Autrement dit, on distingue les types de RdPs via les *labels* attachés à leurs objets. Chaque type de RdP a sa propre manière d'attacher les labels à ses objets. Cette *Interface de définition des types* permet de définir le format d'un type particulier de RdPs. En effet, elle décrit la manière de définir les *labels* légaux d'un type particulier de RdPs. Ces *labels* légaux sont associés aux différents objets des RdPs de ce type ou aux RdPs eux-mêmes. Cette interface définit aussi la structure d'un document qui définit un type de RdPs. Les *Documents de définition de type de RdPs (PNTD-Petri Net Type Definition)* au niveau intermédiaire sur la figure A1.1 sont bien des documents. Ils contiennent les définitions des types de RdPs. On trouve ci-dessous un exemple extrait de [Kinder, 2004] :

```
<grammar ns="http://www.example.org/pnml"
    xmlns="http://relaxng.org/ns/structure/1.0"
    xmlns:conv="http://www.informatik.hu-berlin.de/top/pnml/conv">
    <include href="http://www.informatik.hu-berlin.de/top/pnml/pnml.rng"/>
    <include href="http://www.informatik.hu-berlin.de/top/pnml/conv.rng"/>
    <define name="NetType" combine="replace">
        <text>http://www.example.org/pnml/PTNet</text>
    </define>
    <define name="Net" combine="interleave">
        <optional><ref name="conv:Name"/></optional>
    </define>
    <define name="Place" combine="interleave">
        <interleave>
            <optional><ref name="conv:PTMarking"/></optional>
            <optional><ref name="conv:Name"/></optional>
        </interleave>
    </define>

    <define name="Arc" combine="interleave">
        <optional><ref name="conv:PTArcInscription"/></optional>
    </define>
</grammar>
```

D'après cette définition :

- Les *places* d'un RdP de ce type et le RdP lui-même sont autorisées à contenir un *label* qui s'appelle ***Name***.
- Les *places* d'un RdP de ce type sont aussi autorisées à contenir un *label* qui s'appelle ***PTMarking***.
- Les *arcs* d'un RdP de ce type sont autorisés à contenir un *label* qui s'appelle ***PTArcInscription***.
- Tous ces *labels* sont optionnels. Dans la description XML ci-dessus, on peut voir le mot « *conv:* » avant le nom de chaque *label* (par exemple, *conv:**Name***). Il signifie que les définitions de ces *labels* sont présentées dans le *document des conventions* (*conventions document*) qui est le *Document de définition des caractéristiques (features definition document)* au niveau intermédiaire (cf. figure A1.1). Ce document contient les définitions des *labels* standards comme déjà abordé ci-dessus ; par exemple, il contient la définition du *label* standard ***PTMarking*** que les *places* d'un RdP de ce type sont autorisées à contenir. La définition de ce *label* a déjà été présentée ci-dessus.

Les *Documents de définition de type des RdPs (PNTD-Petri Net Type Definition)* évoluent de temps en temps. On peut élaborer les nouveaux *Documents de définition de type deRdPs (PNTD-Petri Net Type Definition)* pour définir les nouveaux types des RdPs ou modifier les documents existants.

2. Outils

Pour l'instant, il y a plusieurs outils qui soutiennent PNML ; par exemple : PEP [PEP, 2002], Design/CPN [Design/CPN, 2001], CPN Tools [CPN, 2001]. Nous utilisons l'outil Renew [Renew, 2002] pour ouvrir les RdPs générés par l'environnement d'évaluation EISEval. La figure A1.4 montre une capture d'écran d'un RdP généré par cet environnement.

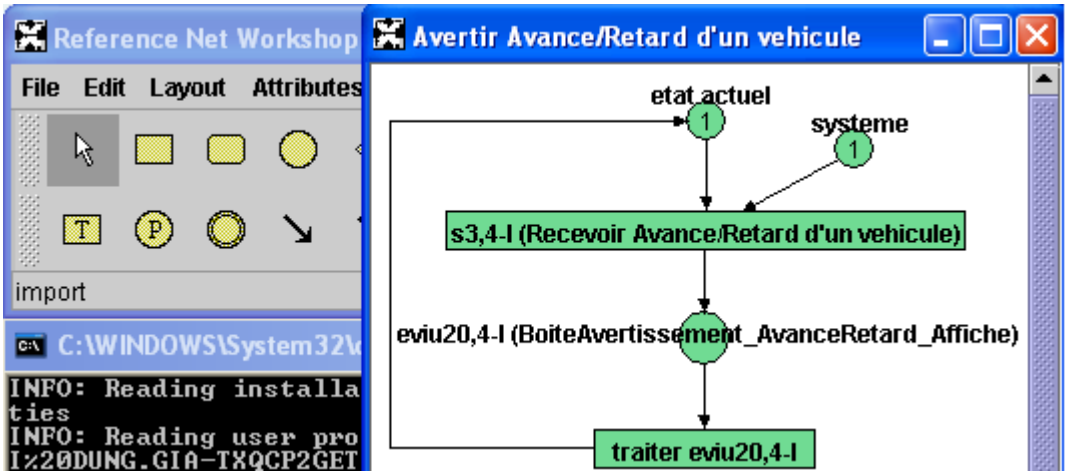

Figure A1.4 : Exemple d'un RdP généré par l'environnement d'évaluation EISEval

Ce RdP décrit la réalisation d'une *tâche* automatique consistant à avertir l'utilisateur de l'avance ou du retard d'un véhicule. Voici le contenu du document PNML qui correspond à ce RdP et qui est généré par l'environnement (les objets en gras sont les éléments visualisés sur la figure A1.4).

```
<pnml xmlns="http://www.informatik.hu-berlin.de/top/pntd/ptNetb">

    <net id="netID1"
        type="http://www.informatik.hu-berlin.de/top/pntd/ptNetb">
        <name><text>Avertir Avance/Retard d'un vehicule</text></name>

        <place id="1">
            <initialMarking>
                <graphics><offset x="0" y="0"/></graphics>
                <text>1</text>
            </initialMarking>

            <name>
                <graphics><offset x="3" y="-13"/></graphics>
                <text>etat actuel</text>
            </name>

            <graphics>
                <position x="413" y="74"/>
                <dimesion x="46" y="44"/>
                <fill color="rgb(112,219,147)"/>
                <line color="rgb(0,0,0)"/>
            </graphics>
        </place>

        <place id="2">
            <initialMarking>
                <graphics><offset x="0" y="0"/></graphics>
                <text>1</text>
            </initialMarking>

            <name>
                <graphics><offset x="3" y="-13"/></graphics>
                <text>systeme</text>
            </name>

            <graphics>
                <position x="486" y="145"/>
                <dimesion x="46" y="44"/>
                <fill color="rgb(112,219,147)"/>
                <line color="rgb(0,0,0)"/>
            </graphics>
        </place>

        <place id="6">
            <name>
                <graphics> <offset x="3" y="-13"/></graphics>
                <text>
                    eviu20,4-I (BoiteAvertissement_AvanceRetard_Affiche)
                </text>
            </name>

            <graphics>
                <position x="413" y="262"/>
                <dimesion x="46" y="44"/>
                <fill color="rgb(112,219,147)"/>
                <line color="rgb(0,0,0)"/>
            </graphics>
        </place>
```

```
<transition id="3">
    <name>
        <graphics><offset x="1" y="1"/></graphics>
        <text>
            s3,4-I (Recevoir Avance/Retard d'un vehicule)
        </text>
    </name>

    <graphics>
        <position x="413" y="216"/>
        <dimesion x="740" y="19"/>
        <fill color="rgb(112,219,147)"/>
        <line color="rgb(0,0,0)"/>
    </graphics>
</transition>

<transition id="8">
    <name>
        <graphics><offset x="1" y="1"/></graphics>
        <text>traiter eviu20,4-I</text>
    </name>

    <graphics>
        <position x="413" y="333"/>
        <dimesion x="370" y="19"/>
        <fill color="rgb(112,219,147)"/>
        <line color="rgb(0,0,0)"/>
    </graphics>
</transition>

<arc id="4" source="2" target="3">
    <graphics><line color="rgb(0,0,0)" style="solid"/></graphics>
</arc>

<arc id="5" source="1" target="3">
    <graphics><line color="rgb(0,0,0)" style="solid"/></graphics>
</arc>

<arc id="7" source="3" target="6">
    <graphics><line color="rgb(0,0,0)" style="solid"/></graphics>
</arc>

<arc id="9" source="6" target="8">
    <graphics><line color="rgb(0,0,0)" style="solid"/></graphics>
</arc>

<arc id="10" source="8" target="1">
    <graphics><line color="rgb(0,0,0)" style="solid"/></graphics>
</arc>

</net>

</pnml>
```

Annexe 2 : BSA (Base de spécification des agents)

Sommaire

Cette annexe présente la BSA qui concerne la description de chaque agent du SAI ainsi que les EVIUs (événements interfaces utilisateurs) et les services associés. Pour configurer l'environnement EISEval afin d'évaluer le système SAI, la BSA correspond à une partie des informations que l'évaluateur doit saisir en utilisant le module 7 (cf. 3.5.6, chapitre 3).

1. Description des agents du SAI

Le SAI est composé des 6 agents *interface* que nous avons présentés dans ce mémoire. Le tableau A2.1 montre la description de ces agents. Le symbole (*) représente l'information obligatoire, c'est-à-dire que l'évaluateur est obligé de la saisir pour configurer l'environnement. Les autres informations sans (*) sont optionnelles.

Tableau A2.1 : BSA – Description des agents

Identificateur (ID) de l'agent (*)	**Nom de l'agent (*)**	**Description de l'agent**	**Liste des IDs des EVIUs associés**	**Liste des IDs des services associés**
1-I	Etat_Trafic	Etat courant des véhicules (retard, avance)	{eviu1,1-I, eviu2,1-I, …,eviu5,1-I}	{s1,1-I, s2,1-I, s3,1-I, s4,1-I}
2-I	Etat_Ligne	Supervision d'une ligne	{eviu1,2-I, eviu2,2-I, …,eviu6,2-I}	{s1,2-I, s2,2-I,…, s6,2-I}
3-I	Station	Représentation d'une station	{eviu1,3-I, eviu2,3-I, …,eviu9,3-I}	{s1,3-I, s2,3-I, s3,3-I}
4-I	Véhicule	Représentation d'un véhicule	{eviu1,4-I, eviu2,4-I, ...,eviu25,4-I}	{s1,4-I, s2,4-I,…, s7,4-I}
5-I	Message	Vue et fonctionnalités concernant l'envoi des messages aux véhicules ou aux stations	{eviu1,5-I, eviu2,5-I, ...,eviu15,5-I}	{s1,5-I, s2,5-I,…, s8,5-I}
6-I	Vue_Globale	Vision globale du trafic dans le réseau	Non	Non

Nous présentons maintenant la description des EVIUs et des services de chaque agent.

2. Description des EVIUs et services des agents

2.1. Agent *interface Etat_Trafic* (1-I)

Les tableaux A2.2 et A2.3 montrent la description des EVIUs et des services de cet agent.[30]

Tableau A2.2 : BSA – Description des EVIUs de l'agent *Etat_Trafic*

ID de l'eviu (*)	**Nom de l'eviu (*)**	**Description de l'EVIU**	**Déclencheur ; IDs des EVIUs déclenchés ; IDs des services déclenchés**
eviu1,1-I	Panel_Ligne_Click	Clic sur un *panel* d'une ligne afin de l'afficher	Utilisateur ; Non ; s1,1-I
eviu2,1-I	Bouton_ZoomAV_Click	Clic sur le bouton *Zoom Avant*	Utilisateur ; Non ; s2,1-I
eviu3,1-I	Bouton_ZoomAR_Click	Clic sur le bouton *Zoom arrière*	Utilisateur ; Non ; s3,1-I
eviu4,1-I	checkbox_ligne_click	Clic sur un *checkbox* d'une ligne afin de modifier la vue des lignes du trafic	Utilisateur ; Non ; s4,1-I
eviu5,1-I	Vue_Etat_Traffic_change	La vue des lignes du trafic est changée	s4,1-I ; Non ; Non

[30] La dernière colonne de ces deux tableaux est décrite sous forme d'une liste dont les éléments sont séparés par des caractères point-virgule (;). Ces éléments sont : l'événement déclencheur de cet EVIU (ou ce service) ; Identificateurs des EVIUs qui sont déclenchés par cet EVIU (ou ce service) ; Identificateurs des services déclenchés par cet EVIU (ou ce service). S'il n'y a pas d'EVIUs ou services déclenchés par cet EVIU (ou ce service), alors le mot « Non » est utilisé pour le préciser.

Tableau A2.3 : BSA – Description des services de l'agent *Etat_Trafic*

ID du service (*)	Nom du service (*)	Description du service	Déclencheur ; IDs des EVIUs déclenchés ; IDs des services déclenchés
s1,1-I	Afficher une ligne	Exécution des traitements afin de montrer une ligne	eviu1,1-I ; Non ; s1,2-I
s2,1-I	Agrandir toutes les lignes	Agrandir l'écran de la vue des lignes	eviu2,1-I ; Non ; s2,2-I
s3,1-I	Réduire toutes les lignes	Réduire l'écran de la vue des lignes	eviu3,1-I ; Non ; s4,2-I
s4,1-I	Modifier la vue des lignes du trafic	Exécution des traitements afin de modifier la vue des lignes du trafic	eviu4,1-I ou s2,4-I ; eviu5,1-I ; Non

2.2. Agent *interface Etat_Ligne* (2-I)

Les tableaux A2.4 et A2.5 montrent la description des EVIUs et des services de cet agent.

Tableau A2.4 : BSA – Description des EVIUs de l'agent *Etat_Ligne*

ID de l'eviu (*)	Nom de l'eviu (*)	Description de l'eviu	Déclencheur ; IDs des EVIUs déclenchés ; IDs des services déclenchés
eviu1,2-I	PopupMenuItem_ZoomAV_Click	Clic sur l'item *Zoom Avant* du menu Pop-up	Utilisateur ; Non ; s3,2-I
eviu2,2-I	PopupMenuItem_ZoomAR_Click	Clic sur l'item *Zoom Arrière* du menu Pop-up	Utilisateur ; Non ; s5,2-I
eviu3,2-I	PopupMenuItem_Editer_Click	Clic sur l'item *Editer* du menu Pop-up	Utilisateur ; Non ; s6,2-I
eviu4,2-I	Ligne_Affichée	La vue d'une ligne est affichée	s1,2-I ; Non ; Non
eviu5,2-I	Ligne_Zoom_Agrandie	La vue des lignes est agrandie	s2,2-I ; Non ; Non
eviu6,2-I	Ligne_Zoom_Reduite	La vue des lignes est réduite	s4,2-I ; Non ; Non

Tableau A2.5 : BSA – Description des services de l'agent *Etat_Ligne*

ID du service (*)	Nom du service (*)	Description du service	Déclencheur ; IDs des EVIUs déclenchés ; IDs des services déclenchés
s1,2-I	Afficher une ligne	Exécution des traitements afin de montrer une ligne	s1,1-I ; eviu4,2-I ; Non
s2,2-I	Agrandir ma ligne	Agrandir l'écran de la vue de chaque ligne	s3,2-I ; eviu5,2-I ; Non
s3,2-I	Agrandir toutes les lignes	Agrandir l'écran de la vue de toutes les lignes	eviu1,2-I ; Non ; s2,2-I
s4,2-I	Réduire ma ligne	Réduire l'écran de la vue de chaque ligne	s5,2-I ; eviu6,2-I ; Non
s5,2-I	Réduire toutes les lignes	Réduire l'écran de la vue de toutes les lignes	eviu2,2-I ; Non ; s4,2-I
s6,2-I	Envoi des messages aux stations ou véhicules	Exécution des traitements afin d'envoyer un message aux stations et /ou aux stations	eviu3,2-I ; Non ; s1,5-I

2.3. Agent *interface Station* (3-I)

Les tableaux A2.6 et A2.7 montrent la description des EVIUs et des services de cet agent.

Tableau A2.6 : BSA – Description des EVIUs de l'agent *Station*

ID de l'eviu (*)	Nom de l'eviu (*)	Description de l'eviu	Déclencheur ; IDs des EVIUs déclenchés ; IDs des services déclenchés
eviu1,3-I	Image_Station_Click	Clic sur l'image d'une station sur l'écran	Utilisateur ; Non ; s1,3-I
eviu2,3-I	Fenêtre_Propriétés_Affichée	La fenêtre des propriétés d'une station est affichée	s1,3-I ; Non ; Non
eviu3,3-I	Fenêtre_Propriétés_Cachée	La fenêtre des propriétés d'une station est cachée	eviu9,3-I ; Non ; Non
eviu4,3-I	bouton_MessagePrédéfini_Click	Clic sur le bouton *Message Prédéfini* de la fenêtre des propriétés d'une station	Utilisateur ; Non ; s3,3-I
eviu5,3-I	Onglet_Passage1_Click	Clic sur l'onglet *passage 1* de la fenêtre des propriétés d'une station	Utilisateur ; Non ; Non
eviu6,3-I	Onglet_Passage2_Click	Clic sur l'onglet *passage 2* de la fenêtre des propriétés d'une station	Utilisateur ; Non ; Non
eviu7,3-I	BoiteTexte_Modifiée	utilisateur saisit le message dans la boîte de texte de la fenêtre des propriétés d'une station	Utilisateur ; Non ; Non
eviu8,3-I	bouton_OK_Click	Clic sur le bouton *OK* de la fenêtre des propriétés d'une station	Utilisateur ; Non ; s2,3-I
eviu9,3-I	bouton_Annuler_Click	Clic sur le bouton *Annuler* de la fenêtre des propriétés d'une station	Utilisateur ; eviu3,3-I ; Non

Tableau A2.7 : BSA – Description des services de l'agent *Station*

ID du service (*)	Nom du service (*)	Description du service	Déclencheur ; IDs des EVIUs déclenchés ; IDs des services déclenchés
s1,3-I	Afficher la fenêtre des propriétés d'une station	Exécuter des traitements afin d'afficher la fenêtre des propriétés d'une station	eviu1,3-I ; eviu2,3-I ; Non
s2,3-I	Envoyer un message à la station	Exécuter des traitements afin d'envoyer un message à la station	eviu8,3-I ; Non ; s7,5-I
s3,3-I	Afficher la fenêtre des messages	Afficher la fenêtre de l'agent *interface Message*	eviu4,3-I ; Non ; s2,5-I

2.4. Agent *interface Véhicule* (4-I)

Les tableaux A2.8 et A2.9 montrent la description des EVIUs et des services de cet agent.

Tableau A2.8 : BSA – Description des EVIUs de l'agent *Véhicule*

ID de l'eviu (*)	Nom de l'eviu (*)	Description de l'eviu	Déclencheur ; IDs des EVIUs déclenchés ; IDs des services déclenchés
eviu1,4-I	Image_Vehicule_Click	Clic sur l'image d'une station sur l'écran	Utilisateur ; Non ; s1,4-I
eviu2,4-I	PopupMenuItem_Retirer_Vehicule_Click	Clic sur l'item *Retirer* du menu Pop-up pour retirer un véhicule	Utilisateur ; eviu3,4-I ; Non
eviu3,4-I	BoiteConfirmation_Retire_Vehicule_Affiché	Affichage d'une boîte pour demander à l'utilisateur de vérifier la suppression d'un véhicule (*Oui* ou *Non* ?)	eviu2,4-I ; Non ; Non

eviu4,4-I	BoiteConfirmation_Retire_Vehicule_bouton_Oui_Click	Clic sur un bouton *Oui* de la boîte de vérification pour retirer un véhicule	Utilisateur ; Non ; s2,4-I
eviu5,4-I	BoiteConfirmation_Retire_Vehicule_bouton_Non_Click	Clic sur un bouton *Non* de la boîte de vérification pour annuler la suppression d'un véhicule	Utilisateur ; eviu6,4-I ; Non
eviu6,4-I	BoiteConfirmation_Retire_Vehicule_Ferme	Fermeture de la boîte de vérification de suppression d'un véhicule	eviu5,4-I ; Non ; Non
eviu7,4-I	PopupMenuItem_Propriétés_Click	Clic sur l'item *Propriété* du menu Pop-up pour afficher la fenêtre des propriétés d'un véhicule	Utilisateur ; Non ; s1,4-I
eviu8,4-I	Fenetre_Propriétés_Véhicule_Affichée	La fenêtre des propriétés d'un véhicule est affichée	s1,4-I ; Non ; Non
eviu9,4-I	Fenetre_Propriétés_Véhicule_Cachée	La fenêtre des propriétés d'un véhicule est cachée	eviu16,4-I ; Non ; Non
eviu10,4-I	bouton_MessagePrédéfini_Click	Clic sur le bouton *Message Prédéfini* de la fenêtre des propriétés d'un véhicule	Utilisateur ; Non ; s5,4-I ;
eviu11,4-I	Onglet_Message_Voyageur_click	Clic sur l'onglet *Voyageur* de la fenêtre des propriétés d'un véhicule pour saisir le message à envoyer aux voyageurs de ce véhicule	Utilisateur ; Non ; Non
eviu12,4-I	Onglet_Message_Conducteur_click	Clic sur l'onglet *Conducteur* de la fenêtre des propriétés d'un véhicule pour saisir le message à envoyer aux conducteurs de ce véhicule	Utilisateur ; Non ; Non
eviu13,4-I	BoiteTexte_Message_Voyageur_Modifiée	L'utilisateur saisit le message dans la boîte de texte de la fenêtre des propriétés d'un véhicule pour l'envoyer aux voyageurs de ce véhicule	Utilisateur ; Non ; Non
eviu14,4-I	BoiteTexte_Message_Conducteur_Modifiée	L'utilisateur saisit le message dans la boîte de texte de la fenêtre des propriétés d'un véhicule pour l'envoyer aux conducteur de ce véhicule	Utilisateur ; Non ; Non
eviu15,4-I	bouton_OK_Click	Clic sur le bouton *OK* de la fenêtre des propriétés d'un véhicule	Utilisateur ; Non ; s6,4-I et s7,4-I
eviu16,4-I	bouton_Annuler_Click	Clic sur le bouton *Annuler* de la fenêtre des propriétés d'un véhicule	Utilisateur ; eviu9,4-I ; Non
eviu17,4-I	Onglet_Arret1_Click	Clic sur l'onglet *Arret1* de la fenêtre des propriétés d'un véhicule	Utilisateur ; Non ; Non
eviu18,4-I	Onglet_Arret2_Click	Clic sur l'onglet *Arret2* de la fenêtre des propriétés d'un véhicule	Utilisateur ; Non ; Non
eviu19,4-I	Onglet_Arret3_Click	Clic sur l'onglet *Arret3* de la fenêtre des propriétés d'un véhicule	Utilisateur ; Non ; Non
eviu20,4-I	BoiteAvertissement_AvanceRetard_Affichée	Affichage d'une boîte pour avertir de l'avance ou du retard d'un véhicule	s3,4-I ; Non ; Non
eviu21,4-I	BoiteAvertissement_AvanceRetard_Fermer_Click	L'utilisateur fait un clic pour fermer la boîte d'avertissement du retard ou de l'avance d'un véhicule	Utilisateur ; eviu22,4-I ; Non
eviu22,4-I	BoiteAvertissement_AvanceRetard_Fermée	Fermeture de la boîte d'avertissement du retard ou de l'avance d'un véhicule	eviu21,4-I ; Non ; Non
eviu23,4-I	BoiteAvertissement_Panne_Affichée	Affichage d'une boîte pour avertir de la panne d'un véhicule	s4,4-I ; Non ; Non
eviu24,4-I	BoiteAvertissement_Panne_Fermer_Click	L'utilisateur fait un clic pour fermer la boîte d'avertissement de la panne d'un véhicule	Utilisateur ; eviu25,4-I ; Non
eviu25,4-I	BoiteAvertissement_Panne_Fermée	Fermeture de la boîte d'avertissement de la panne d'un véhicule	eviu24,4-I ; Non ; Non

Tableau A2.9 : BSA – Description des services de l'agent *Véhicule*

ID du service (*)	Nom du service (*)	Description du service	Déclencheur ; IDs des EVIUs déclenchés ; IDs des services déclenchés
s1,4-I	Afficher fenêtre des propriétés d'un véhicule	Exécution des traitements afin d'afficher la fenêtre des propriétés d'un véhicule	eviu1,4-I ou eviu7,4-I ; eviu8,4-I ; Non
s2,4-I	Mise-moi hors service	Exécution des traitements pour retirer un véhicule	eviu4,4-I ; Non ; s4,1-I
s3,4-I	Recevoir Avance/Retard d'un véhicule	Recevoir l'information concernant l'avance ou le retard des véhicules	système ; eviu20,4-I ; Non
s4,4-I	Recevoir Panne d'un véhicule	Recevoir l'information concernant la panne des véhicules	système ; eviu23,4-I ; Non
s5,4-I	Afficher la fenêtre des messages	Afficher la fenêtre de l'agent *interface Message*	eviu10,4-I ; Non ; s3,5-I
s6,4-I	Envoyer un message aux voyageurs	Exécution des traitements afin d'envoyer un message aux voyageurs du véhicule	eviu15,4-I ; Non ; s6,5-I
s7,4-I	Envoyer un message au conducteur	Exécution des traitements afin d'envoyer un message au conducteur du véhicule	eviu15,4-I ; Non ; s8,5-I

2.5. Agent *interface Message* (5-I)

Les tableaux 2.10 et 2.11 montrent la description des EVIUs et des services de cet agent.

Tableau A2.10 : BSA – Description des EVIUs de l'agent *Message*

ID de l'eviu (*)	Nom de l'eviu (*)	Description de l'eviu	Déclencheur ; IDs des EVIUs déclenchés ; IDs des services déclenchés
eviu1,5-I	Fenêtre_Messages_Affichée	La fenêtre de l'agent *Message* est affichée	s1,5-I ou s2,5-I ou s3,5-I ; Non ; Non
eviu2,5-I	Fenêtre_Messages_Cachée	La fenêtre de l'agent *Message* est cachée	eviu14,5-I ; Non ; Non
eviu3,5-I	Onglet_Editer_Message_click	Clic sur l'onglet *Editer* de la fenêtre pour saisir le message à envoyer	Utilisateur ; Non ; Non
eviu4,5-I	Onglet_Selectionner_Message _click	Clic sur l'onglet *Sélectionner* de la fenêtre pour sélectionner un message à envoyer parmi les messages disponibles	Utilisateur ; Non ; Non
eviu5,5-I	BoiteTexte_Message_Modifiée	L'utilisateur saisit le message dans la boîte de texte de la fenêtre	Utilisateur ; Non ; Non
eviu6,5-I	BoiteCombo_Un_Message_ Sélectionné	Parmi les messages disponibles, un message est sélectionné	Utilisateur ; Non ; Non
eviu7,5-I	BoiteCombo_Lignes_ Selectionnées	Parmi les lignes disponibles, quelques lignes sont sélectionnées	Utilisateur ; Non ; Non
eviu8,5-I	BoiteCheck_ToutesLignes_click	Clic sur le *checkbox* pour sélectionner toutes les lignes disponibles	Utilisateur ; Non ; Non
eviu9,5-I	BoiteListe_Stations_ selectionnées	Parmi les lignes disponibles, quelques stations sont sélectionnées	Utilisateur ; Non ; Non

eviu10,5-I	BoiteCheck_ToutesStations_click	Clic sur le *checkbox* pour sélectionner toutes les stations disponibles	Utilisateur ; Non ; Non
eviu11,5-I	BoiteListe_Véhicules_selectionnés	Parmi les véhicules disponibles, quelques véhicules sont sélectionnés	Utilisateur ; Non ; Non
eviu12,5-I	BoiteCheck_Toutes_Vehicules_ click	Clic sur le *checkbox* pour sélectionner tous les véhicules disponibles	Utilisateur ; Non ; Non
eviu13,5-I	bouton_Envoyer_click	Clic sur le bouton *OK*	Utilisateur ; Non ; s4,5-I et/ou s5,5-I
eviu14,5-I	bouton_Annuler_click	Clic sur le bouton *OK*	Utilisateur ; eviu2,5-I ; Non
eviu15,5-I	BoiteResponse_FeedbackAuEnvoi Messages_Affichee	Boîte de réponse du système à l'envoi des messages est affichée	s6,5-I ou s7,5-I ou s8,5-I ; Non ; Non

Tableau A2.11 : BSA – Description des services de l'agent *Message*

ID du service (*)	**Nom du service (*)**	**Description du service**	**Déclencheur ; IDs des EVIUs déclenchés ; IDs des services déclenchés**
s1,5-I	Afficher edit message	Affichage de la fenêtre de l'agent *Message*.	s6,2-I ; eviu1,5-I; Non
s2,5-I	Afficher edit station	Affichage de la fenêtre de l'agent *Message* avec une station sélectionnée	s3,3-I ; eviu1,5-I; Non
s3,5-I	Afficher edit vehicule	Affichage de la fenêtre de l'agent *Message* avec un véhicule sélectionné	s5,4-I ; eviu1,5-I; Non
s4,5-I	Envoyer un messages aux stations	Envoyer un message aux stations sélectionnées	eviu13,5-I; Non; s7,5-I
s5,5-I	Envoyer un messages aux voyageurs des vehicules	Envoyer un message aux voyageurs des véhicules sélectionnés	eviu13,5-I; Non; s6,5-I
s6,5-I	Recevoir feedback a l'envoi des messages aux voyageurs des vehicules	Recevoir la réponse du système à l'envoi d'un message aux voyageurs des véhicules sélectionnés	s6,4-I ou s5,5-I; eviu15,5-I ; Non
s7,5-I	Recevoir feedback a l'envoi des messages aux Stations	Recevoir la réponse du système à l'envoi d'un message aux stations sélectionnées	s2,3-I ou s4,5-I; eviu15,5-I ; Non
s8,5-I	Recevoir feedback a l'envoi des messages au Conducteur	Recevoir la réponse du système à l'envoi d'un message aux conducteurs des véhicules sélectionnés	s7,4-I ou s4,5-I; eviu15,5-I ; Non

2.6. Agent *interface Vue_Globale* (6-I)

Pour l'instant, cet agent fournit seulement au régulateur une vision globale du trafic dans le réseau. Les utilisateurs ne peuvent pas interagir avec cet agent *interface*. Donc, cet agent n'a aucun service ou EVIU.

Résumé

L'évaluation des systèmes interactifs est un domaine de recherche très actif depuis plus d'une trentaine d'années. Suite à l'apparition de systèmes interactifs à architecture dite à base d'agents, de nouvelles problématiques relatives à l'évaluation de tels systèmes apparaissent. Notre contribution réside dans la proposition d'un environnement générique et configurable baptisé EISEval (Environment for Interactive System Evaluation) pour l'aide à leur évaluation. En effet, il prend en compte leurs spécificités et étend les possibilités des mouchards électroniques traditionnels pour évaluer différents aspects d'un système : l'interface utilisateur, quelques propriétés non fonctionnelles (ex. : temps de réponse) et certaines propriétés de l'utilisateur. Cet environnement a été testé et validé dans le cadre du projet SART (Système d'Aide à la Régulation de Trafic), dans un cadre d'évaluation d'un système interactif dans le domaine des transports appelé SAI (Système d'Aide à l'Information des voyageurs).

Les perspectives de recherche concernent l'évaluation et l'amélioration du système SAI, de même que l'amélioration de l'environnement d'évaluation proposé, afin d'augmenter son automatisation et l'intégrer dans un environnement d'évaluation intégré. Cet environnement permettrait de combiner des résultats d'évaluation provenant de différentes méthodes (y compris de l'environnement EISEval) pour l'aide à l'évaluation des systèmes interactifs.

Mots clés : Évaluation, Interaction Homme-Machine, interface utilisateur, systèmes interactifs, architecture à base d'agents, architecture logicielle, mouchard électronique.

Abstract

The evaluation of interactive systems has been a very active subject of research since more than around thirty years. The appearance of agent-based interactive systems implies new problems relative to the evaluation of such systems. Our contribution is to propose a generic and configurable environment baptized EISEval (Environment for Interactive System Evaluation) to assist the evaluation of agent-based interactive systems. Indeed, it takes into account specificities of such systems and extend the possibilities of the traditional electronic informers to evaluate different aspects of a system: the user interface, some non-functional properties (for instance: response time) and certain user's properties. This environment has been validated by applying it in the SART project, to evaluate an interactive system in the field of transport called IAS (Information Assistance System).

The research perspectives concern the evaluation and the improvement of the SAI system, and the improvement of the proposed evaluation environment to increase its automation and integrate it into an integrated environment. This environment would allow to combine evaluation results from various methods (including the proposed EISEval environment) to assist the evaluation of interactive systems.

Keywords: Evaluation, human-computer interaction, user interface, interactive systems, agent-based architecture, software architecture, electronic informer.