THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par *l'Université Toulouse 3 – Paul Sabatier*
Discipline ou spécialité : *Informatique*

Présentée et soutenue par *Célia Martinie De Almeida*
Le *5 décembre 2011*

Titre : *Une approche à base de modèles synergiques pour la prise en compte simultanée de l'utilisabilité, la fiabilité et l'opérabilité des systèmes interactifs critiques*

JURY
Joëlle Coutaz (Professeure, Université Joseph Fourier, Grenoble, Rapporteure)
Jean Vanderdonckt (Professeur, Université Catholique de Louvain, Rapporteur)
Fabio Paterno (Directeur de recherche, CNR-ISTI, Italie, Examinateur)
Marco Winckler (Maître de Conférences, Université Toulouse 3, Co-encadrant)
Jean-Pierre Jessel (Professeur, Université Toulouse 3, Président du jury)
Philippe Palanque (Professeur, Université Toulouse 3, Directeur de thèse)
Erwann Poupart (Ingénieur CNES, Membre invité)

Ecole doctorale : *Mathématiques Informatique et Télécommunications de Toulouse (MITT)*
Unité de recherche : *IRIT – UMR 5505*
Directeur(s) de Thèse : *Philippe Palanque*
Rapporteurs : *Joëlle Coutaz et Jean Vanderdonckt*

Remerciements

Je remercie tout particulièrement Philippe Palanque, grâce à qui j'ai eu l'opportunité de préparer cette thèse, et la chance d'appréhender le monde de la recherche académique à un niveau international. Son accueil, son ouverture d'esprit, son énergie et la juste rigueur de ses raisonnements sont pour moi une source d'inspiration et une référence dans les moments de questionnement.

La préparation de cette thèse a aussi été rendue possible grâce au financement du Centre National d'Etudes Spatiales (CNES) de Toulouse, notamment par le biais du projet TORTUGA. Ainsi, je remercie Erwann Poupart, co-responsable de ce projet, pour la confiance qu'il nous a accordée, pour son aide, ainsi que pour ses avis constructifs. Je remercie également Eliane Cubero-Castan, qui nous a guidés dans l'observation et l'analyse des opérations menées dans les centres de commande et contrôle du CNES.

D'autre part, cette thèse est l'aboutissement d'un projet professionnel qui a été soutenu par les responsables des ressources humaines de Motorola France, ainsi je remercie particulièrement Catherine Clairefond et Karine Bugarel.

Les travaux présentés dans cette thèse sont issus de nombreuses heures de travail personnel mais leur consistance tient aux nombreux échanges et aux nombreuses discussions avec Eric Barboni, David Navarre et Marco Winckler. Travailler au sein de cette équipe vivace sur des projets de recherche stimulants fût instructif, enrichissant et très plaisant. Je les remercie pour leur soutien, leurs conseils, leurs encouragements, leur excellente humeur (et leur excellent humour) à toute épreuve.

Je remercie vivement Joëlle Coutaz et Jean Vanderdonckt de m'avoir fait l'honneur d'être les rapporteurs de mon mémoire, en consacrant ainsi de leur temps pour étudier cette thèse et me prodiguer leurs conseils et avis. Je remercie Fabio Paterno et Jean-Pierre Jessel d'avoir accepté de faire partie du jury, je suis aussi très honorée de pouvoir recueillir leurs avis.

Je remercie également les membres de l'équipe ICS, actuels et anciens, pour leurs encouragements, et en particulier Martina Ragosta, Christelle Farenc, Regina Bernhaupt, Mickael Pirker, Adrienne Tankeu-Choitat, Camille Fayollas, Arnaud Hamon, Stéphane Conversy, Sandra Steere, Jean-François Ladry, Thomas Mirlacher et Cédric Bach. Je remercie aussi Yannick Jestin pour ses conseils et encouragements.

Je suis très reconnaissante à mon entourage proche et familial, pour toute l'aide et le soutien apporté. Je remercie Delphine, Marie, Virginie, Eva, Sandrine, Karine et Erika pour leur amitié indéfectible et leur joie de vivre. Je remercie mes parents, Daniel et Simone, et ma sœur, Solène, pour leur soutien continu, me permettant d'assouplir mon emploi du temps de mère de famille et ainsi d'assister à plusieurs conférences, plus ou moins éloignées. Je remercie particulièrement Solène, pour sa disponibilité et pour être toujours à mes côtés dans les moments de doute. Je remercie mon mari, Miguel, pour la paix et la confiance qu'il m'insuffle, sans lesquelles je n'aurais sûrement pas eu l'audace de suivre cette voie. Enfin, je remercie Adam et Cléophée, pour me transmettre leur énergie et pour m'émerveiller à chaque instant.

Table des matières

Liste des figures

Liste des tables

Introduction

Le paradigme de la conception centrée utilisateur (Norman & Drapper, 1986) est aujourd'hui au cœur des processus de développement des systèmes informatiques interactifs. Toutes les techniques, méthodes, et processus de développement utilisés visent à connaître et comprendre les utilisateurs (analyser leurs besoins, évaluer leurs manières d'utiliser les systèmes) dans le but de concevoir et développer des systèmes utilisables, c'est-à-dire, en adéquation avec leurs comportements, leurs compétences et leurs besoins. Ces techniques, méthodes, et processus se perfectionnent au fil des ans et se répandent progressivement dans les industries dites « grand public ». La philosophie sous-jacente à la conception centrée utilisateur est de concevoir des systèmes « affordants » pour être utilisés efficacement, avec satisfaction par les utilisateurs, et ce dès leur premier contact avec le système (et donc généralement sans avoir suivi de formation à l'utilisation du système). La mise en œuvre d'une telle conception a pour but de de produire des systèmes interactifs satisfaisant la propriété d'**utilisabilité**. Cette propriété a été normalisée ISO 9241 (International Standard Organization, 1996) et se décompose en 3 facteurs principaux « efficiency », « effectiveness » and « satisfaction ». Ces trois facteurs sont abordés de façon intégrée dans la conception centrée utilisateur en plaçant les utilisateurs potentiels au cœur même du processus de développement.

Actuellement, le domaine de l'interaction homme-machine continue à contribuer avec des méthodes, des techniques et des outils pour mettre en œuvre de façon toujours plus efficace ce principe de conception centrée utilisateur (Blanch & Ortega, 2011) (Wherry, 2003) (Apitz, Guimbretière, & Zhai, 2008) (Höök, et al., 2011). Par exemple le concept de « user experience » qui peut se traduire en Français par le terme « ressenti utilisateur » a pour vocation à étendre la notion d'utilisabilité à des concepts plus impalpables comme « fun », « pleasure » « excitement » (Hassenzahl, 2003).

Le facteur « efficiency » est de loin celui qui a reçu et reçoit toujours le plus d'attention. Ceci est sans doute dû au fait que la performance des utilisateurs est une valeur mesurable et monnayable comme décrit en détail dans le processus de retour sur investissement de l'utilisabilité (Bias & Mayhew, 2005). L'aspect « effectiveness »est moins considéré car les systèmes interactifs innovants grand public se limitent souvent à un nombre de tâches utilisateur réduit et relativement simple (saisir un SMS, retailler une photographie, changer de chaine de télévision…).

Cette volonté d'accroitre la performance des utilisateurs peut être vue de façon générique sous l'angle d'un accroissement de la bande passante entre l'utilisateur et le système interactif. Dans cette optique le domaine de l'interaction homme-machine propose inlassablement des techniques d'interaction toujours plus sophistiquées et souvent en corrélation étroite avec les innovations industrielles en matière de système d'entrée innovants, comme les travaux de (Marquardt, Jota, Greenberg, & Jorge, 2011), (Baudisch, et al., 2003) et (Grossman & Balakrishnan, 2005). Ces exemples d'innovation en matière de techniques d'interaction sont représentés sur les Figure 1 et Figure 2.

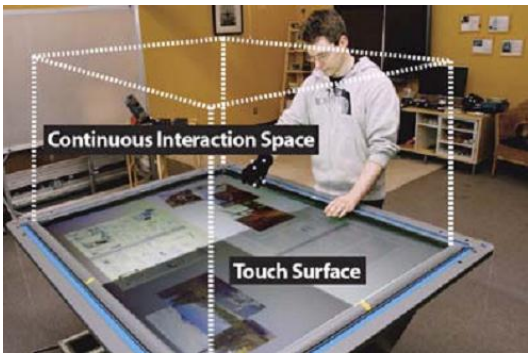


Figure 1. Illustration de la technique d'interaction « Continuous Interaction Space » (Marquardt, Jota, Greenberg, & Jorge, 2011)

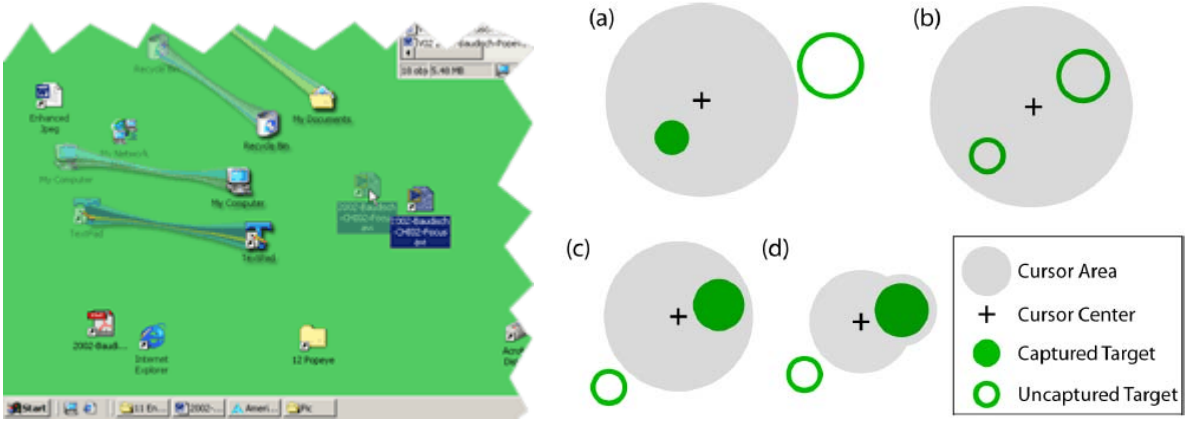


Figure 2. Illustration des techniques d'interaction « Drag and Pop » (Baudisch, et al., 2003) et « Bubble cursor » (Grossman & Balakrishnan, 2005)

Dans le but de démontrer la faisabilité de ces techniques d'interaction, les chercheurs du domaine de l'interaction homme machine développent en général leurs propres outils (bibliothèques, langages, ou autres) qu'ils exploitent pour concevoir, développer et évaluer ces techniques d'interaction. Ces techniques sont en général décrites dans des articles de recherche et souvent enregistrées sous forme de document vidéo démontrant à la fois les aspects visuels et comportementaux (voir par exemple http://www.patrickbaudisch.com/projects/dragandpop/index.html). Une fois ces techniques publiées, il reste un long chemin à parcourir pour permettre à ces techniques d'interaction d'atteindre le niveau de maturité nécessaire à leur exploitation à large échelle. Ce long chemin est semé d'embuches que nous allons détailler ci-dessous. Il est important de noter que l'accent est mis ici sur les techniques d'interaction mais que bien évidemment tous les points soulevés s'appliquent identiquement à l'intégralité du système interactif.

- **Le passage à l'échelle**: Il s'agit d'une étape durant laquelle il faut maitriser en détail l'ensemble de la technique d'interaction et son applicabilité. Un exemple de technique d'interaction où le passage à l'échelle s'avère délicat est le pointage sémantique (Blanch, Guiard, & Beaudoin-Lafon, 2004) qui perd en efficacité s'il y a plusieurs objets interactifs entre la position initiale du curseur et la cible. Il faut ensuite faire passer la technique de l'état prototype/démonstrateur à celui de produit industriel.
- **La fiabilité**: Il faut ici trouver des moyens de garantir le bon fonctionnement du système interactif et ce, indépendamment des variations d'utilisation. Des approches telles que la vérification, la validation le test exhaustif sont les seules façons de prendre en compte cette propriété.

- **L'applicabilité** : Il faut ici proposer des méthodes techniques et outils qui puissent s'intégrer dans les processus de développement et avec les outils de développement actuellement utilisés par les industriels.
- **L'exploitabilité** : Il faut que les systèmes produits (surtout dans le cas où ils sont de grande taille et complexes à gérer) soit exploitables lorsqu'ils sont déployés. Habituellement les exemples donnés correspondent à de petites applications, souvent appelées « toy application » (comme l'exemple de la technique « Drag and Pop » sur la Figure 2), qui par définition ne nécessitent pas de formation du côté des utilisateurs. De manière générale, la formation des utilisateurs aux systèmes interactifs est lourdement sous-estimée, ce qui entraine la production (quand ils existent) de programmes de formation incomplets et superficiels. Toutefois, dans le domaine des jeux vidéo il est actuellement quasiment systématique de fournir aux joueurs un ensemble de scénarios de « prise en main » permettant non seulement de découvrir et maitriser les techniques d'interaction mais aussi l'ensemble des fonctionnalités du jeu.
- **La tolérance aux erreurs** : Du fait de cette simplicité excessive, il est aussi supposé que les utilisateurs ne font pas d'erreurs (ni au niveau manipulation (« slips ») ni au niveau planification (« mistakes ») selon la classification de James Reason (Reason, 1990)). La prise en compte de ces éléments est très laborieuse car elle requiert une étude systématique des déviations possible (par rapport à l'utilisation normative du système) et la mise en place de barrière pour prévenir leur occurrence.

Cet ensemble de problèmes est important à traiter dans le cas où les applications sont complexes et offrent un grand nombre de fonctionnalités, et n'est peut-être pas crucial dans le cadre d'applications très simple ou d'utilisation très standard (e.g. un agenda).

Dans le domaine des systèmes interactifs critiques, une erreur de conception engendrant une défaillance du système peut se chiffrer en nombre de vies humaines ou en coûts matériels exorbitants. Leur conception ne peut donc pas être seulement centrée sur l'utilisateur (Palanque, et al., 2007), mais requiert la prise en compte systématique et exhaustive des problèmes décrits ci-dessus. Pour ce type de systèmes, la prise en compte de l'ensemble des problèmes énoncés ci-dessus est nécessaire car ces systèmes doivent la plupart du temps passer au crible d'une activité de certification observant de façon systématique et détaillée le processus de développement déroulé, et les différents éléments produits. Pendant de très nombreuses années, les systèmes critiques ont intégré des techniques d'interactions archaïques et principalement basées sur la manipulation de composants électroniques pouvant être certifiés. Actuellement, la complexité des systèmes et des informations qu'ils manipulent requiert l'intégration de techniques d'interaction évoluées. Par exemple, la Figure 3, présente une photographie d'un cockpit de l'avion commercial Airbus A380, qui démontre cette récente intégration dans la mesure où la manipulation de certains systèmes bord s'effectue au moyen d'un track ball (appelé Keyboard Cursor Control Unit), permettant la mise en œuvre d'interactions de type WIMP (via les écrans interactifs appelés Display Units).

Figure 3. Cockpit de l'Airbus A380

Dans les paragraphes suivants nous allons décliner les problèmes présentés ci-dessus dans le domaine des systèmes interactifs critiques.

Les systèmes interactifs critiques

La description complète et non ambiguë est actuellement considérée comme une exigence dans le domaine des systèmes critiques étant donné que les interfaces utilisateurs sont désormais le moyen de contrôler ces systèmes. Dans le cas où les interfaces utilisateur de ces systèmes représentent une quantité importante de lignes de codes (en termes de logiciel), l'outil de description doit permettre de gérer ce volume. Il ne s'agit pas ici seulement de support à la configuration logicielle, mais de support à une activité de description d'un plus haut niveau d'abstraction. La modélisation fournit un support à ce type de description et est d'ailleurs largement utilisée en génie logiciel (avec UML[1] par exemple). Cependant, une approche comme UML n'est pas adaptée à la prise en compte des caractéristiques idiosyncratiques des systèmes interactifs. Par exemple, les travaux de (Nunes & Cunha, 2000) ont démontré l'incapacité des cas d'utilisation à capturer les informations nécessaires à la représentation des tâches et des scénarios. De même, les travaux de (Bastide & Palanque, 2003) ont démontré l'inadéquation des diagrammes à états UML, et des diagrammes d'états (Harel, 1987) en général, à représenter le comportement des applications interactives. Au-delà de ce problème, certains spécialistes dans le domaine de l'interaction homme-machine ont mis en avant des approches d'ingénierie logicielle à base de boîtes à outils bas niveau, au détriment d'approches à base de modèles selon eux inadéquate pour l'ingénierie des systèmes interactifs. Ainsi, dans la contribution de (Sukaviriya, et al., 1994), Dan Olsen écrit: "There have been two major problems with this approach [model based ones]. The first is that we always seem to be modelling the interface styles of the previous generation of user interfaces. The second is that the models developed in some way limit the kind of user interface that is possible." Cependant, des contributions plus récentes démontrent l'applicabilité des approches à base de modèles pour la description des applications interactives intégrant des techniques d'interaction innovantes, telles que les techniques post-WIMP (Jacob, 1999) ou les techniques intégrant des animations (Esteban, Chatty, & Palanque, 1995).

[1] http://www.uml.org/

Pour améliorer l'utilisabilité et la facilité d'apprentissage de ces systèmes, des fonctionnalités génériques et spécifiques doivent être mises à la disposition des utilisateurs. Cependant, augmenter le nombre de fonctionnalités en ajoutant par exemple des mécanismes de « Undo/Redo », des capacités de « WYSIWYG », … accroît la probabilité d'occurrences de défaillances (sans prendre en compte le fait que de telles fonctionnalités sont particulièrement difficiles à mettre en œuvre). En particulier, le niveau de fiabilité des suites logicielles, commercialisées par les grandes entreprises de l'industrie du logiciel, démontre que la construction de systèmes interactifs fiables reste un challenge pour les équipes de développement. Alors que redémarrer un système ou y installer une mise à jour corrective suite à une défaillance est un comportement envisageable dans le cadre d'applications grand public, ceci n'est pas une option lorsque l'on s'intéresse à des systèmes interactifs critiques et que la vie de personnes est en jeu.

La raison d'investir dans la définition, l'utilisation et le déploiement de techniques de description formelles est en fait l'un des seuls moyen de permettre la description complète précise et non-ambiguë de l'ensemble des composants (présentation, dialogue et noyau fonctionnel) d'un système interactif tout en proposant des techniques pour raisonner sur ces descriptions. L'application des techniques de description formelles offre des avantages tout au long du processus de développement, depuis les phases amont (au niveau de l'analyse et de l'explicitation des besoins) jusque dans les phases aval en offrant une aide à la mise en place de tests (Memon & Xie, 2005).

Enfin, la certification est une phase du processus de développement entièrement spécifique aux systèmes critiques. Les développeurs de ces systèmes doivent de fait s'assurer (et être capable de démontrer) que le niveau de fiabilité (ou de défaillance) est acceptable eu égard au niveau de criticité de l'application en cours de développement. S'ils ne sont pas capables de fournir une telle démonstration les autorités de régulation (par exemple la Federal Aviation Administration ou FAA dans le domaine aéronautique) n'autoriseront pas l'exploitation du système. Ainsi, sans la définition de processus de développement s'intégrant explicitement dans des processus supportant la certification (comme le standard DO-178B (European Organisation for Civil Aviation Equipment, 1992)), les approches d'ingénierie des systèmes interactifs ne seront pas à même de proposer des solutions à l'intégration d'interfaces homme machine évoluées dans des systèmes interactifs critiques.

Hypothèses de la thèse

Dans cette thèse notre objectif est de démontrer qu'une approche fournissant un support à la conception et au développement de systèmes simultanément utilisables, fiables et opérables est possible. Les deux principales questions de recherche traitées par cette thèse sont :

- Quel support fournir en termes de processus, notations et outils pour la conception et le développement d'un système interactif critique utilisable, opérable et fiable ?
- Quel support fournir pour développer un programme de formation en adéquation avec le système qui permette de garantir un bon niveau de compétences et connaissances des utilisateurs de ce système?

L'hypothèse de cette thèse est centrée autour de la notion de modèles. En effet, le contenu de ce mémoire a pour objectif de démontrer qu'une approche **multi-modèles intégrée** à un **processus de développement supporté** par un **ensemble d'outils** permet d'atteindre ces objectifs.

Les modèles retenus concernent le comportement du système interactif et de ces techniques d'interaction ainsi que l'ensemble des tâches utilisateurs. La conception, l'édition et la vérification de leur compatibilité est encadrée par un processus de développement itératif et systématique. , des

techniques de modélisation, des outils logiciels et intègre la conception du système et du programme de formation lui correspondant, traditionnellement conçus de manière séparée.

Structure du mémoire

Ce mémoire s'organise en sept chapitres, chacun correspondant à une étape de l'argumentaire.

Le **Chapitre 1** présente les fondements des systèmes interactifs critiques et permet de mettre en valeur leurs nombreuses propriétés et caractéristiques associées à leur développement et mise en œuvre.

Le **Chapitre 1** discute des moyens de conception et développement des systèmes interactifs critiques afin de comprendre dans quelle mesure les moyens existants peuvent fournir ou non un support au développement d'un système simultanément utilisable, fiable et opérable. Les avantages et inconvénients des approches de développement et des notations et outils de modélisation sont identifiés et servent de base aux propositions exposées dans les **Chapitre 4**, **Chapitre 1** et **Chapitre 1**.

Le **Chapitre 1** présente les principes et caractéristiques des approches de développement de programmes de formation pour les systèmes interactifs critiques. Nous y montrons que ce type d'approches est centré sur l'analyse des tâches métier, et sur l'évaluation des performances du personnel formé lors de l'exécution de ces tâches, et nos conclusions servent aussi de base aux propositions exposées dans les **Chapitre 4**, **Chapitre 1** et **Chapitre 1**.

Les **Chapitre 4**, **Chapitre 1** et **Chapitre 1** présentent les contributions apportées dans le cadre de cette thèse.

Le **Chapitre 4** présente HAMSTERS, une notation outillée pour la modélisation des tâches utilisateurs de systèmes interactifs critiques et de systèmes interactifs pour lesquels les tâches utilisateur sont complexes. Nous y décrivons la notation de manière détaillée afin de mettre en valeur les mécanismes qu'elle fournit pour aider à concevoir des systèmes interactifs critiques. Un exemple illustratif est exposé pour décrire sa mise en œuvre.

Le **Chapitre 1** présente une approche basée sur les modèles et sur un processus pour la conception et le développement de systèmes interactifs critiques et de leurs programmes de formation associés. Ce chapitre détaille le processus proposé. Ce processus fournit un cadre conceptuel et une association d'étapes pour : le développement d'un système simultanément utilisable et fiable, le développement du programme de formation associé ainsi que la traçabilité des exigences et des choix de conception tout au long des différentes étapes.

Le **Chapitre 1** décrit la mise en œuvre de cette approche avec les notations et outils logiciels sélectionnés à l'issue du chapitre 2 pour la modélisation et le prototypage très haute-fidélité. Cette description de la mise en œuvre de l'approche permet de mettre en valeur l'utilisation de certains principes de la conception centrée utilisateur et l'exploitation synergique des modèles des tâches, des modèles formels du comportement du système et du modèle de développement du programme de formation. Des outils logiciels pour la formation basés sur cette approche sont proposés.

Le **Chapitre 1** présente la mise en œuvre de l'approche proposée sur une étude de cas industrielle. Nous y décrivons la conception et le développement d'une application de commande et contrôle d'une mission satellitaire. Cette mise en œuvre de toutes les étapes du processus permet de valider la faisabilité de notre approche.

Les dernières parties du mémoire sont consacrées à la conclusion ainsi qu'aux perspectives d'utilisation de cette approche dans de futurs travaux.

Dans le cadre de ce mémoire, nous laissons hors du périmètre de recherche la thématique coopérative et les notions de rôle utilisateur. Elle pourra être ajoutée a posteriori et n'est pas incompatible avec le processus proposé.

Chapitre 1- Fondements des systèmes interactifs critiques

Ce chapitre présente les caractéristiques et propriétés des systèmes interactifs et des systèmes critiques afin de :

- Caractériser les systèmes faisant l'objet des travaux de cette thèse.
- Mettre en valeur les problématiques liées à leur conception, développement et exploitation.

La première section décrit les caractéristiques principales des systèmes qui font l'objet des travaux effectués dans le cadre de cette thèse et particulièrement les propriétés d'utilisabilité, d'opérabilité et de fiabilité. Ces trois propriétés sont issues respectivement des domaines de l'IHM, du génie logiciel et de la sûreté de fonctionnement et permettent de décrire les besoins pour un système de fournir les moyens à un utilisateur de système interactif critique d'accomplir sa mission, avec un certain niveau de performances, de manière continue dans le temps et jusqu'à la fin de son exploitation, sans mettre en péril son intégrité, l'intégrité de ses pairs et celle du système.

La seconde section inventorie les propriétés de ces systèmes et les concepts liés à ces propriétés afin de mettre en évidence la complexité de la prise en compte de ces propriétés lors de la conception et développement de ces systèmes et la nécessité de méthodes adaptées.

La troisième section montre les différences entre la mise en exploitation des systèmes interactifs critiques et des systèmes interactifs dits grand public.

1 Définitions et caractéristiques des systèmes interactifs critiques

1.1 Systèmes interactifs et systèmes critiques

D'après les travaux de (Palanque, 1992), (Beaudouin-Lafon, 1997) et (Palanque, 2011), un **système interactif** est un système informatisé réactif prenant en compte, au cours de son exécution, les informations communiquées par le ou les utilisateurs du système, et produisant, au cours de son exécution, une représentation perceptible de son état interne. Les entrées fournies par le ou les utilisateurs dépendent des sorties produites par le système et ne sont pas complètement prévisibles (à cause de la nature non prédictible du comportement d'un être humain). Les sorties fournies par le système dépendent des entrées fournies par l'utilisateur et sont destinées à être interprétées correctement par l'utilisateur.

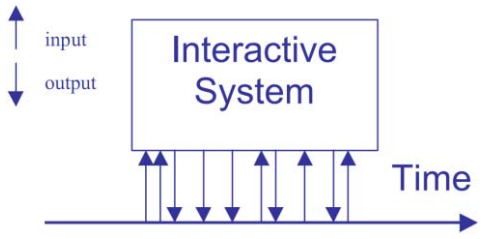


Figure 4. Représentation fonctionnelle générale d'un système interactif (Palanque, 2011)

La Figure 4 présente schématiquement le fonctionnement général d'un système interactif, et permet de montrer les flux d'informations entrantes et sortantes du système au cours du temps.

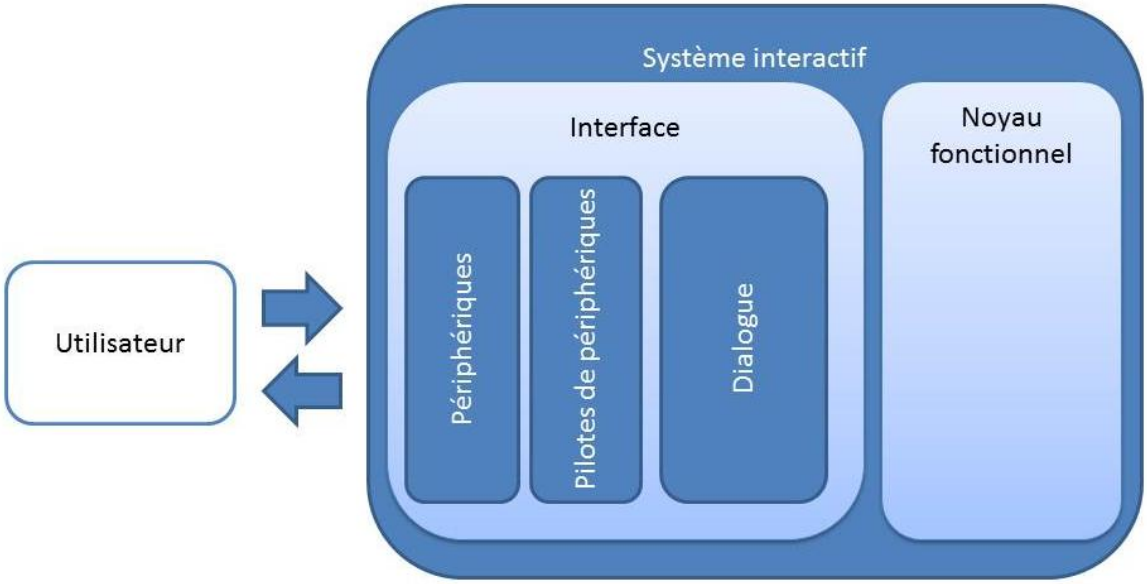


Figure 5. Architecture générale d'un système interactif

La Figure 5 présente l'architecture générale d'un système interactif et permet de montrer les composantes principales de ce type de système. L'utilisateur interagit avec le système via une interface composée de périphériques (matériel, électronique), de pilotes de périphériques (logiciel), et d'un contrôleur de dialogue (logiciel). Le contrôleur de dialogue gère l'ordonnancement des opérations d'entrée/sortie et leur cohérence. Ce concept est issu des modèles de Seeheim et Arch/Slinky (Bass, et al., 1991). Cette représentation de l'architecture d'un système interactif permet de mettre en valeur les différentes parties matérielles, électroniques et logicielles qui doivent être prises en compte lors de la conception et du développement de ce type de système.

Un **système critique** est généralement définit comme un système pour lequel une erreur de fonctionnement peut avoir une ou plusieurs conséquences dramatiques sur l'intégrité d'une ou plusieurs personnes ou sur l'économie d'une entreprise ou organisation (Sommerville, 2006). Les systèmes informatiques critiques en terme de sécurité, appelés « Safety-critical systems », peuvent être définis comme des systèmes qui ne mettront pas en péril des vies humaines ou notre environnement : « System that will not endanger the human life or the environment », page 2, (Storey, 1996). En termes d'ingénierie de sécurité et de sûreté, les concepts de risques, dangers, incidents et accidents sont liés à ces systèmes et pris en compte lors de la conception et de l'analyse de ces systèmes.

1.2 Systèmes interactifs critiques

La définition de **système interactif critique** a été proposée par (Palanque & Bastide, 1994) et correspond aux systèmes interactifs pour lesquels le coût d'une erreur potentielle d'utilisation ou de dysfonctionnement du système peut dépasser le coût de développement du système.

Tout au long de ce mémoire, nous utiliserons le terme système interactif critique pour décrire un ensemble cohérent de composants, intégrant des éléments matériels (mécanique et/ou électronique) et/ou des éléments logiciels, conçu et développé pour assister l'utilisateur dans ses missions en lui fournissant les moyens de :

- Comprendre son état et ses capacités en temps réel.
- Demander l'exécution d'une action.
- Assurer son intégrité, l'intégrité de l'utilisateur et de son environnement.

De plus, les travaux présentés dans le cadre de cette thèse visent à prendre en compte des systèmes complexes, dont le nombre de fonctionnalités et d'états interne est particulièrement important comme : les applications de commande et contrôle de missions satellitaires, les applications de cockpits d'avions civils et militaires, la gestion du trafic aérien. Les utilisateurs opérant ces systèmes sont appelés par différents noms selon le domaine d'application. Dans la suite de ce mémoire, nous emploierons le terme générique opérateur pour désigner un utilisateur de système interactif critique ou de système interactif complexe.

2 Propriétés des systèmes interactifs et des systèmes critiques

Cette section décrit de façon exhaustive les propriétés des systèmes interactifs et les propriétés des systèmes critiques afin de mettre en évidence :

- La complexité de la prise en compte de ces propriétés de la conception à l'évaluation d'un système interactif critique.
- La nécessité des approches, processus et techniques de développement adaptées à ces systèmes.

Pour des raisons d'ambiguïté à l'interprétation de langage entre les termes anglophones et francophones, certaines propriétés sont citées seulement en anglais.

2.1 Propriétés des systèmes interactifs

Certaines propriétés peuvent faire l'objet d'exigence et/ou besoins lors de la conception et du développement d'un système interactif. Ces propriétés font l'objet de plusieurs travaux, standards et recommandations.

Le standard ISO DIS 9241 (International Standard Organization, 1996) définit des exigences ergonomiques pour les systèmes interactifs possédant des périphériques d'affichage visuel. Ce standard apporte des lignes directives quant à l'évaluation de l'utilisabilité de ce type de système et propose des critères de mesures en termes d'efficacité, d'efficience et de satisfaction. Ce standard définit l'**utilisabilité** par « le degré selon lequel un produit peut être utilisé, par des utilisateurs identifiés, pour atteindre des buts définis avec efficacité, efficience et satisfaction, dans un contexte d'utilisation spécifié ». Le concept d'utilisabilité est aujourd'hui largement répandu et mis en œuvre dans toute la communauté de l'IHM pour concevoir et développer des systèmes interactifs dans tous les domaines d'application. Cependant, cette propriété, même si elle se décompose en trois caractéristiques mesurables, reste générique et ne permet pas de caractériser toutes les parties d'un système interactif.

Les travaux de (Graham & Cockton, 1996) proposent une classification des propriétés des systèmes interactifs selon deux catégories:

- Les **propriétés externes** sont liées à l'utilisabilité du système et permettent de caractériser la qualité de l'interaction entre le système et l'utilisateur.
- Les **propriétés internes** sont liées à l'infrastructure matérielle et humaine mise en place pour développer le système et permettent de caractériser la qualité de conception du système.

La Figure 6 a été conçue d'après les propriétés internes et externes proposées par (Graham & Cockton, 1996) afin de structurer en deux catégories des propriétés d'un système interactif.

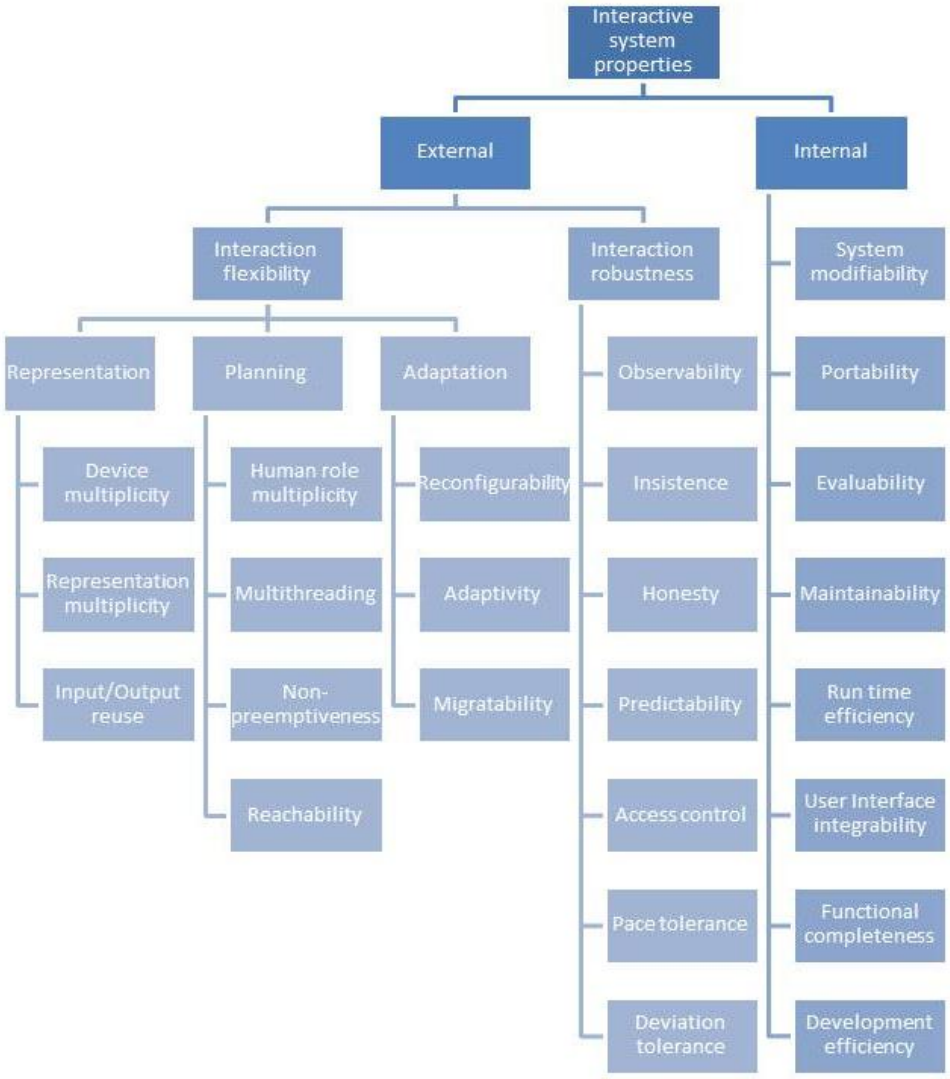


Figure 6. Propriétés externes et internes d'un système interactif d'après (Graham & Cockton, 1996)

Les propriétés externes (bloc « External » sur la Figure 6) se décomposent en deux grandes familles :

- La flexibilité de l'interaction, « Interaction flexibility » sur la Figure 6, permet de qualifier la manière dont le système interactif permet d'échanger des informations avec l'utilisateur et se décompose en plusieurs sous-propriétés :
  - La représentation, « Representation » sur la Figure 6, permet de caractériser : la capacité du système à fournir un ou plusieurs périphériques d'entrée/sortie (« Device multiplicity »), la capacité du système à fournir plusieurs alternatives de

représentation d'entrée et de sortie (« Representation multiplicity ») ainsi que la capacité du système à permettre l'utilisation d'une opération d'entrée/sortie pour une action future (« Input/Output re-use »), comme par exemple l'opération de copier-coller dans un éditeur de texte.

- La planification, « Planning » sur la Figure 6, permet de caractériser : la capacité du système à prendre en compte plusieurs types d'utilisateurs ou rôles, « Human role multiplicity », la capacité du système à autoriser l'utilisateur à décider des prochaines actions, « Non-preemptiveness », ainsi que la capacité du système à permettre à l'utilisateur d'atteindre n'importe quel état du système, « Reachability ».
- L'adaptation, « Adaptation » sur la Figure 6, permet de caractériser : la capacité de l'interface du système à être reconfigurée à l'initiative de l'utilisateur, « Reconfigurability », la capacité du système à initier une reconfiguration de l'interaction, « Adaptivity », la capacité du système à permettre un changement d'allocations de fonctions et tâches entre le système et l'utilisateur, « Migratability ».

- La robustesse de l'interaction, « Interaction robustness » sur la Figure 6, permet de qualifier la capacité du système interactif à accomplir ses tâches sans commettre d'erreurs irréversibles et peut se décomposer en sept sous-propriétés :
  - L'observabilité, « Observability » sur la Figure 6, permet de caractériser la capacité du système à fournir toutes les informations pertinentes nécessaires à l'utilisateur.
  - L'insistance, « Insistence » sur la Figure 6, permet de caractériser la capacité du système à rendre saillantes les informations nécessaires et à les faire percevoir par l'utilisateur.
  - L'honnêteté, « Honesty » sur la Figure 6, permet de caractériser la capacité du système à engendrer une interprétation correcte de l'information perçue par l'utilisateur.
  - Le caractère prévisible, « Predictability » sur la Figure 6, permet de décrire la capacité du système à permettre à l'utilisateur de prédire les futurs états possibles du système.
  - Le contrôle d'accès, « Access Control » sur la Figure 6, permet de caractériser la capacité du système à fournir les moyens de fixer et modifier les politiques de contrôle et d'accessibilité aux informations du système.
  - L'adaptation du rythme, « Pace tolerance » sur la Figure 6, permet de caractériser l'adéquation temporelle entre les actions de l'utilisateur et le traitement des informations d'entrée/sortie du système.
  - La tolérance aux déviations, « Deviation tolerance » sur la Figure 6, permet de caractériser la capacité du système à tolérer les erreurs et dérapages de l'utilisateur.

Ces propriétés externes sont nombreuses car liées au caractère imprédictible du comportement et des actions des utilisateurs du système.

Les propriétés internes du système (bloc « Internal » sur la Figure 6) se décomposent en sept sous-propriétés :

- La modifiabilité, « System modifiability » sur la Figure 6, permet de caractériser la capacité de modification et d'extension des fonctionnalités du système.
- La portabilité, « Portability » sur la Figure 6, permet de caractériser la capacité du système à être modifié (matériel, logiciel, environnement).
- La capacité à être évaluer, « Evaluability » sur la Figure 6.
- Les performances temporelles, « Run time efficiency » sur la Figure 6.
- La capacité d'intégration de l'interface utilisateur, « User interface integrability » sur la Figure 6.

- L'exhaustivité fonctionnelle, « Functionnal completeness » sur la Figure 6, permet de caractériser si le système fournit les fonctions nécessaires à l'utilisateur.
- L'efficience du développement, « Development efficiency » sur la Figure 6, permet de caractériser la manière dont les ressources disponibles (matérielles et humaines) ont été utilisées pendant la conception et le développement du système.

Ces propriétés internes sont fortement liées à l'ingénierie de conception et certaines d'entre elles peuvent être retrouvées dans les standards de conception d'un système informatique. Le standard ISO 9126, « Génie logiciel : qualité des produits », (International Standard Organisation, 2001), classifie aussi les différentes propriétés liées à la qualité du logiciel en deux catégories : internes et externes. On y retrouve la propriété de facilité de modification, citée dans le paragraphe précédent (« System modifiability »). Ce standard décrit aussi la propriété d'**opérabilité**, appelée « Operability », permettant d'évaluer la capacité de l'application logicielle à être exploitée et contrôlée par l'utilisateur. Cette propriété nous intéresse particulièrement car elle prend en compte la capacité du système à prévenir et faire face aux erreurs de l'utilisateur.

Certaines classifications s'intéressent à des types particuliers de systèmes interactifs. Les propriétés CARE (Complémentarité Assignation Equivalence et Redondance) sont proposées par (Coutaz, et al., 1995) afin de permettre la caractérisation des systèmes interactifs offrant des techniques d'entrée et/ou de sortie multimodales.

L'objet de cette section n'est pas de dresser une liste exhaustive des propriétés possibles d'un système interactif critique mais de montrer que le processus de développement d'un système interactif critique requiert d'établir les moyens :

- De concevoir un système capable de satisfaire de nombreuses propriétés, parfois en conflit les unes par rapport aux autres.
- D'évaluer les propriétés satisfaites par le système conçu.

Lors de la conception et du développement du système, il est important de sélectionner les propriétés requises et d'évaluer si le système conçu les rempli selon les objectifs fixés. Certains travaux fournissent des conseils et règles sur la manière de concevoir les systèmes interactifs, les plus largement cités étant ceux de (Bastien & Scapin, 1993) et (Nielsen, 1994). Cependant, ces recommandations et standards ne fournissent pas de lignes directives sur les moyens à mettre en œuvre.

2.2 Propriétés des systèmes critiques

Dans le domaine des **systèmes critiques**, (Avizienis, Laprie, Randell, & Landwehr, 2004) définissent les concepts de sûreté « Dependability » et de « Security » (garantie de fonctionnement) pour les systèmes informatiques afin de fournir un langage commun pour la conception et le développement de ce type de systèmes. Cette classification est décrite dans la Figure 7. Le premier sous-arbre, « Attributes », présente les propriétés liées aux concepts de fiabilité et de sécurité. Le second sous-arbre, « Threats », présente les menaces potentielles. Enfin, le troisième sous-arbre, « Means », propose des moyens de faire face aux menaces.

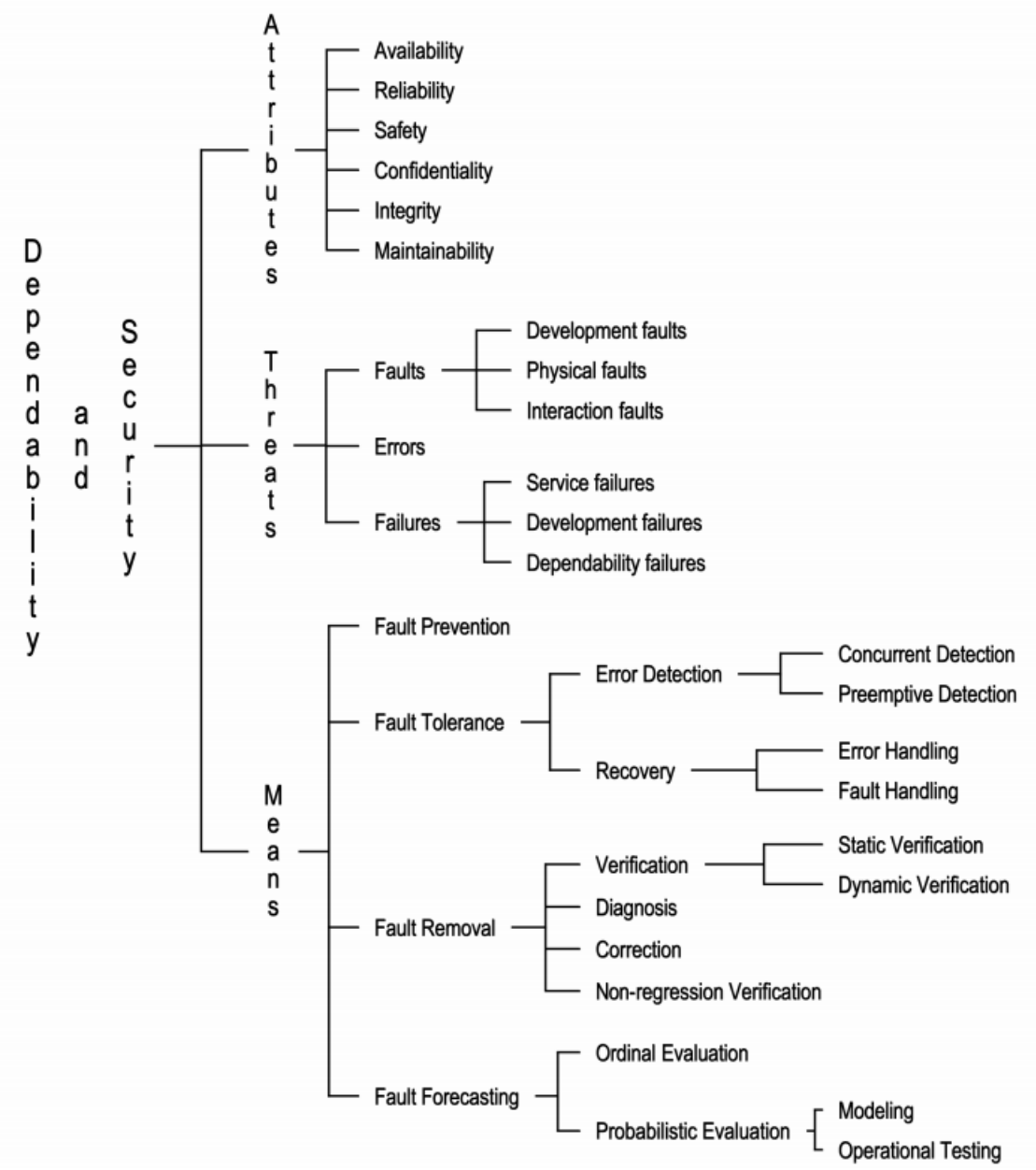


Figure 7. Arbre de Fiabilité et Sécurité extrait des travaux de (Avizienis, Laprie, Randell, & Landwehr, 2004)

Les propriétés proposés par (Avizienis, Laprie, Randell, & Landwehr, 2004), rassemblées dans la catégorie des attributs, « Attributes » sur la Figure 7, sont les suivantes (la notion de ***service*** est introduite dans leurs travaux pour décrire le comportement du système) :

- La disponibilité, « Availability » sur la Figure 7, permet de caractériser la capacité du système à fournir le service attendu par l'utilisateur, i.e. à être prêt à interagir correctement avec l'utilisateur.
- La **fiabilité**, « Reliability » sur la Figure 7, permet de caractériser la capacité du système à fournir le service attendu de manière continue.
- « Safety » sur la Figure 7, permet de caractériser la capacité du système à ne pas altérer l'intégrité de l'utilisateur et de son environnement.
- L'intégrité, « Integrity » sur la Figure 4, permet de caractériser la capacité du système à empêcher l'altération sa propre intégrité.

- La maintenabilité, « Maintainability » sur la Figure 7, permet de caractériser la capacité du système à être modifié et maintenu.

Les menaces pouvant entraver le fonctionnement du système et ainsi l'empêcher de satisfaire les propriétés explorées précédemment, sous-arbre « Threats » de la Figure 7, sont les fautes, les erreurs et les échecs. Un échec, « failure », est défini comme l'accomplissement incorrect d'un service fournit par le système. Une erreur, « Error », est définie comme le changement d'état du système d'un état correct vers un état incorrect. Une faute, « fault », est définie comme la cause hypothétique d'une erreur. Ainsi, nous nous intéressons particulièrement aux fautes car elles permettent de montrer l'importance des facteurs humains et des moyens mis en œuvre pour le développement d'un système.

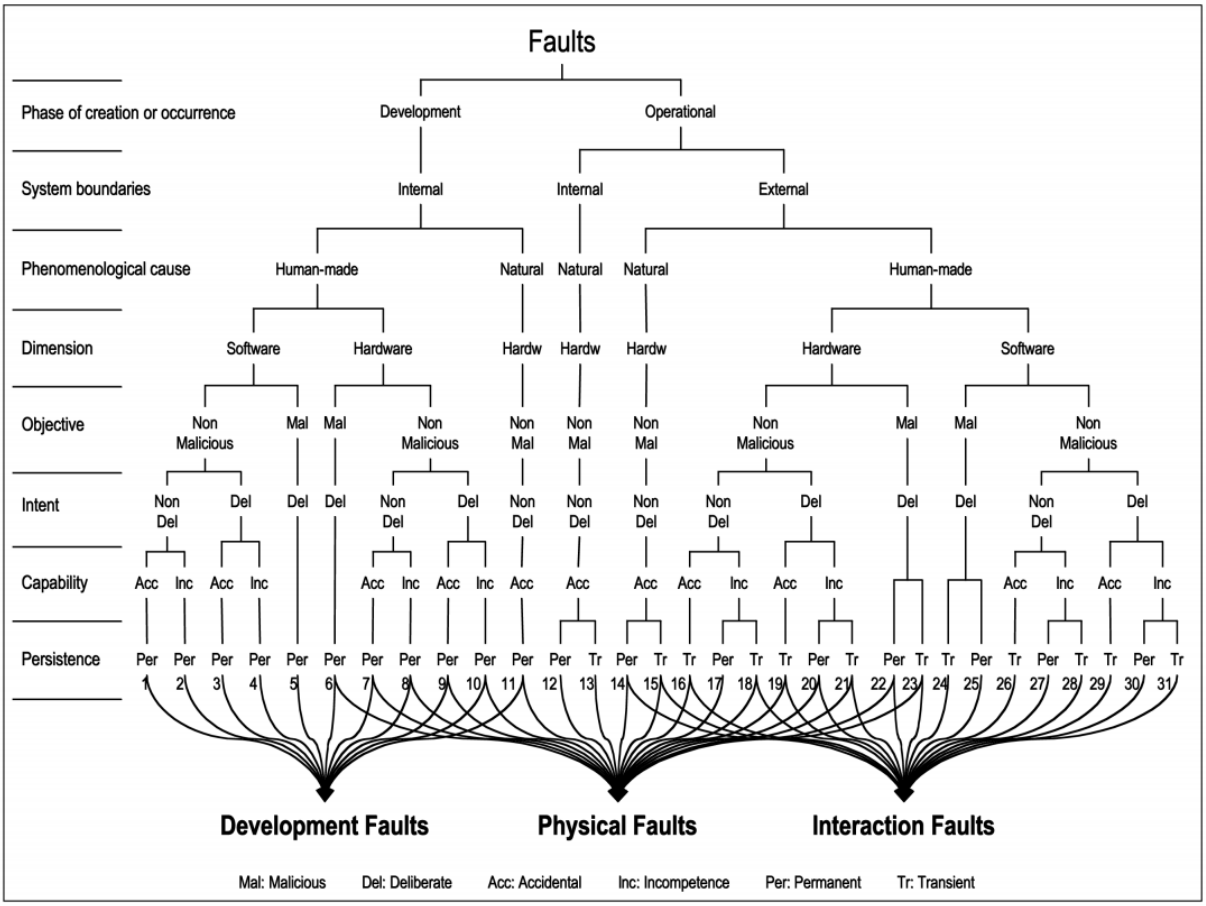


Figure 8. Arbre des fautes extrait de (Avizienis, Laprie, Randell, & Landwehr, 2004)

La Figure 8 présente l'arbre des fautes pouvant conduire à des erreurs et ainsi à mettre en péril le système et ses utilisateurs. Elle fait clairement apparaître un sous arbre lié aux fautes humaines commises pendant le développement du système ainsi qu'un sous arbre lié aux fautes humaines commises en exploitation. Des moyens sont donc nécessaires pour prévenir ces fautes :

- Au niveau des approches et processus de développement.
- Au niveau de la préparation des utilisateurs du système.

Les moyens de faire face à ces fautes, proposés dans le sous arbre « Means » de la Figure 7, sont la prévention, « Fault Prevention », la tolérance aux fautes, « Fault Tolerance », la correction des fautes, « Fault Correction », et la prévision des fautes, « Fault Forecasting ».

Lors du développement d'un système critique, l'approche mise en œuvre doit donc fournir un support à ces moyens faire face aux fautes.

Nous remarquons que la propriété d'opérabilité présente des caractéristiques communes avec l'utilisabilité et la fiabilité. En effet, la capacité du système à être exploité implique le fait que l'utilisateur puisse accomplir ses tâches de manière efficace et la capacité du système à faire face aux erreurs implique qu'il soit fiable.

3 Exploitation des systèmes interactifs critiques

L'exploitation d'un système interactif critique est la phase où un opérateur utilise le système. Plusieurs ressources conditionnent la mise en exploitation du système et sont décrites dans la sous-section 3.1. L'adéquation entre les compétences et connaissances des opérateurs et les fonctionnalités et comportement du système sont discutées dans la sous-section 3.2.

3.1 Ressources associées à la mise en exploitation des systèmes informatiques

A l'issue du développement d'un système informatique, plusieurs ressources sont produites par ses constructeurs, selon les domaines, pour son exploitation ou/et pour l'autorisation de exploitation :

a) La ou les configurations du matériel informatique.
b) Les éléments logiciels s'exécutant sur la ou les configurations matérielles.
c) Le code source des éléments logiciels.
d) La documentation associée aux configurations matérielles et aux éléments logiciels (spécification, architecture, détails de conception, choix de conception, résultats des campagnes de test…) pour certification pas exploitation.
e) Les manuels utilisateurs.
f) Le programme de formation des utilisateurs.
g) Les documents attestant de la certification conforme (ou non conforme) des configurations matérielles, des éléments logiciels et des compétences des utilisateurs. Les documents de certification sont produits par des organismes de certification sur examen de certaines ressources fournies par le constructeur du système.

La Table 1 récapitule ces ressources en indiquant lesquelles sont indispensables pour l'exploitation des systèmes interactifs et des systèmes critiques.

Table 1. Récapitulatif des ressources nécessaires à l'exploitation ou/et à l'autorisation de l'exploitation des systèmes interactifs et des systèmes interactifs critiques

Ressources	Systèmes interactifs	Systèmes interactifs critiques
Configuration du matériel informatique	Indispensable	Indispensable
Configuration logicielle exécutable	Indispensable	Indispensable
Code source	Indispensable	Indispensable
Documentation technique sur les configurations	Facultatif	Indispensable
Manuels utilisateurs	Facultatif	Indispensable
Programme de formation	Facultatif	Indispensable
Documents d'attestation de certification (produit par un organisme de certification sur examen de certaines ressources fournies par le constructeur)	Facultatif	Indispensable

La liste de ces éléments varie en fonction des recommandations et standards des différents domaines d'applications. Tout comme l'arbre des fautes dans la section précédente, cette liste non exhaustive permet de montrer que plusieurs éléments sont nécessaires autour du développement du système lui-même et de son exploitation pour s'assurer qu'il remplira les propriétés liées à sa nature (présentées dans les sections précédentes). Plusieurs éléments sont requis pour vérifier et valider le comportement du système. La spécification du système est un élément important associé au système car elle fournit un support pour les activités de validation et vérification des propriétés du système. Ainsi, une spécification formelle fournira ce support.

Les moyens mis en œuvre pour le développement de systèmes interactifs critiques sont discutés dans le chapitre 2.

3.2 Formation à l'utilisation des systèmes interactifs et des systèmes critiques

Cette section décrit les différentes approches pour former les utilisateurs des systèmes interactifs et des systèmes critiques et met en valeur les différences entre ces approches.

Dans le domaine des systèmes interactifs dits « grand public », le système conçu doit être utilisable par le plus grand nombre de personnes, ceci sans connaissances particulières ni formations effectuées au préalables. Le terme *walk-up and use* est aussi utilisé pour décrire ce type de systèmes. Les travaux fondateurs de la conception centrée utilisateur de (Norman D. A., 2002) soutiennent qu'un système interactif doit être « affordable », c'est-à-dire que dès sa première exploitation par un utilisateur novice, il doit permettre à l'utilisateur d'accomplir ses tâches avec le moins d'erreurs possibles (idéalement sans erreurs). Dans la communauté scientifique et académique de l'interaction homme-machine, certains travaux portent sur la formation des concepteurs de systèmes interactifs, (Bailey, 1993). Le standard ISO TR 18529 (International Standard Organisation, ISO TR 18529:2000 Ergonomics of human system interaction: Human-centred lifecycle process description, 2000) recommande une activité de développement et mise en place de formation dans un processus de développement centré utilisateur. Hormis ces travaux, aucune autre référence n'a été trouvée à ce jour sur le sujet de la formation d'un utilisateur à l'exploitation d'un système interactif.

Dans le domaine des systèmes critiques (et de manière générale dans le milieu industriel), les exigences de formation à l'utilisation du système sont conditionnées par le coût d'une erreur

d'utilisation, tout comme les moyens de conception et développement. Ces exigences de formation seront fixées en fonction des performances attendues lors de l'exploitation du système (Annett & Duncan, 1967). Dans le domaine des systèmes critiques en terme de sécurité, il n'est pas envisageable de laisser un utilisateur exploiter le système sans connaissance particulière ou formation effectuée au préalable. Dans les domaines de l'aéronautique, de l'aviation civile et militaire, du spatial, de l'énergie, les standards requièrent que le personnel exploitant soit formé et qualifié avant d'utiliser le système. Selon les domaines, les directives sont plus ou moins précises sur les moyens à mettre en œuvre et font l'objet d'une présentation dans le chapitre 3.

4 Conclusion

Ce chapitre nous a permis d'explorer les propriétés et caractéristiques d'un système interactif critique et de montrer leur complexité. Cette complexité est due en partie :

- A la nature imprédictible de l'interaction avec un être humain.
- Aux contraintes de conception et développement des systèmes informatiques liées au nombre croissant de leurs fonctionnalités.
- A la nécessité de maintien du bon fonctionnement du système malgré les fautes d'utilisation.
- Aux contraintes de conception et développement liées à la certification.

Nous avons aussi mis en valeur les propriétés d'utilisabilité, d'opérabilité et de fiabilité car les travaux de cette thèse fournissent des moyens pour concevoir et développer des systèmes interactifs dans le but qu'ils remplissent ces propriétés. Ainsi, le chapitre 2 parcourt les moyens (processus et approches de développement, notations et outils de modélisation) existants pour concevoir et développer des systèmes interactifs critiques, à la recherche d'une approche de développement prenant en compte leurs propriétés. Le chapitre 3 s'attache à décrire les moyens de formation existants pour les systèmes interactifs critiques dans le but de s'assurer que les utilisateurs puissent opérer le système de manière sûre.

Chapitre 2- Développement des systèmes interactifs critiques : approches et moyens

Ce chapitre présente les moyens existants pour développer des systèmes interactifs et des systèmes critiques afin de trouver lesquels fournissent un support au développement de systèmes interactifs critiques et/ou complexes, simultanément utilisables, fiables et opérables. Dans ce mémoire, le terme « **développer** » fait référence à l'enchaînement d'activités à mettre en œuvre pour produire un système et n'est pas seulement réservé au développement logiciel.

Dans le chapitre précédent, nous avons observé que le concept de système interactif critique n'est pas forcément étudié dans son ensemble mais plutôt selon les deux thématiques sous-jacentes que sont les systèmes interactifs et les systèmes critiques. De la même manière, les moyens de développement existants ne sont pas non plus dédiés à ce type de système mais se concentrent sur une thématique ou sur l'autre. Ce chapitre tente de mettre en valeur les spécificités des moyens existants mais aussi leurs limitations.

La section 1 de ce chapitre est consacrée aux approches, processus et recommandations pour le développement de systèmes informatiques qu'ils soient génériques, interactifs et/ou critiques.

L'utilisation de modèles pour la conception et le développement de systèmes informatiques est répandue car elle permet, entre autres de gérer la complexité du système développé. Ce moyen fait l'objet de nombreuses recherches dans le domaine de l'IHM (Hussman, Meixner, & Zuehlke, 2011) mais aussi dans le domaine du génie logiciel. Selby (Selby, 2009) identifie les principes importants pour la conception de systèmes critiques complexes et expose que la modélisation en fait partie. Ainsi, la section 2 identifie les différents types de modèles utilisés pour la conception et le développement de systèmes interactifs et de systèmes interactifs critiques ainsi que les notations et outils de modélisation employés.

Les modèles utilisés pour la conception des systèmes interactifs critiques pouvant être de différents types et chaque type représentant chacun une vision du système ou de son utilisation, plusieurs travaux se sont attachés à employer ces modèles de manière synergique. La section 3 est consacrée à ces approches d'intégration synergiques.

1 Approches, processus et recommandations pour le développement de systèmes interactifs critiques

Les approches, processus et recommandations pour le développement permettent de mettre en place un ensemble d'étapes conduisant à la production et à la livraison d'un système. Elles permettent de fournir un cadre conceptuel commun aux différentes parties prenantes du développement d'un système ; elles permettent de gérer la complexité et de faciliter la communication entre les différents intervenants. Selon leur degré d'aboutissement et leur but, les approches et processus fournissent un support pour maitriser les coûts, les temps de développement et l'industrialisation. Selon leur nature, elles intègrent ou non dans leur démarche un support pour développer un système utilisable, fiable et opérable ainsi qu'un support à la conception d'un programme de formation correspondant au système développé.

Cette section utilise et étend l'état de l'art des processus de développement proposé dans les travaux de (Ladry J.-F. , 2010). Nous exposons les approches, processus et recommandations existants pour le développement, dans le but d'analyser si toute ou parties de ces approches peuvent être appliquées au développement de systèmes interactifs critiques. Cette section décrit l'état actuel des contributions scientifiques sur :

- Les approches, processus et recommandations génériques pour le développement de systèmes informatiques (sous-section 1.1).
- Les approches, processus et recommandations pour le développement de systèmes interactifs (sous-section 1.2) afin de montrer les spécificités, améliorations et ajouts effectués au cours du temps sur les processus génériques pour satisfaire les propriétés d'utilisabilité d'un système.
- Les approches, processus et recommandations pour le développement de systèmes critiques (sous-section 1.3) afin de mettre en valeur les spécificités, améliorations et ajouts effectués au cours du temps sur les processus génériques pour satisfaire les propriétés de fiabilité d'un système.

Chacune des sous-sections contient aussi, le cas échéant, les spécificités du domaine dans les approches de développement afin de mettre en lumière les contraintes existantes sur les approches de développement.

Au vu du nombre important d'approches et processus de développement existants, cette section ne fournit pas de descriptions détaillées de chacun d'entre eux mais tente de mettre en valeur les principes et limitations de chaque approche représentative. De plus, bien que la plupart de ces approches et processus soit applicables à l'ensemble d'un système informatique, la recherche et l'analyse ont été effectuées dans une perspective de développement logiciel, la partie sur laquelle nous nous concentrons pour le développement de systèmes interactifs critiques.

La conclusion de cette section discute de la possibilité d'utiliser tout ou partie des approches et processus présentés pour le développement d'un système interactif critique.

1.1 Approches, processus et recommandations génériques de développement

Cette sous-section s'intéresse à examiner les approches génériques de développement afin d'en dégager les types de systèmes (interactifs, critiques) auxquels elles sont favorables.

1.1.1 Processus séquentiels

1.1.1.1 Processus de développement en cascade

Le **processus de développement en cascade** décrit par la Figure 9, (Royce, 1970), est une séquence d'activités dont l'enchaînement ordonné permet à chaque activité de profiter des réflexions et des artefacts produits par la précédente.

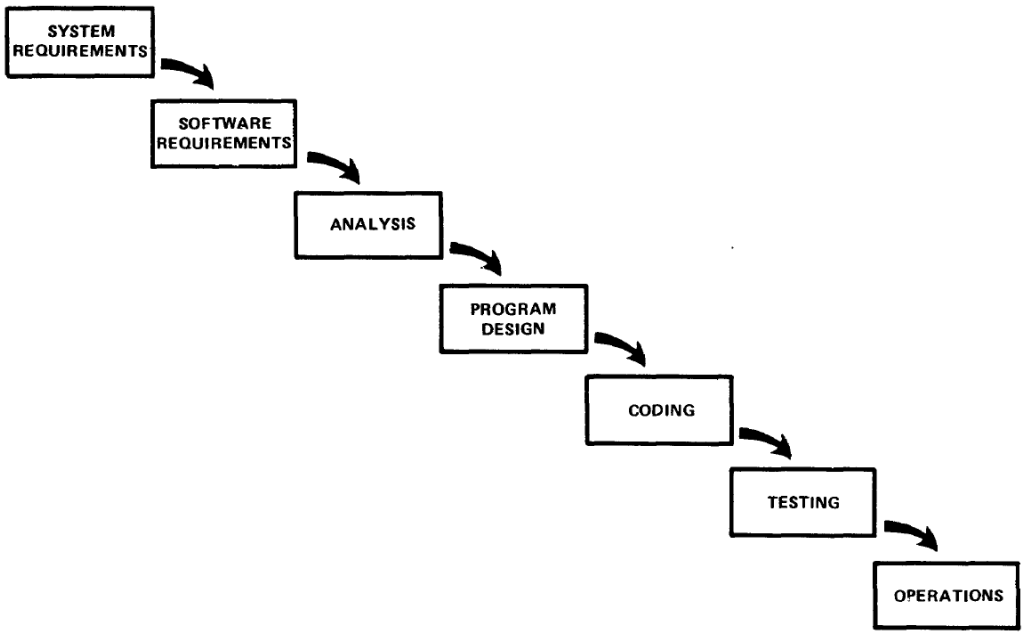


Figure 9. Processus de développement en cascade (Royce, 1970)

Des problèmes d'utilisabilité, des problèmes techniques ou des réajustements nécessaires dans la définition des besoins peuvent émerger relativement tard dans le cadre de ce processus. Cela peut se produire au moment de la confrontation des utilisateurs avec l'application, au cours des tests finaux, voire après le déploiement de l'application. Des retours sont alors proposés vers des phases antérieures dans le processus. Par exemple en cas de dysfonctionnements rencontrés lors de la mise en production (« OPERATIONS » sur la Figure 9), il peut être nécessaire de retourner à la phase d'implémentation (« CODING » sur la Figure 9). Si un retour en arrière est nécessaire vers une phase de début de cycle, le coût de conception peut être élevé. Les versions plus récentes du cycle de vie en cascade font apparaître à chaque étape une activité de vérification (McConnell, 1996) : la gestion de projet doit garantir la validité du passage d'une étape à l'autre. Le cycle de vie en cascade est aujourd'hui utilisé dans des projets de très grande envergure pour lesquels une spécification solide des besoins est nécessaire, rendant peu probable la redéfinition de ceux-ci. Malgré tout, quel que soit le domaine d'application, des retours vers des étapes antérieures se produisent et ce type de retour doit passer au travers de chaque étape engendrant des retards de développement.

1.1.1.2 Processus de développement en V

Le cycle de développement en V (McDermid & Ripken, 1983) détaille le processus de développement de telle sorte qu'à chaque étape productrice d'artefact (l'analyse des besoins, le document de spécifications, l'architecture et le logiciel proprement dit) corresponde une étape qui évalue cet artefact. Trois catégories d'étapes se dégagent (Figure 10):

- Étapes dans la phase descendante (le bras gauche du V) qui couvrent toutes les étapes de raffinement des besoins jusqu'à l'implémentation de l'application. Les étapes de conception sont détaillées en fonction de la granularité des problèmes considérés, des plus abstraits aux plus concrets.
- Codage de l'application (la base du V) qui vient suite à cette phase de conception.
- Étapes dans la phase ascendante, ensuite (le bras droit du V), chacune correspondant à un contrôle d'une étape de la phase descendante.

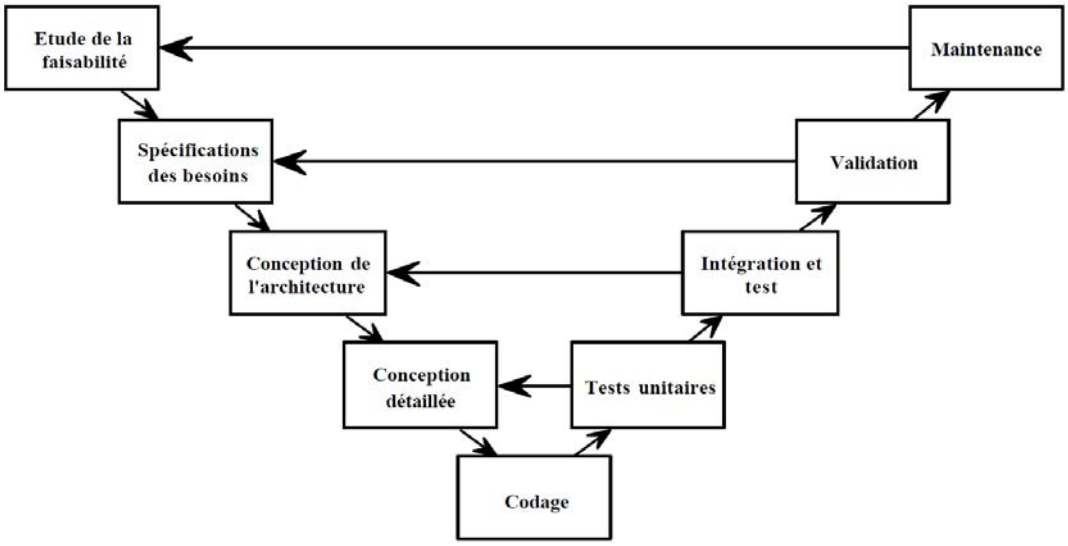


Figure 10. Processus de développement en V (McDermid & Ripken, 1983)

A chaque étape de conception correspond une étape de vérification. Certains tests de vérification sont réalisables directement lors de l'étape de conception qui leur correspond, mais la plupart des tests ont lieu après la phase de codage, conformément au sens de lecture du cycle en V. Les tests à effectuer dans la phase ascendante sont à spécifier dès l'étape de phase descendante associée. Par exemple, en phase de spécification, les personnes responsables de cette phase créeront un jeu de tests permettant de vérifier que l'application finale sera conforme aux spécifications ayant émergé lors de cette phase de conception. Cette vérification aura lieu lors des tests de validation. En cas de correction à effectuer, il y a un retour vers le début du cycle, et un réajustement nécessaire des phases suivantes, ainsi qu'une mise à jour des jeux de tests qui seront à effectuer pour la phase de validation.

La flexibilité et la large renommée du cycle en V font qu'il est utilisé dans de nombreuses entreprises. La documentation nécessaire pour la mise en correspondance d'une phase de production (ligne descendante) et de la phase de vérification qui lui est associée (ligne ascendante) sert à la gestion de projet et à la cohérence de la conception.

Ces types d'approches fournissent un support avéré à la traçabilité des choix de conception et des exigences pour le système tout au long du cycle de développement. Ainsi, combinées à d'autres moyens, elles sont actuellement utilisées dans le domaine des systèmes critiques (section 1.3). Cependant, elles ne fournissent aucun support pour le développement d'un système utilisable.

1.1.2 Les processus itératifs et/ou incrémentaux

1.1.2.1 Processus de développement en spirale

Le **cycle de développement en spirale** (Boehm, 1986) correspond à plusieurs itérations sur quatre étapes (Figure 11) dont le contenu évoluera au fil des itérations, mais dont la nature reste la même :

- Définition des objectifs, des alternatives possibles et des contraintes du projet
- Évaluation des alternatives en réponse aux besoins et en tenant compte des contraintes: cette phase comporte un travail de prototypage, et le prototype sera de plus en plus opérationnel au fur et à mesure des itérations
- Conception d'une solution apparaissant comme la meilleure des alternatives étudiées dans la phase précédente : cette phase englobe les étapes de conception et de vérification déjà observées dans les cycles de développement en cascade et en V (spécification, conception préliminaire puis conception détaillée, tests unitaires puis tests d'intégration et déploiement)
- Planification de la phase suivante et notamment l'affectation des tâches à réaliser au sein de l'équipe de conception

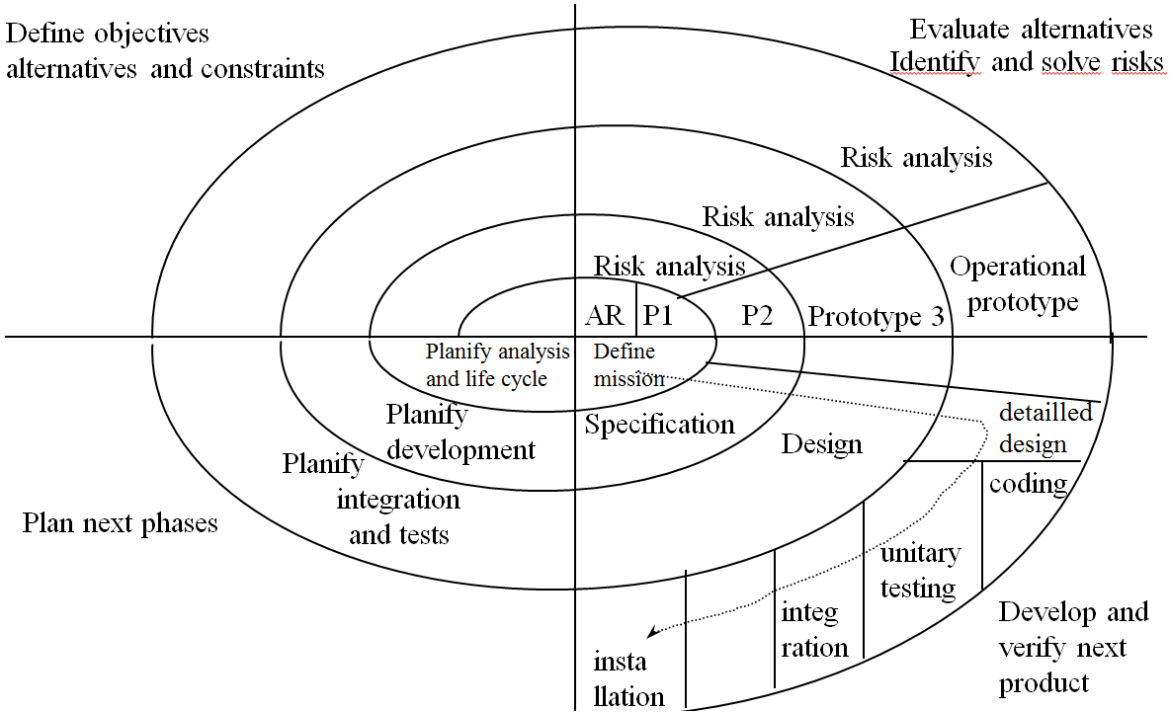


Figure 11. Processus de développement en spirale (Boehm, 1986)

Ce cycle de développement permet ainsi d'explorer plusieurs alternatives. Il s'agit en réalité de travailler itérativement à la définition des besoins et à la spécification de l'application. Lorsque la réflexion a atteint un degré de raffinement permettant la prise de décision, l'équipe de conception enchaîne avec un cycle de développement en cascade ou en V pour mener à bien les phases concernant la conception détaillée puis l'implémentation proprement dite. Bien qu'il prenne en compte l'utilisabilité finale du système développé, ce cycle de développement est long et coûteux à mettre en place.

1.1.2.2 Rational Unified Process

Rational Unified Process (RUP) (Kruchten, 2004) est un processus de développement d'applications logicielles dont le but est d'assurer la production d'un produit répondant aux besoins des utilisateurs. RUP est un processus :

- Itératif et incrémental : le projet est découpé en itérations de courte durée, à la fin de chaque itération une version exécutable de l'application est fournie, cette version est incrémentée à chaque itération.
- Basé sur la modélisation : l'architecture logicielle doit être modélisée graphiquement. Ce processus est ainsi basé sur la notation UML (Unified Modeling Language), (Object Management Group, 2011).

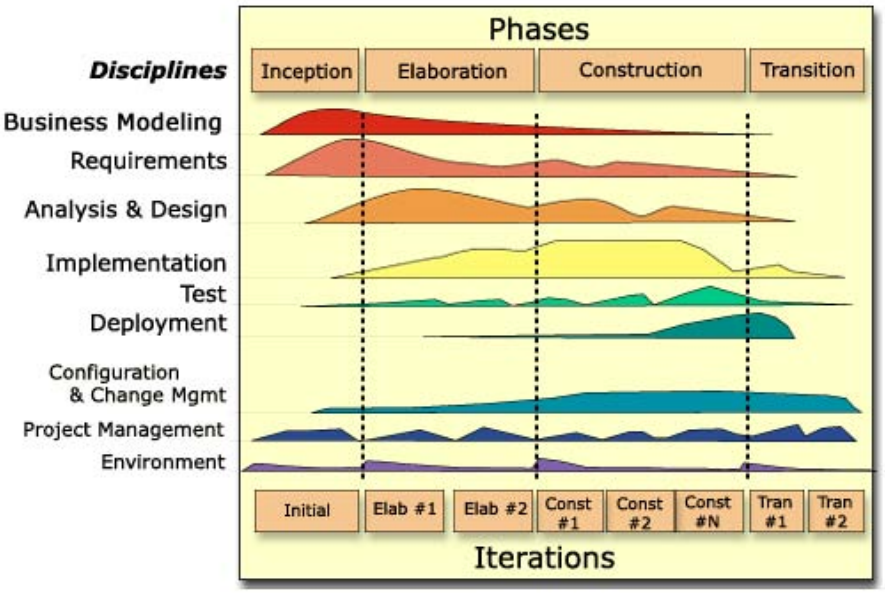


Figure 12. Processus de développement RUP

La Figure 12 présente une vue d'ensemble du processus, le processus est décomposé en deux dimensions :

- L'axe horizontal représente le temps et le déploiement du cycle de vie comprenant le lancement, l'élaboration, la construction et la transition.
- L'axe vertical représente les disciplines, les intervenants, les artefacts impliqués dans le processus comprenant la modélisation des processus de gestion, les exigences, la conception et analyse, l'implémentation, les tests, le déploiement, la gestion de projet, la gestion des changements et l'environnement.

RUP est une approche «clef en main » proposé par IBM[2] qui fournit un processus et des outils. Les outils proposent des canevas de projets, un partage des documents entre les intervenants et favorisent aussi la réutilisation des modèles. Cependant l'approche est liée aux outils Rational, et bien qu'elle soit favorable à la production d'un système utilisable, les outils quasi indispensables ont un coût significatif.

1.1.2.3 Méthodes AGILES

Les **méthodes AGILE** (Cockburn, 2002) officialisées en 2001 par le Manifeste Agile (AGILE, 2001) visent à la satisfaction réelle du client en l'impliquant dans le processus de développement. Elles prônent quatre valeurs fondamentales :

[2] http://www-01.ibm.com/software/awdtools/rup/

- L'équipe (« Personnes et interaction plutôt que processus et outils »)
- L'application (« Logiciel fonctionnel plutôt que documentation complète »)
- La collaboration (« Collaboration avec le client plutôt que négociation de contrat »)
- L'acceptation du changement (« Réagir au changement plutôt que suivre un plan »)

Les approches SCRUM (pour la gestion du développement) et eXtreme Programming (pour la programmation logicielle) suivent ces préceptes.

- **SCRUM** (Shwaber & Beedle, 2002) est une approche de gestion et de suivi d'avancement du développement logiciel. Elle est itérative et incrémentale.
  Un représentant du client nommé propriétaire du produit (*product owner*) est responsable de transmettre à l'équipe l'orientation du projet, de définir les fonctionnalités et de suggérer l'ordre dans lequel elles devraient être développées par l'équipe afin de fournir un logiciel qui en adéquation avec ses besoins. Il consigne ces informations dans le carnet de produit (*product backlog*). Le carnet de produit est une liste d'éléments, représentant des besoins ou fonctionnalités désirées par le propriétaire du produit. Les éléments y sont classés selon la valeur d'affaire accordée par le propriétaire du produit. La valeur accordée aux éléments par le propriétaire dépend des critères qu'il considère : le retour sur investissement (ROI), la criticité d'une fonctionnalité dans le système ou pour les utilisateurs, le coût en effort de développer la fonctionnalité, etc. Le carnet de produit est visible pour toute l'équipe et permet ainsi une meilleure communication des objectifs puisque tous les membres de l'équipe peuvent voir les fonctionnalités demandées par le propriétaire du produit.

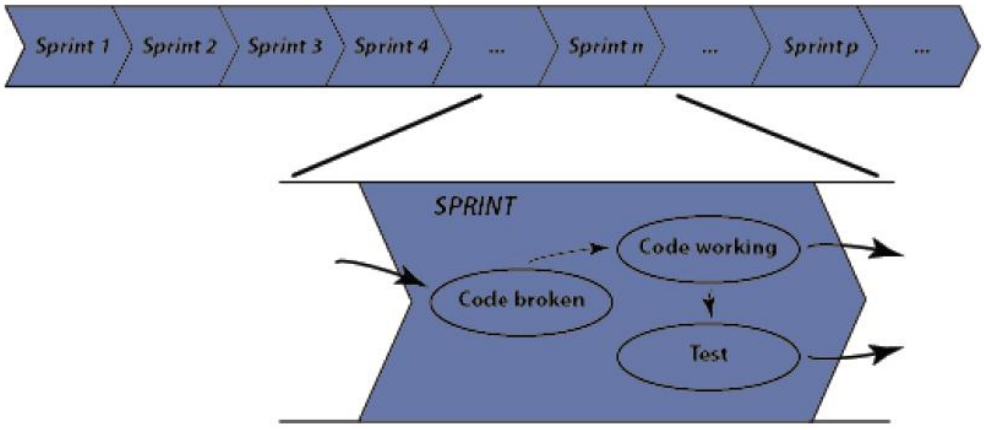


Figure 13. Déroulement des phases de développement logiciel avec l'approche SCRUM

Le travail de développement est découpé en itérations (*sprints*) qui ont généralement une durée de trois ou quatre semaines. Cette courte durée permet de livrer rapidement au propriétaire du produit un incrément de logiciel comportant le plus de valeur d'affaire. Le logiciel livré a de la valeur pour le propriétaire du produit puisque les fonctionnalités développées sont celles qui ont la plus haute priorité, celles se trouvant en tête du carnet de produit. Un autre avantage de la courte durée des itérations est de permettre au propriétaire du produit de modifier les priorités des fonctionnalités demandées au fur et à mesure de l'avancement du projet de développement, et ce, selon l'évolution de sa réalité d'affaire. En effet, s'il désire ajouter des fonctionnalités, celles-ci seront potentiellement prises en compte lors de la prochaine itération.
Cette approche de développement est fortement centrée sur les besoins du client mais, dans le cas où le client n'est pas l'utilisateur final, elle ne prend pas en compte l'utilisabilité du produit.

- **L'eXtreme Programming (XP)** (Beck, 1999) (présentée sur la **Figure 14**) définit un ensemble de pratiques techniques destinées à guider la production du logiciel.

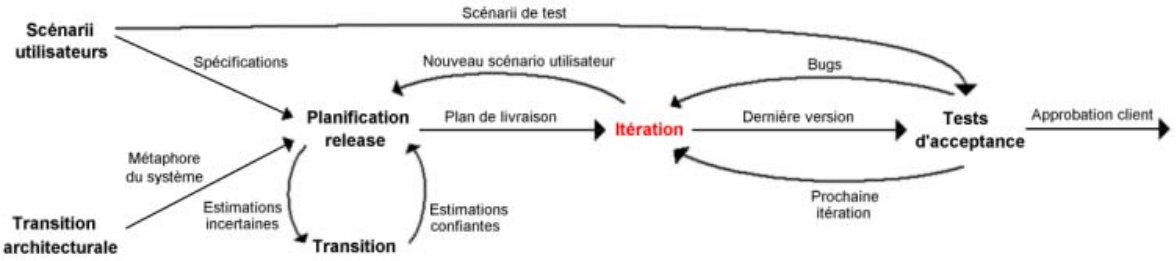


Figure 14. Processus de développement eXtreme Programming

Les scenarios utilisateurs (user stories) sont similaires aux cas d'utilisation définis par UML et remplacent les spécifications des clients. Ils sont écrits en langage naturel par les clients. Ces scenarios utilisateurs permettent l'évaluation (par les concepteurs) de la durée de l'itération et sont ensuite utilisés pour réaliser les tests.

Une itération démarre par une planification (release planning) de l'itération, la réunion définit les scenarios à implémenter ainsi que les tests à mettre en œuvre. La partie planification est réalisée par les concepteurs et les clients. Les clients choisissent les scénarii à intégrer pour la prochaine itération, et les concepteurs évaluent la durée de l'itération. Les solutions de pointe (spike solution) se font en explorant plusieurs solutions potentielles et sont envisagées uniquement pour les problèmes difficiles à résoudre. Pour chaque scénario un planning est fixé: un scénario doit être implémenté entre une et trois semaines, sinon il doit être décomposé. Un test représente un comportement spécifique attendu du système. Un scenario utilisateur est validé lorsque tous les tests unitaires établis pour ce scenario sont passés. Lorsque les tests sont réalisés avec succès pour un scenario, la version courante de l'application est mise à jour avec ses dernières fonctionnalités (small releases).

L'eXtreme Programming définit en plus du processus de conception des règles et des pratiques à associer au processus (planification, codage, conception et tests). L'avantage de cette méthode est sa facilité de mise en œuvre, son faible coût, ainsi que la fréquence élevée des itérations et des livraisons de versions. Le temps est principalement investi dans les aspects techniques: développement et tests. Cependant la méthode ne couvre pas les phases en amont et aval au développement comme la capture des besoins utilisateurs, le support et la maintenance.

De manière générale, les méthodes AGILE, par leur nature itérative et incrémentale et par l'implication qu'elles requièrent de la part du client, sont intéressantes pour maîtriser les coûts et temps de développement et améliorer la satisfaction du client. Cependant, elles ne fournissent aucun cadre pour s'assurer de la fiabilité ou de l'utilisabilité du produit livré.

1.1.3 Les approches basées sur l'ingénierie dirigée par les modèles

L'Ingénierie Dirigée par les Modèles (IDM) ou « Model Driven Engineering » (Schmidt, 2006) a pour objectif de modéliser l'application que l'on veut créer de manière indépendante de l'implémentation cible (niveau matériel ou logiciel). Ceci afin d'améliorer la productivité de même que l'évolutivité (le développement d'un nouveau module, l'implémentation sur une nouvelle plate-forme).

L'IDM est basée sur trois types de modèles. Les modèles indépendants de l'implémentation (PIM - Platform Independent Model) sont associés à des modèles de plate-forme (PM - Platform Model) et transformés pour obtenir un modèle d'application spécifique à la plate-forme (PSM - Platform Specific Model). Des outils de génération automatique de code permettent ensuite de créer le programme directement à partir des modèles.

Parmi les processus de développement possibles pour mettre en place l'IDM, figure l'approche incrémentale qui est basée sur les étapes suivantes:

1) Prototype : Réalisation d'une architecture comportant les fonctionnalités minimales (IHM et fonctionnalités clés) selon la méthode linéaire.
2) Architecture complète : Intégration des différents principes de fonctionnement : type de client (lourd/léger), support de mobilité, etc., selon la méthode linéaire. On obtient un squelette d'application, comportant tous ses éléments, mais sans implémentation.
3) Mise en place des fonctionnalités : Compléter chaque fonctionnalité l'une après l'autre, selon la méthode linéaire.
4) Récursivité : Les étapes 2, 3 peuvent être réalisées par suite d'affinements successifs. Chaque affinement doit permettre de nouveaux types d'usage (extension des fonctionnalités), les versions intermédiaires doivent être exploitables.
5) Validation de la version finale
6) Evolutions : Elles peuvent se faire selon le même principe que les affinements successifs ayant conduit à la version complète du produit.

En étendant l'IDM au cycle de développement d'un système, (Schmidt, 2006) propose l'utilisation de processus de développement centrés sur les modèles « Model-Centric Software Development ». Cependant, ce processus est très générique et centré sur la conception du système. Par exemple, il ne décrit pas les phases d'analyse des besoins.

Cette approche met en évidence le besoins de modèles de différents types et d'outils de modélisation dans les processus de développement pour gérer la complexité croissante des systèmes informatiques. Cependant, elle ne prend pas en compte les propriétés d'utilisabilité du système développé.

1.1.4 ISO/IEC - IEEE 12207 : recommandations pour les processus de développement

Le standard ISO/IEC - IEEE 12207 (International Standard Organisation, ISO/IEC 12207:2008(E), Systems and Software Engineering - Software Life Cycle Processes, 2008) a pour objet de fournir un cadre commun aux différentes parties prenantes impliquées dans le cycle de vie d'un système informatique. Ce standard considère le logiciel comme une partie intégrante du système et des processus de conception du système. La Figure 15 présente les différents ensembles de processus pouvant être mis en œuvre pour développer un système informatique.

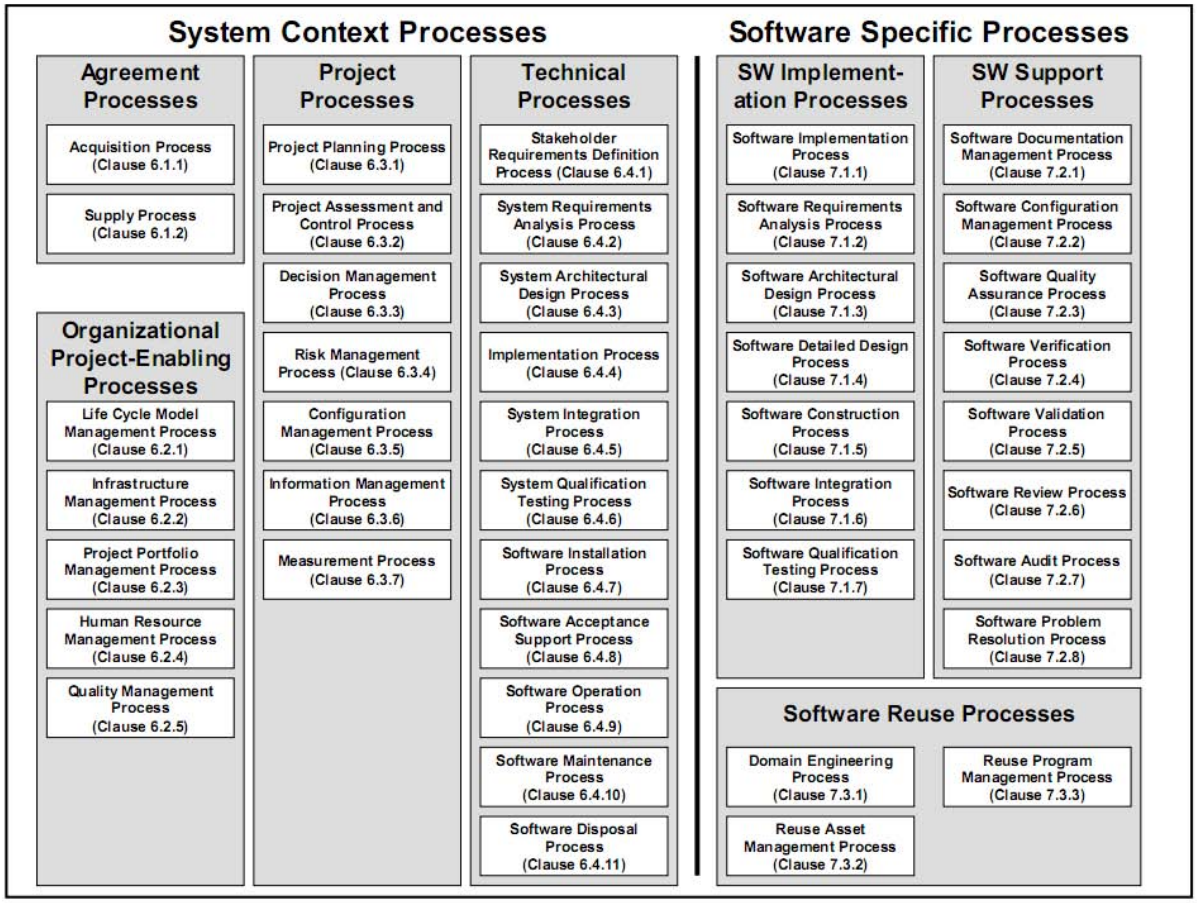


Figure 15. Cadre conceptuel des processus de développement issu du standard ISO/IEC-IEEE 12207 :2008

Ces différentes phases sont très génériques et le standard ISO/IEC – IEEE 12207 a pour but de fournir des recommandations pour l'établissement d'un processus de développement d'un système sans imposer un enchaînement d'étapes particulier ni indiquer des moyens de mise en œuvre. Ce standard définit aussi la notion de *vue processus* (« Process view ») pour personnaliser le processus global en fonction des objectifs de développement du système. Par exemple, dans le cas d'un système interactif centré utilisateur, si la vérification de la propriété d'utilisabilité est l'objectif principal lors de la conception du système, ce standard propose d'appliquer d'autres standards IEEE fournissant des recommandations supplémentaires sur les processus de développement centrés utilisateurs. Ainsi, il le standard ISO TR 18529 (International Standard Organisation, ISO TR 18529:2000 Ergonomics of human system interaction: Human-centred lifecycle process description, 2000) est recommandé pour la mise en place d'un processus de développement centré utilisateur. Ce dernier est exposé dans la section 1.2.6.

1.2 Approches, processus et recommandations pour le développement de systèmes interactifs

Les approches et processus de développement des systèmes interactifs amènent des techniques et étapes supplémentaires pour prendre en compte l'utilisateur. Par rapport aux approches et processus présentés dans la section précédente, de nouvelles étapes sont nécessaires :

- L'analyse des besoins de l'utilisateur, de son comportement et de ses activités pour concevoir un système en adéquation.

- Le prototypage du système pour pouvoir le confronter à l'utilisateur avant d'engager trop de frais de développement sans être sûr qu'il sera accepté par l'utilisateur.
- L'évaluation informelle des prototypes et du système en utilisation pour s'assurer de l'adéquation entre l'utilisateur et le système.

Les techniques courantes employées dans la conception centrée utilisateur sont décrites par (Dix, Finlay, Abowd, & Beale, 2003, 3rd edition).

1.2.1 Le processus de développement en étoile

Le modèle présenté par la Figure 16 est le modèle en étoile (Hartson & Hix, 1989). Chacune des boîtes aux extrémités des branches de l'étoile représente une phase du processus de développement et est un point d'entrée potentiel de ce processus. Le principe général de ce modèle est que, quelle que soit la phase accomplie, on doit passer par une phase d'évaluation, pour ensuite procéder aux autres phases le cas échéant. Sur le schéma, chaque flèche aller-simple représente un point d'entrée possible dans le processus, et les flèches aller-retour représentent la nécessité d'une évaluation après chaque phase.

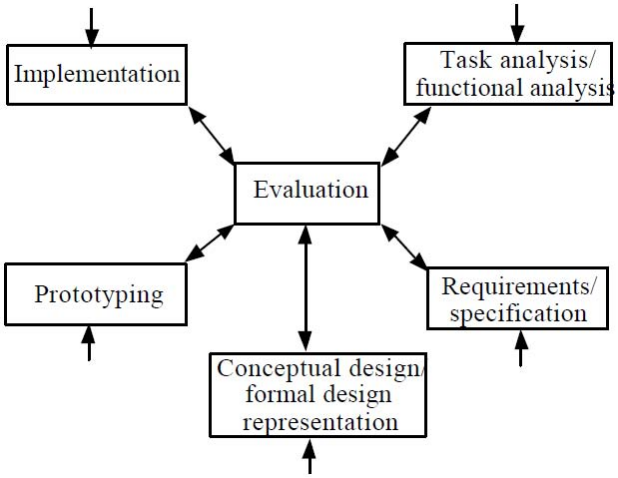


Figure 16. Processus de développement en étoile

Particulièrement ouvert, ce modèle est très flexible et n'indique pas un cheminement particulier dans le processus.

1.2.2 Le processus de développement en couches

La Figure 17 présente un modèle de processus de développement en couches issu de (Curtis & Hefley, 1994). Le principe général de ce modèle est de proposer une séparation en deux parties de chaque phase du développement, une partie concernant le développement classique de logiciel (« SOFTWARE ENGINEERING », partie de droite), et une partie concernant le problème de l'interaction homme -machine (« USER INTERFACE ENGINEERING », partie de gauche).

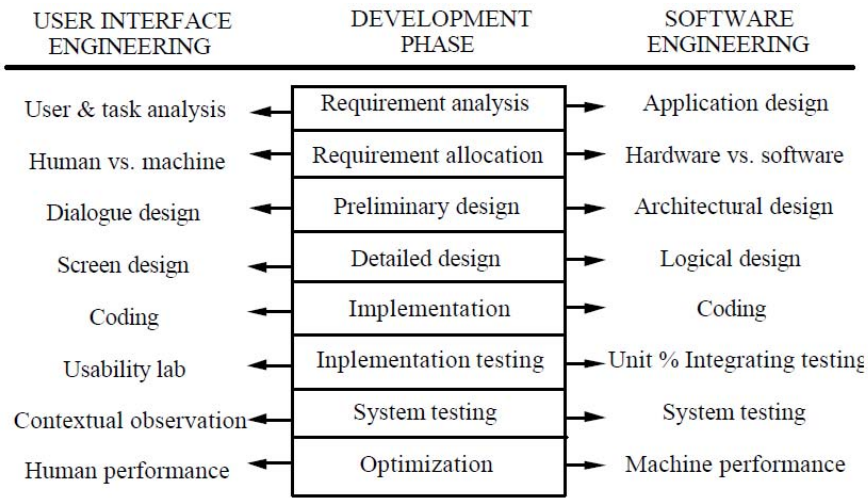


Figure 17. Processus de développement en couches

Ce processus met précisément en relation, pour chaque phase du développement, la phase de développement logiciel (comme le design architectural et les tests système) ainsi que la phase de développement de l'interface utilisateur (comme le design du dialogue et les tests d'utilisabilité). Ce processus conserve une séparation claire entre ces deux parties d'un système interactif.

1.2.3 Le processus de développement en cercle

Le processus de développement en cercle (Collins, 1995) présenté par la Figure 18 est un processus itératif. Il accorde une grande importance à l'analyse des tâches, la reconnaissant ainsi comme étape clé dans une conception centrée sur l'utilisateur. Il en résulte une tentative d'intégration des facteurs humains et du génie logiciel. Ce processus fournit plus une description de la relation entre facteurs humains et génie logiciel qu'un moyen précis de développer un système interactif.

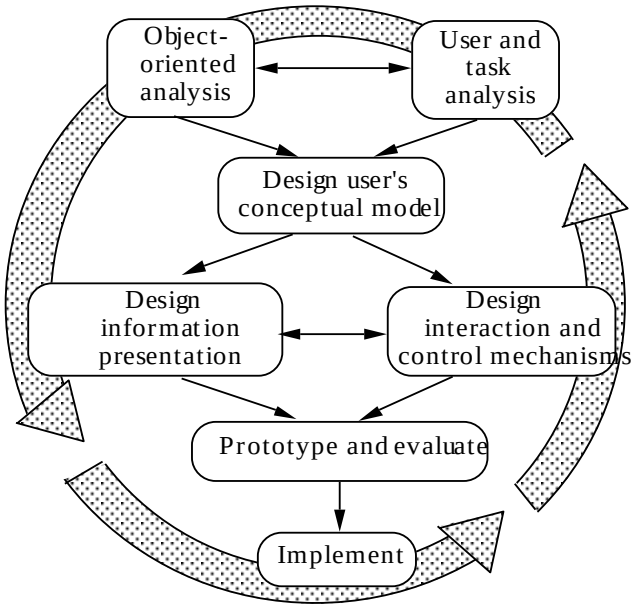


Figure 18. Processus de développement en cercle

1.2.4 Le processus itératif-cyclique

Le processus de développement itératif-cyclique (Rauterberg, 1992) est un processus participatif centré sur l'utilisateur. Il est composé de quatre quadrants (Figure 19) :

- En haut à gauche, l'analyse : Les tâches utilisateurs et les exigences sont recueillies
- En bas à gauche, la spécification : l'interface est créée ainsi que la description conceptuelle et organisationnelle de l'interface
- En bas à droit, l'implémentation : l'application est créée
- En haut à droite, les essais et la validation. : L'exactitude technique est testée et la validation des exigences utilisateurs sont vérifiées.

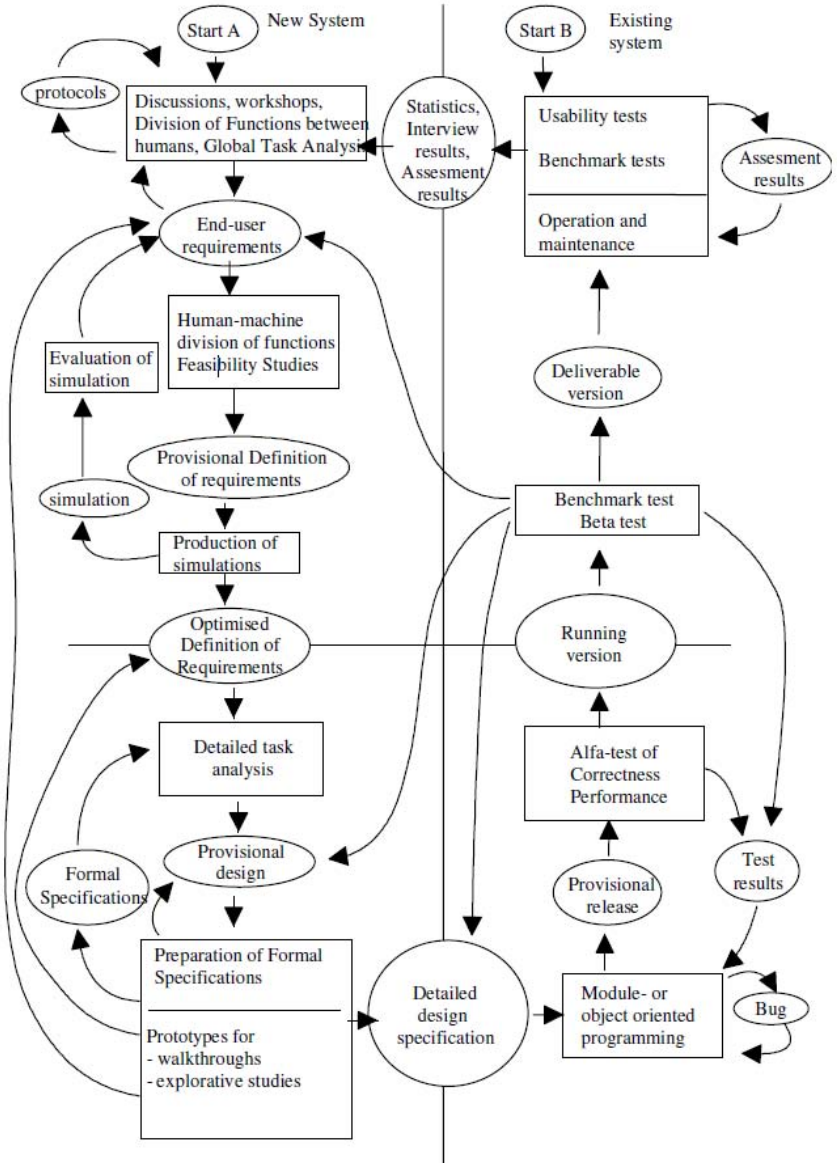


Figure 19. Le processus de développement itératif-cyclique

Le processus de développement itératif-cyclique ajoute aux traditionnels processus de développement :

- Les exigences de l'utilisateur final en phase d'analyse des besoins
- L'utilisation de prototypes dans la phase de spécification
- Une étape de spécification formelle dans la phase de spécification
- Les tests d'utilisabilité en phase de validation

Ce processus fournit un support au développement d'un système interactif utilisable. Cependant, il ne fournit pas de support pour le développement d'un système interactif fiable. En effet, l'utilisation de la spécification formelle est surtout recommandée pour les fonctionnalités haut-niveau afin de s'assurer que l'utilisateur et les concepteurs ont la même vision du futur système, et non pour décrire la totalité du système de manière complète et non-ambigüe.

1.2.5 User-centered System Design

Le User-centered System Design (USCD) (Gulliksen & Goransson, 2003) est un processus dédié à l'utilisabilité tout au long du processus de développement ainsi que tout au long du cycle de vie du système.

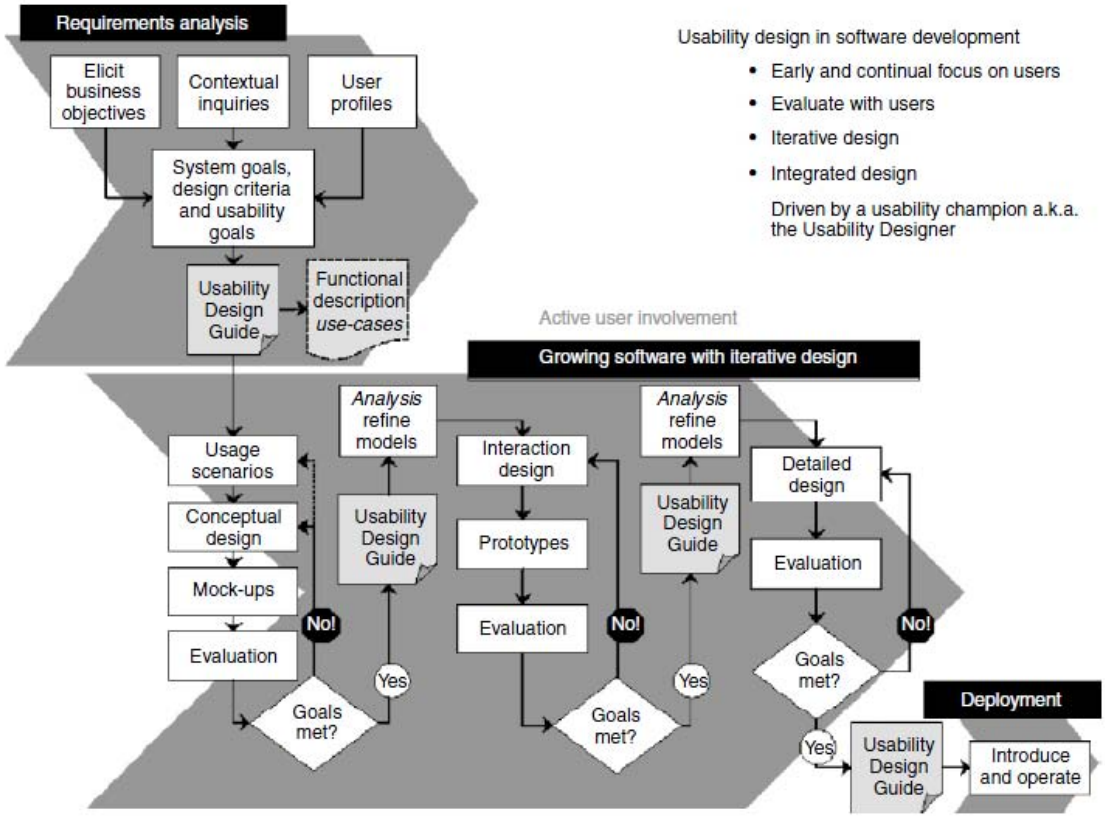


Figure 20. Usability Design Process

Ce processus met en valeur les éléments suivants :

- L'orientation utilisateur.
- L'engagement actif de l'utilisateur : dès le début du processus et tout au long de celui-ci.
- Le développement itératif et incrémental.
- La représentation simple du développement : compréhensible par les utilisateurs et les autres parties prenantes.
- La mise en place de prototypes.
- Les objectifs d'utilisabilité doivent conduire le développement.
- Les activités de développement explicites et claires.
- Le processus de développement devra être conduit par une équipe pluridisciplinaire comprenant notamment des experts en utilisabilité pour toute la durée du processus.
- Un développement holistique : développement en parallèle des aspects qui influenceront la future utilisation.
- Une customisation du processus selon l'organisation.

Ce processus a pour but de donner un poids égal au design d'interaction, à l'analyse et à l'évaluation. Ce processus peut être divisé en trois phases (Figure 20) : l'analyse des besoins, la conception du logiciel avec un développement itératif, et le déploiement. Ce processus impose de livrer un *Usability Design Guide*. Ce document devra contenir les informations provenant de l'analyse des besoins ainsi que les différents choix de conception durant le processus.

Dans la phase d'analyse des besoins, l'équipe se consacre à la compréhension des objectifs de l'entreprise, des tâches et des besoins des utilisateurs finaux et établit les objectifs du système du développement et d'utilisabilité. Cette partie évolue continuellement durant le processus de développement tant que des informations nécessaires peuvent y être ajoutées.

La deuxième phase contient trois boucles principales : le design conceptuel, le design d'interaction et le design détaillé. Ces boucles sont itératives comprenant le design, l'évaluation, le re-design, la réévaluation etc.., ce design devenant de plus en plus détaillé.

Enfin la phase de déploiement contient les différentes aides à l'utilisation du système : manuels, aides en ligne, entrainements. Cette phase n'est pas nécessairement finale. Elle peut simplement être la fin d'un cycle de ce processus itératif. Le déploiement à chaque cycle permet de présenter des sous-parties du système et donne l'opportunité de réaliser des ajustements sur le système ou le processus.

Néanmoins, malgré le fait que ce processus est fortement orienté vers l'utilisabilité, il ne traite pas de certaines phases de conception. Il doit donc être intégré dans un autre processus plus complet.

1.2.6 Les recommandations ISO TR 18529

Le standard ISO TR 18529 (International Standard Organisation, ISO TR 18529:2000 Ergonomics of human system interaction: Human-centred lifecycle process description, 2000), met en valeur des étapes nécessaires dans un processus de développement d'un système centré utilisateur.

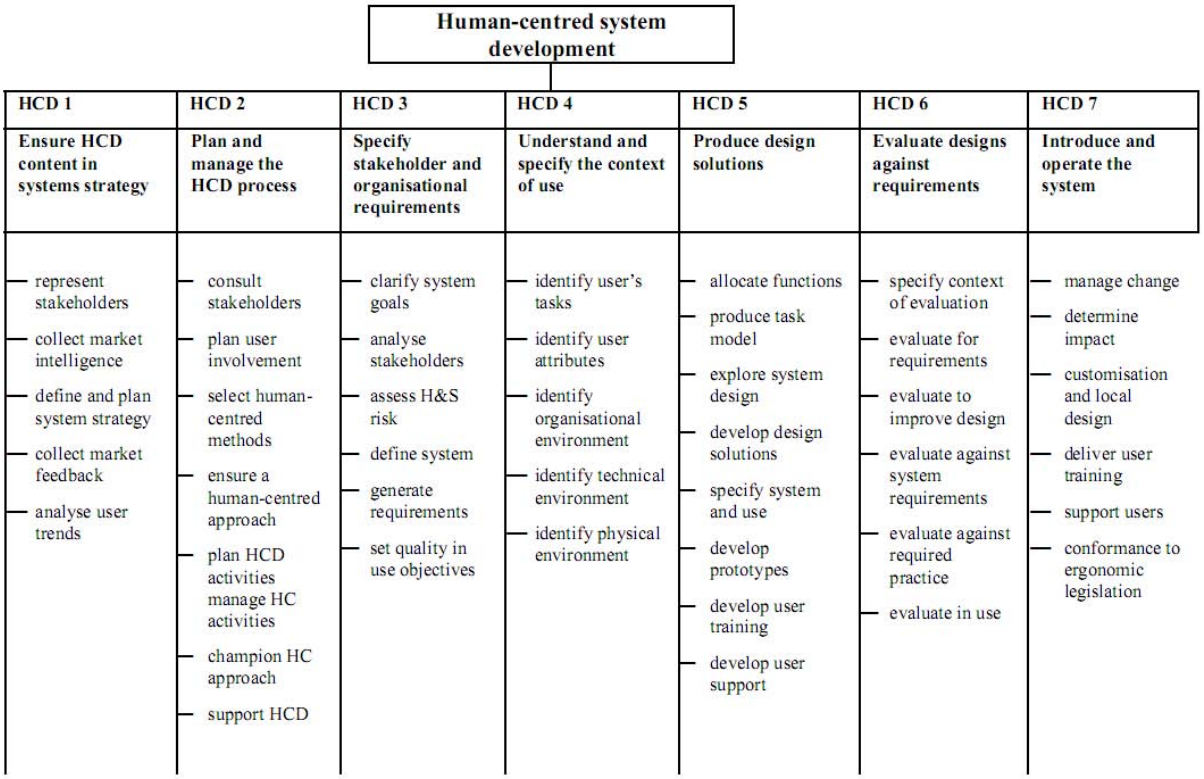


Figure 21. Phases de développement ISO TR 18529

La Figure 21 présente les différentes étapes nécessaires dans un processus centré utilisateur. On y remarque que le développement et la mise en place de la formation de l'utilisateur en fait partie. Le standard ne met pas en valeur un processus en particulier, ni une séquence d'étapes particulière à suivre.

1.2.7 Les méthodes et approches basées sur la génération et l'exploitation de modèles

Plusieurs méthodes basées sur la modélisation des données ou/et du traitement des données ont été proposées pour développer des systèmes interactifs vérifiant des propriétés d'utilisabilité.

Les méthodes Diane+ (Tarby & Barthet, 1996) et MUSE* (Lim, Long, & Silcock, 1992) sont des exemples de méthodes de conception et développement associant une approche ergonomique et une approche de génie logiciel. Ces méthodes sont présentées et comparées dans les travaux de (Palanque, Long, Tarby, Barthet, & Lim, 1994) et spécifient précisément des étapes de conception et développement avec des notations associées pour modéliser les données du système et le traitement des données du système.

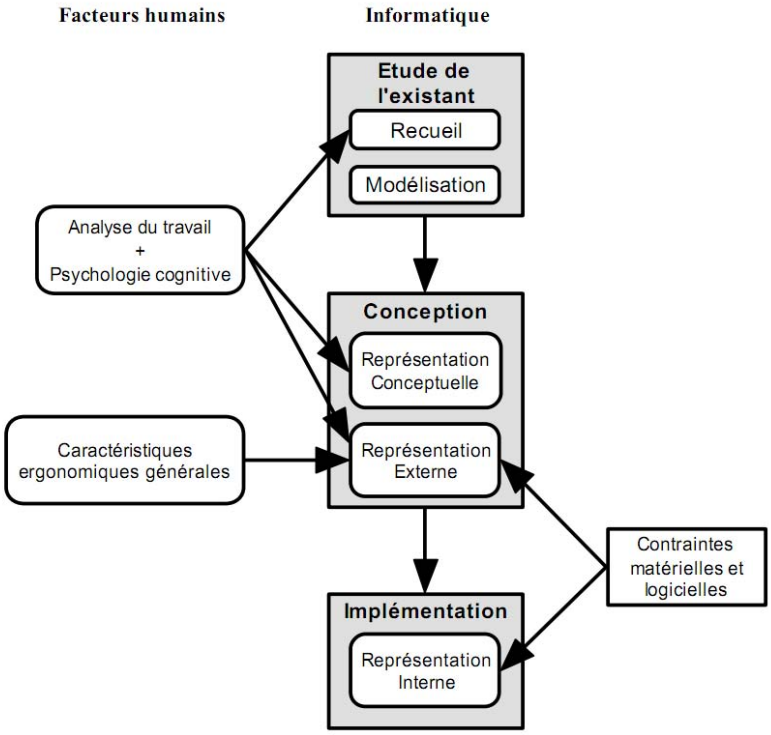


Figure 22. Processus de développement Diane+

Les Figure 22 et Figure 23 sont issues des travaux de (Palanque, Long, Tarby, Barthet, & Lim, 1994) et décrivent les processus de développement des méthodes Diane+ et MUSE*. Ces deux méthodes prennent appui sur des méthodes et processus de conception et développement d'applications logicielles et y ajoutent des étapes liées à l'ergonomie. La méthode Diane+ prend complète la méthode de conception MERISE (Tardieu, Rochfeld, & Colletti, 1986).

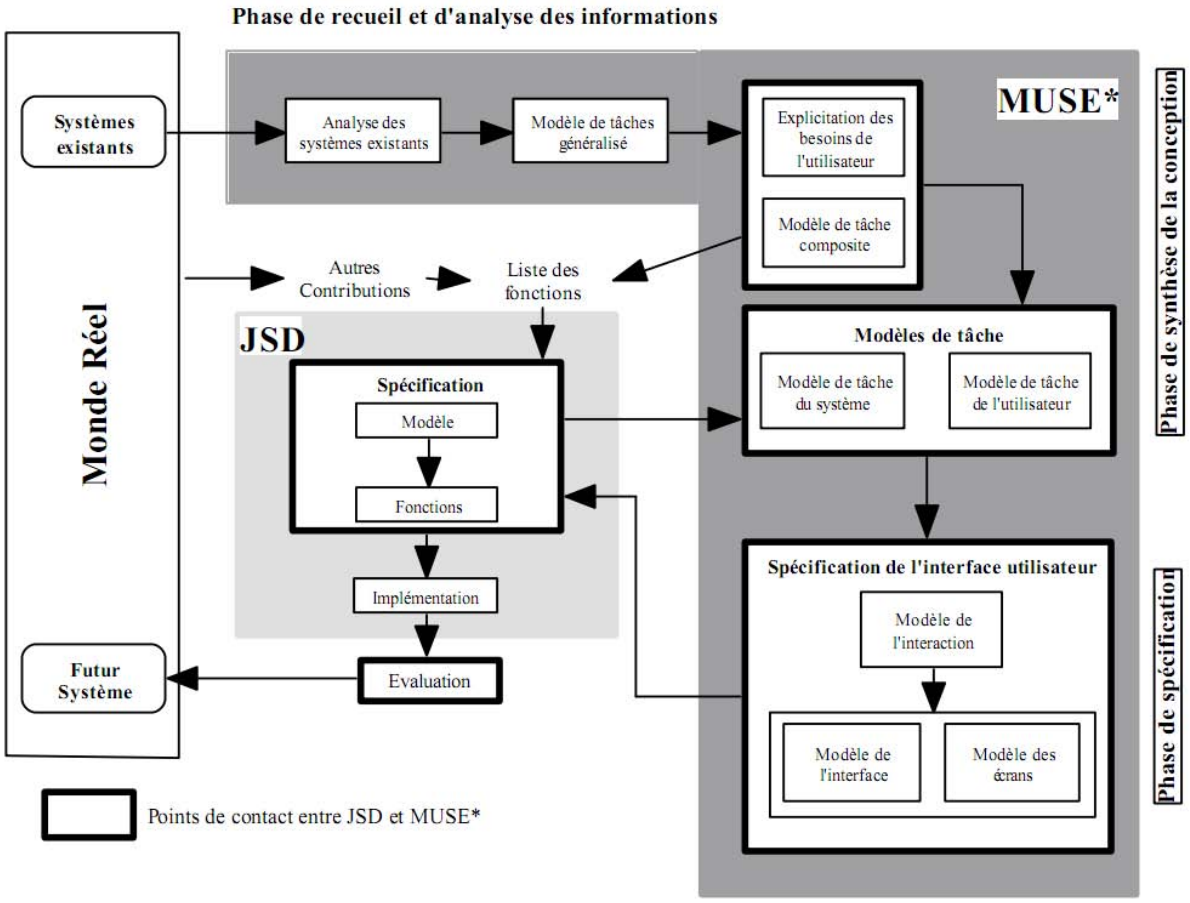


Figure 23. Processus de développement MUSE*

La méthode MUSE* prend appui sur le processus de développement Jackson System Development (JSD) (Jackson, 1983).

Ces approches fournissent des étapes détaillées pour la conception mais ne prévoient pas l'exploitation synergique des modèles et la vérification des uns par rapport aux autres.

Le cadre conceptuel CAMELEON (Calvary, et al., 2002) (Calvary, et al., 2003) propose un référentiel pour classifier et développer des interfaces homme machine adaptables à différentes configurations matérielles et à différents contextes d'utilisation en préservant les propriétés d'utilisabilité.

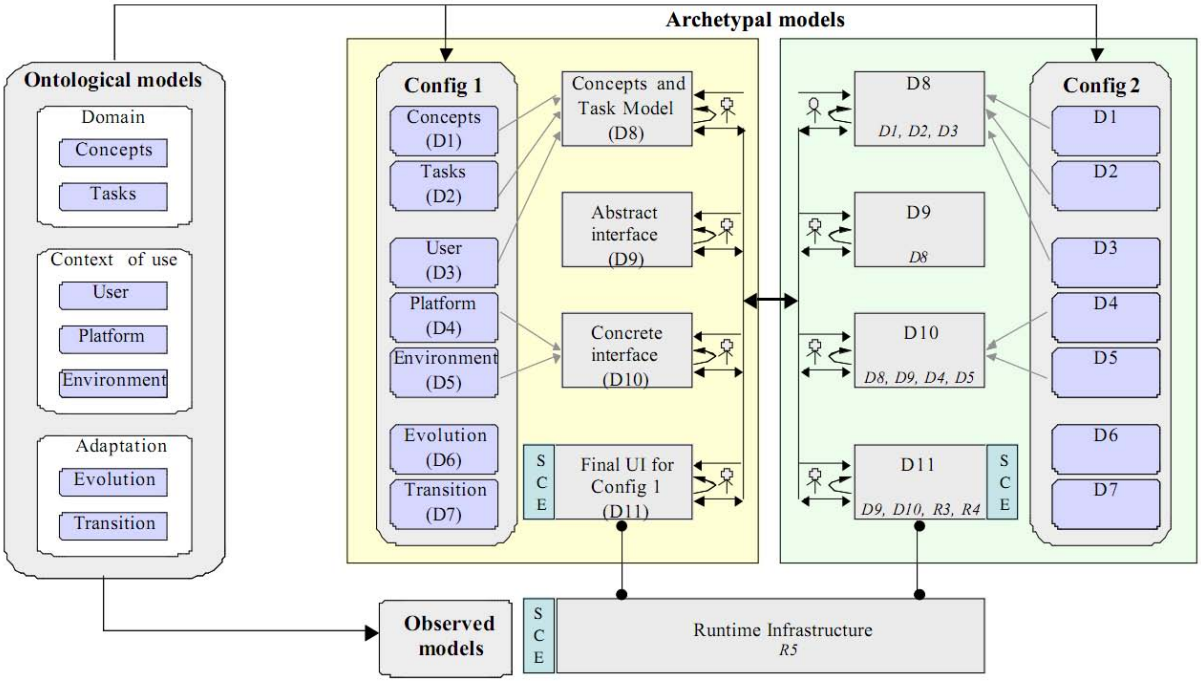


Figure 24. Cadre conceptuel CAMELEON

La Figure 24 présente le référentiel CAMELEON :

- « Ontological models » (bloc sur partie gauche) : un ensemble de méta-modèles liés au domaine, au contexte d'utilisation ainsi qu'aux opérations d'adaptation à effectuer en cas de changement de contexte
- Processus d'implémentation logicielle pour chaque configuration matérielle (deux blocs de la partie supérieure droite) : ce processus prend en entrée les modèles du contexte d'utilisation concerné. Dans cette figure, l'exemple est pris de deux versions logicielles à développer pour deux configurations matérielles. Pour chacune des configurations, le démarrage s'effectue par l'étape d'instanciation des modèles conceptuels (« Concept and Task Model », D8) puis chaque étape suivante concrétise un peu plus l'interface utilisateur jusqu'à la version exécutable (« Final UI for Config x », D11)
- « Runtime Infrastructure » (bloc sur la partie inférieure) : une infrastructure sous-jacente s'exécutant en même temps que la version exécutable de l'application et permettant les adaptations en temps-réel sur un changement de contexte.

1.3 Approches, processus et recommandations pour le développement de systèmes critiques

Le développement de systèmes critiques requiert des étapes particulières supplémentaires afin d'assurer les propriétés de sûreté et fiabilité. Par rapport aux approches et processus génériques présentés dans la section 1.1, des étapes détaillées de vérification, validation et certification sont nécessaires (Storey, 1996).

1.3.1 DO 178 B

Le processus de développement définit dans le standard DO-178B (European Organisation for Civil Aviation Equipment, 1992) est présenté en Figure 25.

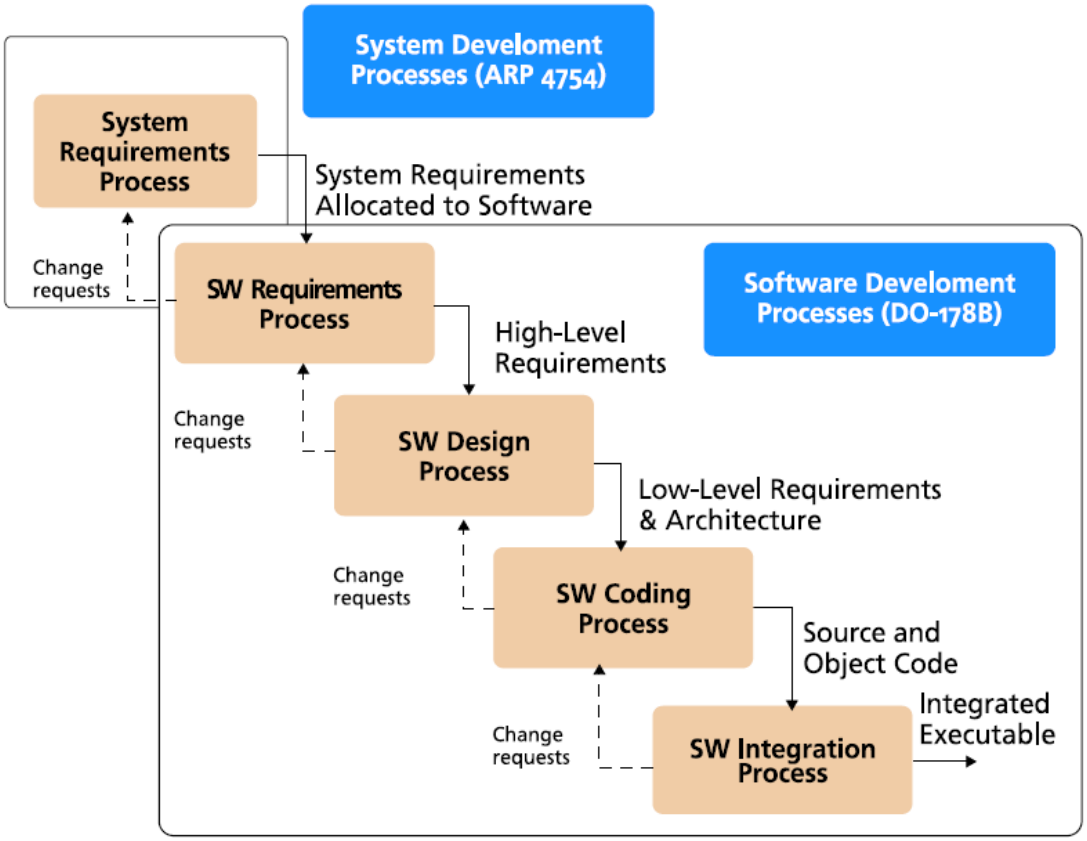


Figure 25. Processus de développement DO-178B

Ce processus est composé de quatre phases :

- La phase d'analyse des exigences (« SW Requirements process ») qui produit les exigences logicielles de haut niveau (HLR)
- La phase de design (« SW Design process ») qui produit les exigences logicielles de bas niveau (LLR) ainsi que l'architecture du logiciel à partir des exigences de haut niveau(HLR)
- La phase de codage (« SW Coding process ») qui produit le code source et le code de l'application
- La phase d'intégration (Integration Process) qui produit le code exécutable et lie le logiciel au système intégré.

Les exigences logicielles de haut niveau (HLR) sont produites directement par l'analyse des exigences du système et l'architecture du système. Ils comportent des spécifications d'exigences fonctionnelles et opérationnelles, les contraintes de mémoire, interfaces matérielles et logicielles, les détections de panne et les exigences de sécurité. Les HLR sont ensuite développées au cours du processus de conception de logiciels, produisant ainsi l'architecture logicielle et les exigences de bas niveau (LLR). Il s'agit notamment des descriptions des entrées / sorties, les données et les flux de contrôle, la limitation des ressources, la planification et des mécanismes de communication, ainsi que des composants logiciels.

Grâce au processus de codage, les exigences bas-niveau sont mises en œuvre sous forme de code source. Le code source est compilé et lié par le processus d'intégration en un code exécutable lié à l'environnement cible. À toutes les étapes, la traçabilité est requise: entre les exigences du système et HLR, HLR et LLR, entre LLR et le code, et aussi entre les essais et les exigences.

De plus, le standard DO-178B fournit des lignes directives pour les processus de vérification du logiciel, de gestion de la configuration du logiciel et d'assurance qualité du logiciel.

1.3.2 ESA PSS-05

Le processus de développement PSS-05 (European Space Agency, 1994), nommée « Software Engineering Standard », est utilisé pour différents projets de l'ESA (European Space Agency). Ce standard propose des lignes directives pour le développement et la maintenance d'applications logicielles.

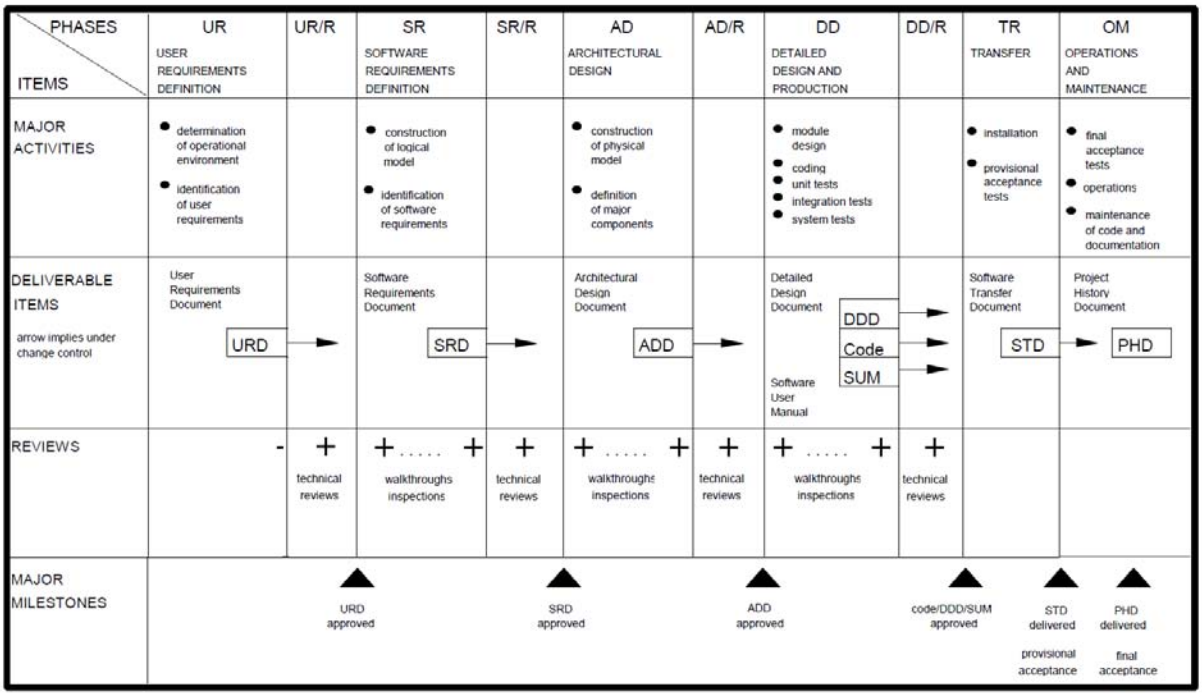


Figure 26. Processus de développement ESA PSS-05

Le processus de développement issu de la norme PSS-05 (European Space Agency, 1994) (présenté en Figure 25) de l'ESA (European Space Agency) est composé de six phases :

- La définition des exigences utilisateurs (UR) : La phase UR est la «phase de définition du problème» du logiciel. Le champ d'application du système et les besoins des utilisateurs doivent être précisées. Cela peut être réalisé via des interviews ou enquêtes, ou par la création de prototypes. Les besoins spécifiques des utilisateurs doivent être identifiés et consignés dans le document des besoins des utilisateurs (URD).
- La définition des exigences logicielles (SR) : La phase de SR est la phase d'«analyse» du logiciel. Une partie essentielle de l'activité d'analyse est la construction d'un «modèle» décrivant ce que le logiciel doit faire, et non pas «comment» le faire. La construction de prototypes permet de clarifier les exigences logicielles. Le livrable principal de cette phase est le document sur les exigences du logiciel (SRD).
- La conception de l'architecture (AD) : Le but de la phase d'AD est de définir la structure du logiciel. Le modèle construit dans la phase de SR est le point de départ. Ce modèle est transformé dans la conception architecturale en affectant des fonctions aux composants logiciels et en définissant le contrôle et les flux de données entre eux. Cette phase peut comporter plusieurs itérations de conception. Les parties complexes ou critiques doivent être identifiées dans cette conception.
    - La conception détaillée et la production (DD) : Le but de la phase de DD est de détailler la conception du logiciel, le coder, le documenter et de le tester. Le document de conception détaillée (DDD) et le Software User Manual (SUM) sont produits en même temps que le code et les tests. Initialement, le DDD et SUM contiennent les sections correspondant aux plus hauts niveaux du système. Au fur et à mesure de la conception, les sous-sections

relatives à des niveaux inférieurs sont ajoutées. À la fin de la phase, les documents sont complétés et, avec le code, constituent les éléments livrables de cette phase.
- Le transfert (TR) : Le but de cette phase est d'établir que le logiciel répond aux exigences fixées dans l'URD. Cela se fait par l'installation du logiciel et des essais d'acceptation. Lorsque le logiciel a été démontré capable de fournir les capacités requises, le logiciel peut être accepté provisoirement et la phase d'opérations peut commencer.
- La phase d'opérations et maintenance (OM) : Une fois le logiciel mis en service, il doit être soigneusement contrôlé afin de vérifier s'il satisfait à toutes les exigences définies dans l'URD. Lorsque le logiciel a passé tous les tests d'acceptation, il peut être finalement validé. L'historique du projet de document (PHD) résume les informations importantes de gestion accumulées au cours du projet.

Après acceptation définitive, le logiciel peut être modifié pour corriger les erreurs détectées lors des étapes antérieures ou parce que de nouvelles exigences se posent.

1.3.3 Spécificité des systèmes critiques : la certification

Les responsables de la conception et du développement d'un système critique doivent démontrer que le système est suffisamment sûr pour que les autorités de certification émettent l'autorisation de mise en circulation du système.

Storey (Storey, 1996) indique trois aspects techniques importants pour accéder aux phases de certification :

- La démonstration que les risques majeurs ont été identifiés et pris en compte.
- La preuve que le système est conforme aux standards en vigueur.
- L'argumentation montrant que le système est sûr et qu'il le restera au cours de son cycle de vie.

Les étapes-clés des approches de développement des systèmes critiques fournissant un support pour les phases de certification sont : la vérification, la validation et le test.

La vérification sert à montrer que les ressources produites par une phase du cycle de développement sont conformes aux exigences soumises en entrée de cette phase du cycle.

La validation sert à montrer que les exigences émises au début du cycle de vie du cycle de développement sont appropriées et cohérent avec les besoins du client et/ou de l'utilisateur.

Le test est une manière de vérifier et valider tout ou partie du système (les revues de spécifications et de code sont d'autres manières). Dans le cas d'un système interactif, il est impossible de couvrir toutes les possibilités de combinaisons d'interactions avec le test. Ainsi, la **modélisation formelle** est une activité de plus en plus utilisée et faisant l'objet de nombreuses recherches (Miller, Tribble, Whalen, & Heimdal, 2006).

1.4 Discussion sur l'applicabilité de ces approches aux systèmes interactifs critiques

Lors de l'établissement des moyens de développement d'un système interactif critique, les propriétés de fiabilité, d'utilisabilité et d'opérabilité du système final doivent être prises en compte par l'approche. Le programme de formation associé doit être en phase avec le système produit. De plus, un nombre important de ressources associées au développement doivent être gérées et leurs modifications traçables (comme décrit dans la section 3 du chapitre 1).

La Table 2 classe les approches présentées précédemment selon cinq critères :

- Fournit un ensemble d'étapes ordonné partant des phases amont d'analyse des besoins jusqu'aux phases d'implémentation et de livraison. Leur description est nécessaire pour la traçabilité des nombreuses ressources associées produites lors du développement (passage à l'échelle, industrialisation).
- Autorise les itérations pendant la conception et le développement. C'est une condition nécessaire (mais non suffisante) pour fournit un support au développement d'un système utilisable.
- Est basée sur l'utilisation de modèles pour représenter le système et/ou son utilisation.
- Fournit un support au développement d'un système utilisable. L'approche doit permettre d'appliquer les méthodes et techniques de l'approche centrée utilisateur.
- Fournit un support au développement d'un système fiable. L'approche doit permettre d'appliquer les méthodes et techniques de traçabilité, vérification et validation.
- Intègre le développement du programme de formation. L'approche doit permettre de concevoir un programme de formation en adéquation complète avec le système.

Comme décrit à l'issue de la section 2 du chapitre 1, la propriété d'opérabilité présente des caractéristiques communes avec les propriétés d'utilisabilité et de fiabilité, ainsi la Table 2 ne contient pas de colonne dédiée à ce critère.

Table 2. Tableau comparatif des approches de développement existantes

Approches de développement		Ensemble complet et ordonné d'étapes de l'analyse des besoins à l'implémentation	Itérations autorisées dans l'approche	Basée sur les modèles	Support à l'utilisabilité	Support à la fiabilité	Support à la conception d'un plan de formation
Approches génériques	Cycle de développement en cascade (Royce, 1970)	Oui	Non	Non	Non	Non	Non
	Cycle de développement en V (McDermid & Ripken, 1983)	Oui	Non	Non	Non	Non	Non
Approches et processus pour les systèmes interactifs	Cycle de développement en spirale (Boehm, 1986)	Oui	Oui	Non	Non	Non	Non
	RUP (Kruchten, 2004)	Oui	Oui	Non	Non	Non	Non
	XP (Beck, 1999)	Oui	Oui	Non	Non	Non	Non
	SCRUM (Shwaber & Beedle, 2002)	Oui	Oui	Non	Non	Non	Non
	Cycle de développement en étoile (Hartson & Hix, 1989)	Non	Oui	Non	Oui	Non	Non
	Processus de développement en couches (Curtis & Hefley, 1994)	Non	Oui	Non	Oui	Non	Non
	Processus de développement en cercle (Collins, 1995)	Non	Oui	Non	Oui	Non	Non
	Processus de développement itératif cyclique (Rauterberg, 1992)	Non	Oui	Non	Oui	Non	Non
	UCD (Gulliksen & Goransson, 2003)	Oui	Oui	Non	Oui	Non	Non
	IDM	Oui	Non	Oui	Non	Non	Non
	CAMELEON	Non	Non explicité	Oui	Oui	Non	Non
	Muse*	Non	Non explicité	Oui	Oui	Non	Non
	Diane+	Non	Non explicité	Oui	Oui	Non	Non
Processus standards pour les systèmes critiques	DO 178B	Oui	Non	Non	Non	Oui	Non
	ESA PSS 05	Oui	Non	Non	Non	Oui	Non

Les standard ISO/IEC – IEEE 12207 et ISO TR 18529 ne sont pas représentés dans le tableau car ils n'explicitent pas un moyen de développer un système mais des recommandations pour l'établissement des moyens de développement. Leur description dans les sections précédente permet néanmoins de montrer que la communauté internationale scientifique et industrielle s'associe pour établir des recommandations sur les processus de développement dans le but qu'ils puissent fournir un support à des systèmes nécessitant de nouvelles propriétés.

Ce tableau de synthèse permet de mettre en évidence qu'aucune approche ne fournit de support de à l'ensemble des critères requis. D'une part, le développement du programme de formation associé à un système est conçu de manière totalement asynchrone et extérieure au développement du système. D'autre part, les approches fournissent un support soit au critère d'utilisabilité, soit au critère de fiabilité (excepté les deux premières et plus anciennes approches).

Afin de fournir un support au développement d'un système utilisable et fiable, dans les contextes où le nombre de ressources associées au système est considérable et croit avec les capacités et projets technologiques, des moyens spécifiques sont nécessaires :

- L'utilisation d'approches systématiques de développement. Nous proposons une approche systématique de développement intégrant le développement du programme de formation associé (présentée au chapitre 5).
- L'utilisation de modèles du système et des tâches utilisateur. Nous proposons de mettre en œuvre l'approche de développement proposée dans le cadre de cette thèse avec des techniques de modélisation du comportement du système et des tâches utilisateur (présentée au chapitre 6).

2 Notations et outils de modélisation pour les systèmes interactifs et les systèmes interactifs critiques

Cette section présente les notations et outils de modélisation pour la conception et le développement de systèmes interactifs critiques. L'activité de modélisation est utilisée dans plusieurs approches de développement comme exposé dans la section précédente. Elle permet de mettre en valeur certaines informations concernant le système (structure, comportement, flux de données, états, architecture…). Elle permet aussi de mettre en valeur certaines informations concernant les utilisateurs (activités, tâches, flux de travail,…). La modélisation permet aussi de filtrer des informations moins importantes et les modèles permettent d'aider à raisonner quand le sujet étudié est compliqué, c'est pourquoi cette technique fournit un support à la conception et au développement de systèmes interactifs complexes ou possédant un grand nombre de fonctionnalités.

Dans la sous-section 2.1, nous étudions les techniques de modélisation du comportement du système et choisissons une technique de modélisation formelle. En effet, dans les domaines de l'ingénierie logicielle et des systèmes interactifs, les modèles d'architectures fonctionnelles et les modèles de séquences sont largement utilisés, mais ce type de modèles ne permet pas de vérifier le comportement du système. C'est pourquoi, dans le domaine des systèmes critiques, les modèles formels de comportement sont particulièrement utilisés pour prédire le comportement du système dans différents contextes et cas d'utilisation afin de fournir un support à la vérification et à la validation du système (Storey, 1996). Ainsi, la sous-section 2.1 propose de sélectionner une notation outillée pour modéliser le comportement du système afin de fournir un support au développement de systèmes interactifs critiques fiables.

Dans la section 2.2, nous étudions les techniques de modélisation des tâches. En effet, dans le domaine des systèmes interactifs, l'analyse des tâches utilisateur est une étape clé de la conception centrée utilisateur (Diaper & Stanton, 2004). Les modèles de tâches permettent de consigner et structurer une partie de cette analyse et sont aussi souvent utilisés. Dans le cadre de systèmes interactifs critiques où le nombre de tâches utilisateur est très important, les modèles de tâches sont indispensables pour pouvoir exploiter cette analyse de tâches. La sous-section 2.2 détermine quelle notation outillée est la plus à même de fournir un support au développement d'un système interactif critique.

Dans les deux cas, pour les techniques de modélisation du comportement du système et pour les techniques de modélisation des tâches utilisateur, nous précisons notation « outillée ». En effet, dans chaque cas, au vu de la complexité du système et du nombre d'information à gérer, un outil logiciel de modélisation associé à la notation est nécessaire pour prouver la faisabilité de l'approche de développement.

Dans la suite du mémoire, la distinction est faite entre une notation et un modèle. Le modèle est le résultat de l'activité de modélisation avec une notation. Lorsqu'une notation peut être définie mathématiquement, elle est appelée formalisme. Lorsqu'un modèle est créé à partir d'un formalisme, il est formel. Les modèles formels fournissent un support à l'évaluation des propriétés d'un système.

2.1 Notations et outils de modélisation du comportement des systèmes interactifs critiques

Plusieurs notations ont été proposées pour la conception de systèmes informatiques qu'ils soient génériques, interactifs, ou critiques. Ces notations permettent de produire des modèles de différents types selon le but de l'activité de modélisation. Cependant, dans le domaine des systèmes interactifs critiques et dans le domaine des systèmes critiques de manière générale, les travaux de recherche s'orientent vers une modélisation formelle du système (propriétés, états, comportement).

2.1.1 Besoins et exigences pour la modélisation de systèmes interactifs critiques

Dans le domaine des systèmes interactifs, le besoin de notations formelles a été soulevé il y a plusieurs années. Les travaux de (Palanque & Bastide, 1994), (Johnson C. , 1995) et (Bastide & Palanque, 1995) (Thimbleby, 2007) soulignent leurs importances dans divers domaines d'application en démontrant qu'elles permettent de :

- Vérifier si les propriétés sont satisfaites.
- Prévoir les implications des choix de conception en s'abstrayant des choix de développement.
- Obliger le concepteur à expliciter ses choix.

De plus les travaux de (Dix, 1995) ajoutent aussi qu'elles permettent de :

- Lever les ambiguïtés liées au langage naturel.
- Améliorer la maîtrise du temps de développement.

Dans le domaine des systèmes critiques, les méthodes formelles (modélisation et vérification) font l'objet de nombreux travaux de recherche, comme souligné par les travaux de (Miller, Tribble, Whalen, & Heimdal, 2006).

Les utilisateurs interagissant de manière asynchrone et imprévisible, il est nécessaire de trouver une notation qui permet de modéliser les actions d'entrée et sortie concurrentes de l'utilisateur. Par exemple, les interfaces utilisateur d'un cockpit d'avion sont multimodales. En effet, le pilote et son co-

pilote ont chacun les mêmes périphériques d'entrée/sortie disponibles et le système doit être capable de gérer les actions combinées des pilote et co-pilote.

Les travaux de (Navarre, Palanque, Ladry, & Barboni, 2009) fournissent une présentation détaillée des exigences pour la sélection d'une notation de modélisation du comportement du système. Nous en retenons les points-clés:

- La notation doit être compatible avec l'architecture de gestion d'évènements des systèmes interactifs. En effet, les interactions entre l'utilisateur et le système se matérialisent à travers la création d'un évènement système suite à une action d'entrée de l'utilisateur.
- La notation doit pouvoir décrire de manière complète et non ambigüe tous les états possibles du système et la manière dont les évènements entraînent les changements d'états.
- La notation doit être capable de représenter et gérer les structures de données et le comportement du système de manière intégrée, les systèmes interactifs critiques manipulant de grandes quantités de données qui influencent aussi leur comportement.

Table 3. Pouvoir d'expression des notations de description des interfaces utilisateur extrait de (Navarre, Palanque, Ladry, & Barboni, 2009)

	Expressiveness of UIDL						
	Data Description	State Representation	Event Representation	Time		Concurrent Behaviour	Dynamic Instantiation
				Quantitative	Qualitative		
Coral	Code	N	N	N	Y	N	N
Apogee	Code	N	N	N	N	N	N
Squeak	Code	N	Code	N	Y	Y	N
CSP	Code	Code	Code	N	Y	Y	N
DMI	Code	Code	Code	N	Y	N	N
Subarctic	N	Code	Code	N	Y	N	N
XISL	Code	N	Code	N	Y	Y	N
Usixml	Code	N	Code	N	Y	Y	N
GWUIMS	Y	N	Code	N	Y	Y	N
Tatsukawa	Y	N	Y	N	Y	Y	N
Marigold	Y	Y	Y	N	Y	Y	N
Wizz ed	Y	N	Y	N	Y	Y	N
ICON	Y	N	Y	N	Y	Y	N
Swingstates	N	Code	Code	N	Y	N	N
Hierarchical	Code	Code	Code	N	Y	Y	N
3-State Model	N	Y	Y	N	Y	N	N
GTN	N	Y	Y	N	Y	N	N
HephaisTK	Code	Y	Y	N	Y	Y	N
IOG	Y	Y	Y	N	Y	N	N
Shadow	Y	Y	Y	N	Y	Y	N
NiMMiT	Y	Y	Y	N	Y	Y	N
Hinckley	N	Y	Y	N	Y	Y	N
OSU	Y	Y	Y	N	Y	Y	N
MIML	Y	Y	Y	N	Y	Y	N
ICO	Y	Y	Y	Y	Y	Y	Y

Les Table 3 et Table 4 sont extraites des travaux de (Navarre, Palanque, Ladry, & Barboni, 2009) et permettent de montrer que seule la notation outillée ICO répond aux exigences pour la conception et le développement d'un système interactif critique utilisable, fiable et opérable. Nous l'utilisons pour

mettre en œuvre l'approche proposée dans cette thèse (son emploi dans ce cadre est développé dans les chapitres 6 et 7). La section suivante (section 2.1.2) présente cette notation.

Table 4. Applicabilité de la notation et de son outil logiciel pour des systèmes interactifs possédant un grand nombre de fonctionnalités extrait de (Navarre, Palanque, Ladry, & Barboni, 2009)

	Scalability			Tool Support		
	Toy Example	Case Study	Real Size	Snapshot	Demo	Downloadable
Coral	Y	N	N	N	N	N
Apogee	Y	Y	N	Y	N	N
Squeak	Y	N	N	N	N	N
CSP	Y	Y	N	N	N	N
DMI	Y	Y	N	Y	N	N
Subarctic	Y	Y	Y	Y	Y	Y
XISL	Y	Y	N	Y	N	Y
Usixml	Y	Y	Y	Y	Y	Y
GWUIMS	Y	Y	N	Y	N	N
Tatsukawa	Y	N	N	N	N	N
Marigold	Y	Y	N	Y	N	N
Wizz ed	Y	Y	N	Y	Y	N
ICON	Y	Y	N	Y	Y	Y
Swingstates	Y	N	N	Y	Y	Y
Hierarchical	Y	Y	Y	Y	N	Y
3-State Model	Y	N	N	N	N	N
GTN	N	N	N	N	N	N
HephaisTK	Y	Y	N	Y	Y	N
IOG	Y	Y	N	Y	Y	N
Shadow	Y	Y	N	Y	N	N
NiMMiT	Y	Y	N	Y	Y	N
Hinckley	Y	N	N	N	N	N
OSU	Y	N	N	Y	N	N
MIML	Y	Y	N	Y	N	Y
ICO	Y	Y	Y	Y	Y	Y

2.1.2 ICO : notation outillée de modélisation des systèmes interactifs critiques

Navarre et al. (Navarre, Palanque, Ladry, & Barboni, 2009) proposent la notation outillée ICO (Interactive Cooperative Objects) pour concevoir, spécifier, prototyper et valider tout ou partie d'un système interactif, qu'il soit de type WIMP (Window Icon Menu Pointing) ou post-WIMP (interactions multimodales). Cette notation est basée sur les réseau de Petri (Petri, 1962).

Bref rappel sur les réseaux de Petri

Les réseaux de Petri sont composés de trois types d'éléments (exemple Figure 27) :

- Les **places** (représentées par des ellipses sur la Figure 27) permettent de représenter les variables d'état du système selon qu'elles contiennent ou non des **jetons**.
- Les **transitions** (représentées par des rectangles sur la Figure 27) permettent de représenter les actions et changements d'états.
- Les **arcs** (représentés par des flèches sur la Figure 27) permettent de représenter les flux des jetons dans le réseau de Petri. Les arcs peuvent être pondérés.

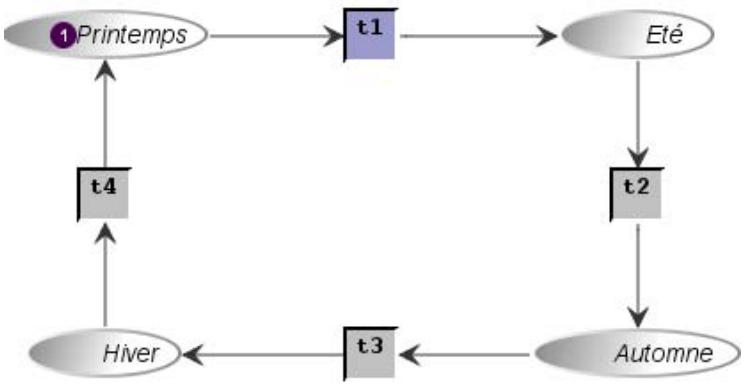


Figure 27. Exemple de réseau de Petri

Un état du système modélisé avec cette notation est représenté par une distribution de jetons parmi l'ensemble des places. Cette distribution est appelée **marquage**.

Le comportement dynamique des réseaux de Petri est exprimé par deux règles :

- La règle de possibilité de franchissement : une transition est franchissable si les places en amont contiennent le au moins autant de jetons que la valeur du poids des arcs auxquels elles sont liées.
- La règle de franchissement : lorsqu'une transition est franchie, les jetons sont enlevés des places d'entrée et déposés dans les places de sortie (selon le nombre défini par le poids des arcs).

Cette notation est formelle car elle a une équivalence algébrique et fournit un support à la spécification de systèmes dont le nombre d'états internes est infini de par son pouvoir d'expression et elle fournit un support à la vérification (grâce au modèle mathématique sous-jacent).

La notation ICO répond aux exigences citées précédemment et permet la spécification complète et non ambigüe de tout ou partie du système :

- Elle utilise le formalisme des réseaux de Petri et convient donc pour spécifier le comportement et les états internes des systèmes évènementiels et les interactions concurrentes entre l'utilisateur et le système (Bastide & Palanque, 2001). Ce formalisme inclut aussi deux extensions proposées par (Lakos & Christensen, 1994): les arcs de test et les arcs inhibiteurs.
- Elle est basée sur le paradigme des réseaux de Petri à objets (Lakos, 1991), qui fournit un support au traitement de structures de données complexes (places et jetons typés, transition avec actions et préconditions, noms de variables sur les arcs).
- Elle s'appuie sur la notation à base d'Objets Coopératifs (Bastide & Palanque, 2001). Cette notation ajoute des capacités de description d'interfaces logicielles et de communication aux réseaux de Petri orientés objets. Les Objets Coopératifs réagissent aux évènements externes en fonction de leur état interne et produisent eux-mêmes des évènements. Ils sont aussi capables d'offrir des services à d'autres Objets Coopératifs (à la manière d'appels de méthodes bloquants ou non bloquants). Ce type de communication utilise le patron de conception Client-Serveur (Ramamoorthy & Ho, 1980) et est décrit par le formalisme des Service Input Port et Service Output Port (Bastide & Palanque, 2001).
- Elle définit un objet comme l'ensemble de quatre éléments : un Objet Coopératif, un moyen de présentation à l'utilisateur, une fonction d'activation et une fonction de rendu.
    - L'Objet Coopératif permet de décrire le comportement de l'objet

- Le moyen de présentation à l'utilisateur est définit par l'apparence externe de l'objet (une fenêtre applicative, un composant graphique, ou de manière plus générale une information perceptive visuelle, sonore,…)
- La fonction d'activation associe un évènement d'entrée (issue de l'action de l'utilisateur sur un périphérique d'entrée) à un service de l'Objet Coopératif correspondant.
- La fonction de rendu associe un changement d'état interne de l'Objet Coopératif à une sortie possible du moyen de présentation.

Ce formalisme a été utilisé dans plusieurs domaines d'application (Navarre, Palanque, Ladry, & Barboni, 2009) et il présente des avantages supplémentaires:

- La spécification prend en compte les aspects d'entrée de l'interaction (la manière dont les actions de l'utilisateur sont répercutées sur l'état interne de l'application, et quelles actions sont disponibles pour l'utilisateur à un instant donné). Elle prend aussi en compte les aspects de sortie de l'interaction (selon l'état vers lequel le système a changé, il présente l'information conséquente à l'utilisateur). L'entrée est évènementielle et la sortie basée sur les états.
- Grâce à son outil logiciel associé, PetShop, la spécification est entièrement exécutable et modifiable à l'exécution, ce qui fournit un support au prototypage et au test à tout ou partie du système avant qu'il ne soit déployé (Palanque, Ladry J-F., Navarre, & Barboni, 2009). Et un grand nombre de modèles peuvent être exécutés simultanément. A ce jour, dans certaines études de cas effectuées avec PetShop[3] (Barboni, Navarre, Palanque, & Bazalgette, 2006), environ trois cent composants logiciels utilisés dans les applications de cockpit d'avions civils et définis par le standard aéronautique ARINC 661 ont été modélisés et testés en exécution, comme par exemple dans (Barboni, Conversy, Navarre, & Palanque, 2006).

2.2 Notations et outils de modélisation des tâches utilisateur

La modélisation des tâches utilisateur consiste à décrire les activités de l'utilisateur dans un contexte donné pour accomplir un ou plusieurs buts. L'activité de modélisation des tâches est une approche systématique et complémentaire à l'analyse de tâches. Elle permet de structurer et consigner les informations issues de l'analyse de tâches.

La pratique de l'analyse de tâches est répandue depuis de nombreuses années dans divers domaines (industrie, sciences de l'éducation, interactions homme-machine) et avec différents objectifs (organisation du travail, formation, représentation de l'activité mentale). Les travaux de (Hollnagel, 2006) et (Anett, 2004) dressent un récapitulatif de la pratique de l'analyse de tâche et des différentes approches de modélisation jusqu'à aujourd'hui.

Dans le cas de systèmes possédant un grand nombre de fonctionnalités et permettant de gérer des opérations et tâches complexes, le nombre de données à collecter amène le besoin de notations et/ou outils pour structurer les tâches et les modèles. Les caractéristiques de ces notations et outils vont ainsi dépendre : du domaine d'application, du but de l'activité de modélisation et de l'activité à modéliser.

La section suivante décrit les critères de sélection habituellement utilisés dans le domaine des systèmes interactifs ainsi que les critères que nous mettons en valeur pour la conception d'un système interactif critique.

[3] http://www.irit.fr/recherches/ICS/softwares/petshop/

2.2.1 Besoins et exigences pour la modélisation des tâches des utilisateurs d'un système interactif critique

Plusieurs travaux de recherche rassemblent des critères pour la sélection d'une notation et d'un outil de modélisation de tâches.

Les travaux de (Balbo, Ozkan, & Paris, 2004) proposent une taxonomie de choix d'une notation basée sur :

- Le but de l'utilisation de la notation.
- L'utilisabilité pour la modélisation.
- L'utilisabilité pour la communication.
- L'adaptabilité.
- La couverture (en termes de : variété de sélection des types de tâches, décomposition hiérarchique, décomposition temporelle, description de tâches non standards ou erreurs).
- L'extensibilité de la notation.

Les travaux de (Limbourg & Vanderdonckt, 2004) comparent un échantillon représentatif parmi les notations de modélisations provenant de plusieurs disciplines (psychologie cognitive, planification des tâches, ingénierie logicielle et ethnographie) afin de mettre en valeur leurs avantages et inconvénients. Les points de comparaisons utilisés sont liés à la syntaxe et à la sémantique de la notation :

- Critères syntaxiques : relation entre une tâche et le but de haut niveau, moyen de mise en correspondance entre les tâches de haut niveau et les tâches de plus bas niveau, types des d'éléments de plus bas niveau (non décomposables), niveau dans le modèle correspondant à la réalisation d'une actions
- Critères sémantiques : discipline d'origine, aspect formel, capacité de description des aspects collaboratifs, adaptabilité aux variations de contexte d'utilisation, capacité de description des aspects cognitifs, portée des opérateurs temporels (le cas échéant, comment sont-ils connectés au tâches), manipulation des objets.

Dans une étude empirique sur le pouvoir d'expression des notations outillées de modélisation de tâches, (Caffiau, Scapin, Girard, Baron, & Jambon, 2010) utilisent des critères de couverture en termes de description :

- Des informations et propriétés concernant les tâches (nom, but, exécutant, importance, identifiant, plateforme, fréquence, lieu, support matériel, commentaires)
- De la planification temporelle interne à une tâche (durée, interruptible, optionnelle, itération, coopération)
- De la planification temporelle externe à une tâche (ordonnancement temporel)
- Des objets

Pour fournir un support au développement d'un système interactif critique ou d'un système interactif complexe, nous retenons les critères suivants parmi ceux cités précédemment pour la comparaison des notations et outils :

- But de l'utilisation
    - Dans le but est de **concevoir et développer un système interactif critique et son programme de formation associé**, la notation doit fournir un moyen de structurer et conserver la totalité des informations liées aux tâches utilisateur et donc de gérer une quantité importante de données.

  - La notation doit être outillée afin de permettre la simulation des modèles, la production de scénarios associés et les activités d'évaluation.
- Utilisabilité pour la modélisation
  - La notation et son **outil logiciel associé** doivent fournir des **mécanismes de structuration des modèles** de tâches afin de permettre l'édition d'une quantité importante de tâches utilisateur.
  - Afin de permettre la validation des modèles, l'outil logiciel doit fournir une interface de simulation d'instances de modèles de tâches ainsi qu'une interface de navigation entre ces différentes instances.
- Pourvoir d'expression, couverture
  - La notation doit permettre la décomposition hiérarchique de l'activité humaine de buts en sous-buts.
  - La notation doit permettre de décrire les relations temporelles entre les tâches de manière qualitative et quantitative afin de fournir un support à l'évaluation quantitative des performances de l'utilisateur ainsi qu'à l'estimation de la faisabilité d'une tâche en un intervalle de temps donné.
  - La notation doit permettre de différencier finement les types de tâches et en particulier celles de l'utilisateur (interaction, motrice, cognitive,…) pour fournir un support à l'évaluation des performances de l'utilisateur ainsi qu'à l'analyse de la répartition des tâches entre l'utilisateur et le système.
  - La description des éléments manipulés par l'utilisateur doit être le plus proche possible des objets utilisés par le système, et permettre de montrer le passage de l'objet de l'utilisateur vers le système ou du système vers l'utilisateur.
- Adaptabilité
  La notation et l'outil associé pourront être utilisés afin de modéliser les tâches liées à différents domaines, à différents contextes d'utilisation ainsi qu'à des systèmes de caractéristiques différentes.
- Extensibilité
  La notation et l'outil associé pourront facilement être étendus afin de permettre une augmentation nécessaire de son pouvoir d'expression.

2.2.2 Comparaison des notations existantes et discussion

Cette section fournit une vue d'ensemble sur les notations de modélisation de tâches. Les notations de modélisation de tâches ont été à maintes reprises décrites et analysées. Les travaux les plus récents sont ceux de (Limbourg & Vanderdonckt, 2004) et (Caffiau, Scapin, Girard, Baron, & Jambon, 2010). Aussi nous ne les exposons pas en détail dans ce mémoire. Dans un premier temps, nous citons les notations les plus représentatives et mettons en valeur leurs caractéristiques. Puis un tableau de synthèse expose une vue d'ensemble sur les modèles et des propriétés qu'ils remplissent ou non.

HTA (Hierarchical Task Analysis) (Annett & Duncan, 1967) est la notation la plus ancienne. Initialement textuelle, (Shepherd, 1985) en propose une version graphique permettant de mieux visualiser la décomposition hiérarchique d'un but en sous-tâches. La numérotation des buts et sous-buts permets d'y ajouter des informations de séquence.

GOMS (Goals, Operators, Methods and Selection rules) (Card, Newell, & Moran, 1983) est une notation textuelle venant du domaine de la psychologie cognitive et a pour but de créer des modèles permettant de prédire les performances des utilisateurs opérant un système. Cette notation et à l'origine d'une famille de notation (Baumeister, John, & Byrne, 2000) comprenant Keystroke-Level

Model (KLM), Natural GOMS Language (NGOMSL) et Cognitice-Perceptual-Motor GOMS (CPM-GOMS). GOMS ne permet pas d'exprimer les contraintes temporelles entre les différentes tâches.

MAD (Méthode Analytique de Description des tâches) (Scapin & Pierret-Golbreich, 1989) est une notation graphique permettant de décrire les relations hiérarchiques et temporelles entre les tâches. Dans les versions les plus récentes (Caffiau, Scapin, Girard, Baron, & Jambon, 2010), elle permet aussi de décrire les objets et conditions liées aux tâches.

TKS (Task Knowledge Structure) (Johnson & Johnson, 1989) est une notation provenant du domaine de la psychologie cognitive. Elle permet de modéliser le savoir requis pour effectuer une tâche. Elle se présente comme un ensemble structuré d'objets, d'actions, de procédures permettant d'accomplir un but.

GTA (Groupware Task Analysis) (Van Der Veer, Lenting, & Bergevoet, 1996) est une notation conçue pour modéliser des tâches complexes dans un environnement coopératif. Ainsi, elle introduit la notion de rôle pour la personne en charge d'une tâche.

UAN (User Action Notation) (Hix & Rex Harston, 1993) est une notation textuelle permettant de modéliser les actions de l'utilisateur mais aussi l'interface côté système (retours et état courant). Cette notation fournit principalement un support à la modélisation formelle des tâches interactives.

Diane+ (Tarby & Barthet, 1996) est une méthode de spécification et de conception d'un système interactif et propose une notation de modélisation des tâches utilisateur permettant de décrire les relations hiérarchiques et temporelles entre les tâches.

VTML (Visual Task Modeling Language) (Brown & Leveson, 1998) est une notation graphique conçue dans le but de pouvoir analyser formellement les actions d'un opérateur humain par rapport à des problématiques de sûreté. Elle propose de consigner les actions des opérateurs sous la forme de diagrammes de flux.

CTT (Concur Task Tree) (Paterno, Mancini, & Meniconi, 1997) est une notation graphique permettant d'ordonnancer hiérarchiquement et temporellement les tâches utilisateur. Elle utilise le formalisme LOTOS (ISO, 1989) pour ses opérateurs d'ordonnancement temporel. Elle fournit un support à la description de tâches coopératives et de rôle utilisateur. Les opérateurs d'ordonnancement temporels s'appliquent entre deux tâches seulement et la manière de décomposer les tâches peut donner lieu à des ambiguïtés dans l'interprétation du modèle.

AMBOSS (Giese, Mistrzik, Pfau, Szwillus, & von Detten, 2008) est un environnement de modélisation des tâches utilisateur conçu pour prendre en compte les aspects liés à la sûreté de fonctionnement des systèmes critiques complexes. Les auteurs ne fournissent pas de description précise sur la notation sous-jacente, cependant elle semble utiliser les mêmes concepts de structuration hiérarchique et temporelle que MAD. AMBOSS permet aussi d'associer des informations liées aux erreurs humaines et à la sécurité dans les modèles de tâches (barrières, facteur de risque, criticité de la tâche). AMBOSS prend en compte la notion de rôle.

La Table 5 présente une synthèse des caractéristiques de cet échantillon représentatif de notations de modélisation de tâches utilisateur, afin de montrer les critères généralement remplis par les notations existantes.

Table 5. Comparaison d'un échantillon représentatif de notations de modélisation de tâches utilisateurs

Dénomination		Hierarchical Task Analysis	Goals, Operators, Methods and Selection rules model	GroupWare Task Analysis	Méthode Analytique de Description de tâches	Task Knowledge Structure	Task Analysis for Knowledge Descriptions	User Action Notation	Diane+	Visual Task Modelling Language	Concurrent Task Trees	AMBOSS
Auteurs Année		Annett & Duncan 1967	Card et al. 1983	van der Veer 1996	Scapin et al. 1989	Johnson, 1992	Diaper & Johnson, 1989	Hix & Hartson 1993	Tarby and Barthet 1996	Brown & Leveson 1998	Paternò, 1997	Giese & al., 2008
Acronyme		HTA (Duncan & Annett, 1967)	GOMS (Card, Newell, & Moran, 1983)	GTA (Van Der Veer, Lenting, & Bergevoet, 1996)	MAD (Scapin & Pierret-Golbreich, 1989)	TKS (Johnson & Johnson, 1989)	TAKD	UAN (Hix & Rex Harston, 1993)	Diane+ (Barthet & Tarby, 1996)	(Leveson & Brown, 1998)	CTT (Paterno, Mancini, & Meniconi, 1997)	AMBOSS (Giese, Mistrzik, Pfau, Szwillus, & von Detten, 2008)
NOTATION / LANGAGE	Textuelle	√	√	x	x	√	√	√	x	√	x	x
	Graphique	√	x	√	√	√	√	x	√	√	√	√
	Décomposition Hierarchique	√	√	√	√	√	√	√	√	x	√	√
	Evaluation quantitative des performances ut.	x	√	x	√	x	x	x	x	x	x	x
	Relations temporelles entre les tâches	x	√	√	√	√	x	√	√	√	√	√
	Différents types de tâches	x	√	√	√	√	x	√	√	x	√	x
	Orientée ingénierie des systèmes interactifs	x	x	x	√	x	x	x	√	x	√	√
	Orientée facteurs humains	√	√	√	x	√	√	√	x	√	x	x
	Objets, artefacts	x	x	√	√	√	x	√	√	x	√	√
	Semi-formelle	x	√	x	√	x	x	√	x	x	√	√
	Orientée systèmes interactifs critiques	x	x	x	x	x	x	x	x	√	x	√
	Outil logiciel d'édition et simulation (disponible et maintenu)	x	x	x	K-MADe	x	x	x	x	x	CTTe	AMBOSS

La Table 5 nous permet de montrer que la majeure partie des notations permettent la décomposition hiérarchique des tâches, l'ordonnancement temporel des tâches, la description des objets et artefacts liés aux tâches. Cependant elle met aussi en évidence les points suivants :

- La possibilité de structurer les modèles, lorsque le nombre de tâches est considérable, n'est fournie par aucune des notations existantes.
- Seules trois notations ont un outil associé disponible et utilisable : AMBOSS, CTT et MAD. Ces notations orientées ingénierie des systèmes interactifs et outillées remplissent une grande partie des critères exposés dans la table. De plus, nous remarquons que ces notations ont un aspect semi-formel intéressant pour la description d'un système interactif critique. En effet, leurs opérateurs d'ordonnancement temporels peuvent être décrits avec un langage mathématique et permettent ainsi des opérations de vérification sur l'ordonnancement temporel des tâches (Santoro, 2005).
    - La notation AMBOSS est conçue pour les systèmes interactifs critiques (avec notamment la possibilité de décrire les moyens d'empêcher le déclenchement d'une tâche si l'intégrité de l'utilisateur ou/et du système est/sont menacée/és, ces moyens étant appelés des barrières). Mais elle ne permet pas de décrire finement les différents types de tâches.
    - La notation CTT permet de décrire les objets utilisés mais ne permet pas de montrer précisément le passage de ces objets entre l'utilisateur et le système.
    - La notation MAD permet aussi de décrire les objets utilisés mais ne permet pas non plus de montrer précisément le passage de ces objets.

Aucune des notations étudiées, mêmes les trois précédentes les mieux positionnées en termes de satisfaction des besoins pour la conception d'un système interactif et/ou critique, ne fournissent pas :

- Une manière de décrire finement les tâches interactives.
- Une manière de décrire finement les tâches utilisateur et en particulier les tâches cognitifs.
- de mécanismes de structuration dans le cas de descriptions de nombreuses tâches.

2.2.3 Problématique de structuration des modèles de tâches

L'utilisation des notations exposées précédemment est souvent décrite à travers des exemples de modélisation d'activités simple. Leur utilisation devient plus ardue lorsque les tâches à décrire sont nombreuses et compliquées, comme c'est le cas dans le domaine des systèmes interactifs critiques, où il faut être capable de structurer l'activité d'utilisateurs surveillant et contrôlant des centaines de paramètres afin de mener à bien des missions complexes et variées . Par exemple, la Figure 28 montre un modèle de tâche construit avec la notation CTT pour décrire une partie de l'activité d'un opérateur d'application de commande et contrôle d'un satellite. Ce modèle, même agrandi, est très peu lisible et difficilement exploitable.

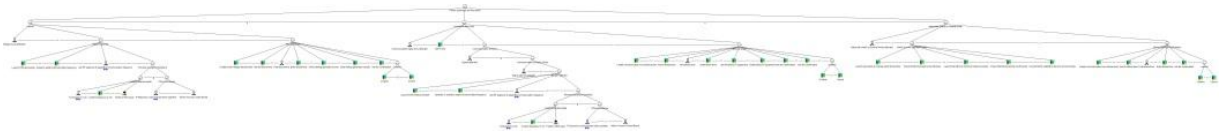


Figure 28. Tentative de modélisation des tâches d'un utilisateur d'application de commande et contrôle d'un satellite

Quel que soit le domaine d'application, cette problématique est apparue il y a quelques années et certains travaux proposent des solutions à ce problème. Deux grandes tendances se dégagent : structurer les modèles grâce à de nouveaux principes ajoutés aux notations et structurer les modèles grâce aux outils logiciels de modélisation.

2.2.3.1 Structuration des modèles grâce aux notations de modélisation

Dans le cadre de modélisation d'activités complexes, la réutilisation et la modularité de tout ou partie des modèles peuvent être envisagées.

Les travaux de (Breedvelt, Paterno, & Severiins, 1997) soutiennent l'approche de la réutilisation. En effet, certaines tâches (comme s'identifier dans un système) sont structurellement similaires dans beaucoup d'applications.

Les travaux de (Gaffar, Sinnig, Seffah, & Forbrig, 2004) discutent de la notion de patrons de tâches (structures récurrentes) pouvant être utilisés dans les modèles de tâches. Ils proposent une méthode et un outil pour modéliser des patrons génériques d'ensembles de tâches à accomplir dans un but récurrent. Ces patrons peuvent ensuite être instanciés et configurés en fonction du système couramment modélisé.

Les travaux de (Sinnig, Forbrig, Wurdel, Chalin, & Khendek, 2007) proposent la notion modèles de tâches modulaires. Ces modules peuvent être structurés dans des diagrammes de tâches décrivant les relations qu'ils peuvent avoir entre eux. La notion de diagramme ajoute un niveau de complexité car une nouvelle notation en plus de celle exprimant les relations entre les tâches. De plus, les auteurs ne fournissent aucun mécanisme pour gérer les flux de données entre ces modules.

2.2.3.2 Structuration des modèles grâce aux fonctionnalités des outils de modélisation

Divers moyens de structuration et d'aide à la gestion de la taille et de la complexité des modèles ont été fournis ces dernières années.

L'outil CTTe (associé à la notation CTT) permet ainsi (Paterno, 2004) :

- De plier et déplier les sous arbres de tâches dans les modèles.
- D'utiliser des sous-ensembles de tâches d'un modèle à un autre en effectuant une action de *copier-coller*. Cependant, la modification d'une ou plusieurs tâches copies ou sources n'est pas propagée aux autres tâches sources ou copies.

Les travaux de (Paterno & Zini, 2004) proposent aussi une version enrichie de l'outil CTTe en y ajoutant des techniques de visualisation (fisheye et zoom sémantique).

De plus, au travers des modèles de tâches collaboratifs, CTTe est le seul environnement fournissant un support à la décomposition des tâches en plusieurs diagrammes communiquant entre eux, même si le but originel est la représentation de la collaboration.

Pour conclure cette sous-section, nous avons montré l'inadéquation entre les notations outillées de modélisation de tâche existantes et les critères requis pour le développement d'un système interactif critique. Cette inadéquation actuelle nous conduit à proposer une notation outillée dans le chapitre 4.

3 Approches d'intégration synergique des modèles

Les sections précédentes ont montré l'importance de l'utilisation outillée des modèles au cours de l'approche de développement d'un système interactif critique. Nous avons montré que les modèles du comportement du système et les modèles de tâches utilisateur permettaient de fournir un support pour la conception et le développement des systèmes interactifs critiques fiables et utilisables.

Les modèles de tâches utilisateurs et les modèles du comportement du système représentent, entre autres, deux points de vue du passage d'informations entre l'utilisateur et le système. Les deux vues

doivent être cohérentes pour que le système soit opérable et que les actions conjointes de l'utilisateur et du système n'altèrent pas l'intégrité de l'un de deux. S'assurer que ces deux types de modèles sont cohérents doit donc faire partie de l'approche de développement des systèmes interactifs critiques complexes et des outils logiciels doivent fournir un support à cette activité.

La première sous-section décrit les solutions proposées jusqu'à aujourd'hui pour intégrer des modèles de différents types lors de la conception et du développement de systèmes interactifs et la seconde sous-section détaille une approche de développement prenant en compte l'intégration synergique de ces deux types de modèles, seule du genre répertoriée à jour.

3.1.1 Solutions existantes pour l'intégration entre des modèles de différents types

Depuis un dizaine d'années, plusieurs propositions d'intégration des modèles de tâches avec les modèles du système ont été apportées dans la communauté scientifique.

3.1.1.1 Génération automatique de modèles

Mobi-D (Puerta & Eisenstein, 1999) et Trident (Bodart, Hennebert, Leheureux, & Vanderdonkt, 1994) sont des exemples d'outils logiciels utilisant les informations contenues dans les modèles pour fournir un support à la conception et au développement d'interfaces utilisateur. Alors qu'avec Mobi-D le concepteur peut choisir différentes stratégies d'utilisation de l'information contenue dans les modèles de tâches et de domaine pour générer l'interface utilisateur, Trident utilise les descriptions et recommandations à propos des interactions abstraites pour la génération automatique d'interface utilisateur indépendante de la plateforme.

D'autres travaux se concentrent sur la génération de modèles à partir d'autres modèles. Les travaux de (Lu, Paris, Vander Linde, & Colineau, 2003) proposent de générer les modèles du système à partir de modèles de tâches ; ces mêmes auteurs proposent aussi la génération de modèles de tâches à partir de modèles du système (Lu, Paris, & Vander Linde, 1998). Les approches dirigées par les modèles (sous-section 1.2.7) proposent d'utiliser plusieurs types de modèles en leur appliquant des règles de transformation. Les travaux de (Limbourg Q. , Vanderdonckt, Michotter, Bouillon, & Lopez-Jaquero, 2004) et (Reichart, Dittmar, Forbrig, & Wurdel, 2008) proposent de créer l'interface utilisateur du système à partir d'autres modèles comme les modèles de tâches, de contexte et d'interface utilisateur abstraite. Cependant, sans l'intervention des concepteurs et développeurs, les règles de transformation peuvent mener à des descriptions irréalistes de l'activité utilisateur (Winckler, Vanderdonckt, Trindade, Stan, & Stanciulescu, 2008).

3.1.1.2 Evaluation de la cohérence entre différents types de modèles

Alors que ces propositions se concentrent sur la production de modèles, d'autres travaux se concentrent sur l'évaluation de la compatibilité entre les tâches utilisateur et le système (Sawyer, Minsk, & Bisantz, 1996). Les travaux de (Palanque, Bastide, & Sengès, 1995) vérifient la compatibilité entre des modèles de tâches (conçus avec la notation UAN) transformés en réseaux de Petri et les modèles du système décrits eux aussi avec des réseaux de Petri. Les travaux de (Palanque, Bastide, & Paternò, 1997) présentent l'utilisation de la notation CTT pour modéliser les tâches utilisateur abstraites et l'utilisation des réseaux de Petri pour modéliser les tâches concrètes. Ces modèles de tâches concrets sont utilisés pour évaluer la complexité des tâches qui seront exécutées par l'utilisateur, au moyen des techniques d'évaluation de performances fournies par les apports théoriques des réseaux de Petri.

Afin de fournir une intégration synergique des modèles de tâche et des modèles du système, (Palanque & Bastide, 1997) proposent une méthode outillée pour exécuter des scénarios extraits de modèles de

tâches (modélisés avec la notation CTT) sur les modèles du système (modélisés avec la notation ICO). Cette approche permet de vérifier la correspondance et l'exhaustivité des tâches utilisateurs avec les possibilités d'interaction fournies par le système grâce à des scénarios concrets. La limitation de cette solution proposée est que des scénarios préenregistrés sont nécessaires pour effectuer cette correspondance et qu'elle ne peut se faire sur une exécution du modèle de tâche.

Plus récemment, (Basnyat, 2006) propose un cadre outillé de modélisation pour l'analyse, la conception et la validation de systèmes critiques interactifs tolérants aux erreurs, afin d'inclure les facteurs humains dans chaque phase de modélisation d'un système interactif critique, en particulier dans le cadre de systèmes critiques en termes de sûreté de fonctionnement. Les travaux de (Blumendorf, Lehman, Feuerstack, & Albayrak, 2008) argumentent qu'une approche dirigée par les modèles serait une solution permettant de fournir un support à la co-exécution des modèles de différents types. Cette co-exécution doit assurer la cohérence et l'exécution bi-directionnelle des modèles (un changement dans un modèle doit déclencher les changements nécessaires dans les modèles liés de l'environnement). Cependant, ces auteurs n'expliquent pas les liens et mises en correspondances nécessaires pour exécuter les modèles de tâches et les modèles du système de manière simultanée.

3.1.2 Approche synergique pour la conception de systèmes interactifs critiques: méthode MEFISTO

La méthode MEFISTO, « Modelling, Evaluating and Formalising Interactive Systems using Tasks and interaction Objects », est une approche basée sur les modèles de tâches et du système proposée dans les travaux de (Palanque, Navarre, & Gaspard-Boulinc, 2000). Elle utilise :

- Le concept d'abstraction durant le processus de conception issu de la méthode Muse*/MERISE (section 1.2.7).
- L'exploitation synergique des modèles de tâches et des modèles formels du système pour la conception de systèmes interactifs vérifiant les propriétés d'utilisabilité et de fiabilité.

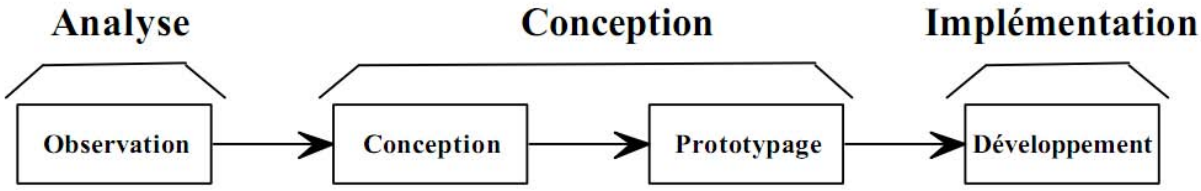


Figure 29. Processus de développement MEFISTO

La Figure 29 décrit les étapes principales du processus de développement MEFISTO.

La phase d'observation (ou analyse) correspond à la collecte d'informations sur le domaine, incluant la pratique déjà existante, les artefacts déjà existants, les contraintes commerciales ou organisationnelles déjà existantes.

La phase de conception structure et abstrait les informations collectées lors de la phase d'observation. Cette phase nécessite la construction de modèles pour représenter ces informations. L'analyse et le raisonnement autour de ces modèles permettent de découvrir les dysfonctionnements dans l'organisation, dans la pratique du travail, dans l'intuition des utilisateurs et dans les artefacts ou le système utilisés. La Figure 30 décrit cette étape de modélisation et son positionnement par rapport à l'effort d'abstraction effectué durant le processus.

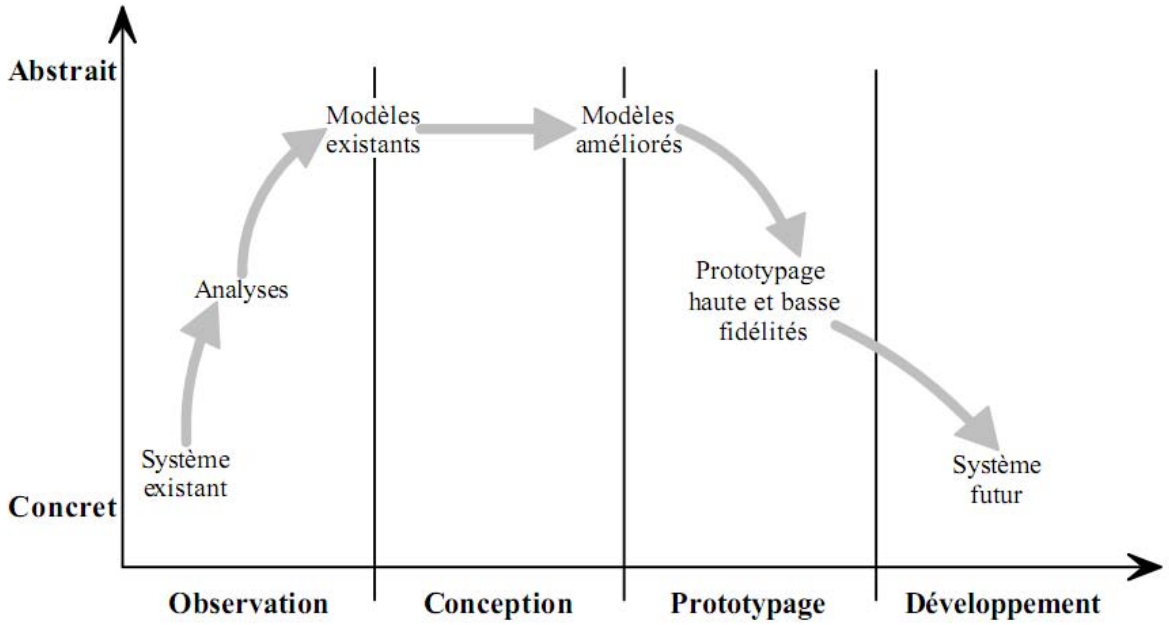


Figure 30. Niveau d'abstraction des phases de développement de la méthode MEFISTO

La phase de prototypage se décompose en deux parties : la phase de prototypage basse-fidélité et celle de prototypage haute-fidélité:

- Le prototypage basse-fidélité a pour but de permettre la compréhension du problème pour produire les besoins précis pour le système à construire. Les techniques utilisées sont dans ce cas le prototypage à l'aide d'un stylo et de feuilles de papier, en demandant régulièrement la participation de l'utilisateur en vue de valider ou invalider la conception. Cela suppose, bien sûr, un processus très itératif qui propose de nombreux choix aux utilisateurs.
- Le prototypage haute-fidélité a pour but de produire un système interactif aussi proche que possible du système final. Ce prototype est construit en tenant compte des leçons tirées par le prototypage basse-fidélité et évolue en fonction de modifications nécessaires après les phases de validation à travers des tests utilisateurs. Le prototype ainsi produit reste tout de même de qualité faible, si bien qu'il ne peut être utilisé comme le système final. La phase de développement vise à produire un logiciel de haute qualité respectant les besoins exprimés tout au long du processus qui a permis de le construire.

Cette phase de conception est itérative et est présentée de manière plus détaillée sur la Figure 31.

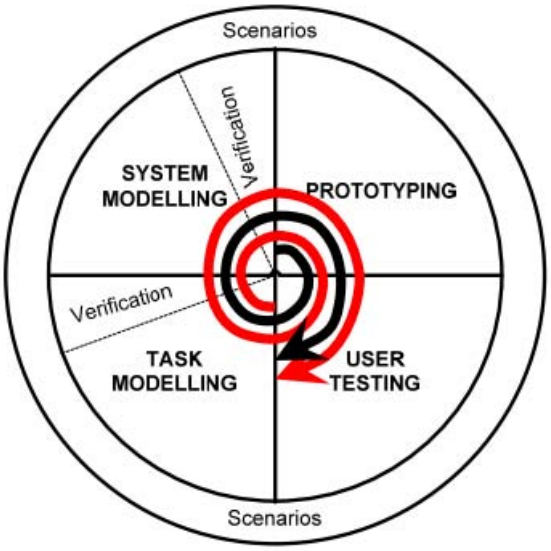


Figure 31. Phase de conception du processus MEFISTO

A l'intérieur du cercle présenté en Figure 31, quatre cadrans représentent les quatre principales activités du processus de conception : la modélisation des tâches, la modélisation du système, le prototypage et les tests utilisateur. Les spirales au centre du diagramme montrent que le processus de conception peut commencer aussi bien avec la modélisation des tâches (spirale gris clair) qu'avec la mise en place d'un premier prototype (spirale noire). Le cercle extérieur, nommé Scénario, illustre l'utilisation des scénarios dans toutes les phases du processus. Ces scénarios peuvent être utilisés pour conduire les expérimentations utilisateur, comme point de départ pour la modélisation de l'activité ou comme un moyen de vérifier le modèle du système.

Un élément important de cette méthode est la possibilité d'utiliser les leçons tirées du prototypage haute-fidélité comme source d'information et de solutions de conception pour la phase de développement. De plus, les modèles formels développés au cours de la phase de conception peuvent être utilisés pour générer des tests qui pourront être appliqués sur le système final. Cela fournit une façon supplémentaire de vérifier que le système construit dans la phase de développement est conforme au prototype développé dans un cadre centré sur l'utilisateur, garantissant un certain niveau d'utilisabilité, là où les méthodes classiques de développement en cascade ne l'encouragent pas.

4 Conclusion

Les approches et techniques présentées dans ce chapitre nous ont permis de dégager les exigences et besoins importants portant sur l'approche de conception et développement de systèmes simultanément fiables, utilisables et opérables :

- L'approche doit prendre en compte l'analyse des besoins utilisateurs et les tests utilisateur mais aussi permettre les itérations dans certaines phases (support à l'utilisabilité).
- L'approche doit prendre en compte la traçabilité des choix de conception et des exigences tout au long du processus de développement (exigence courante dans les standards de développement d'un système critique).
- L'approche doit prendre en compte le développement d'un programme de formation associé au système développé (support à l'opérabilité et à la fiabilité).
- L'approche doit utiliser des modèles formels de comportement du système pour fournir un support à sa vérification et à sa validation (support à l'opérabilité et à la fiabilité).

- L'approche doit utiliser des modèles de tâches utilisateur (support à l'analyse de gros volumes de tâches et au développement du programme de formation).
- L'approche doit fournir un support à la mise en correspondance synergique des modèles de tâches et des modèles du système (vérification de la cohérence entre les tâches utilisateur et le comportement du système).

Parmi les techniques de modélisation présentées dans ce chapitre, nous pouvons retenir une notation outillée de modélisation du comportement du système : ICO – PetShop. Elle répond aux besoins et exigences en termes de support à la conception et au développement d'un système interactif critique simultanément utilisable, fiable et opérable. Cette notation outillée est utilisée dans les chapitres 6 et 7. Concernant la notation outillée de modélisation des tâches utilisateur, aucune des notations outillées existantes n'ayant pu être retenues, une nouvelle notation outillée est proposée dans le chapitre 4.

Parmi les approches de développement existantes présentées dans ce chapitre, aucune d'entre elles ne peut être retenue comme fournissant un support à la conception et au développement d'un système simultanément utilisable, fiable et opérable. Ainsi, le Chapitre 1 tente de répondre aux exigences décrites précédemment en proposant une nouvelle approche de développement. Afin de comprendre et d'investiguer comment intégrer le développement du programme de formation à cette nouvelle approche, nous présentons les approches de développement existantes des programmes de formation pour les systèmes interactifs critiques.

Chapitre 3- Programmes de formation associés aux systèmes interactifs critiques : approches de développement et moyens

La formation à l'utilisation d'un système permet (Salas & Cannon-Bowers, 2001) (Aguinis & Kraiger, 2009):

- De s'assurer que les utilisateurs ont atteint un certain niveau de compétences et connaissances avant l'utilisation d'un système.
- D'améliorer les performances des utilisateurs.
- De diminuer le nombre d'erreurs humaines lors de l'utilisation du système.

Dans ce chapitre, nous présentons dans la première section différents types de formation dans le domaine des systèmes critiques. Dans la seconde section, nous décrivons l'approche systématique généralement utilisée pour concevoir et développer ces programmes de formation. Ensuite, dans la troisième section, nous montrons des manières de mettre en œuvre cette approche ainsi que l'état des connaissances scientifiques sur les outils logiciels fournissant un support au déroulement des programmes de formation.

1 Programmes de formation dans le domaine des systèmes critiques

Dans le domaine des systèmes critiques, les utilisateurs ne sont pas autorisés à prendre leur fonction s'ils ne sont pas qualifiés et certifiés par une autorité particulière ou par leur employeur, selon les domaines d'application. En effet, les utilisateurs (aussi appelés opérateurs comme décrit dans la section 3.2 du chapitre 1) doivent être capables d'appliquer des procédures spécifiques selon des contextes particuliers. Dans cette section, nous montrons que les pratiques de formation des systèmes critiques sont généralement très encadrées et réglementées. Les différentes recommandations expliquent en détail les différentes phases de la formation et mettent en valeur l'importance des objectifs et des mesures de performance avec leur description et taxonomie. Ces recommandations contiennent aussi :

- Une description précise du besoin d'identification des tâches des contrôleurs.
- Une description des différentes phases de formation et de leurs objectifs.
- Les types de support pédagogiques à utiliser pour chaque phase (cours, formation assistée par ordinateur, formation sur simulateur,…).

Les trois sous-sections suivantes appuient notre argument en décrivant les pratiques liées à trois domaines d'application : le contrôle de trafic aérien, le pilotage d'avion commerciaux et la supervision de missions satellitaires.

1.1 La formation à la fonction de contrôleur aérien

Les exigences portant sur la formation d'un contrôleur aérien sont définies par l'organisme EUROCONTROL (European Organisation for the Safety of Air Navigation). Différents documents de recommandations sont en vigueur et certains sont liés à un type de système utilisé pour le contrôle de trafic aérien dans les aéroports. Dans cette sous-section, nous prenons l'exemple du document de

recommandations « EATM Training Progression and Concepts" (EUROCONTROL, European Air Traffic Management Program, 2004) définissant les exigences portant sur la formation d'un contrôleur de trafic aérien.

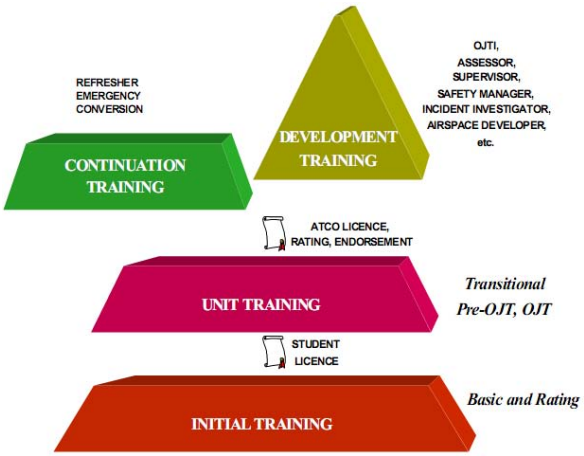


Figure 32. Vue schématique du plan de formation d'un contrôleur de trafic aérien extrait de (EUROCONTROL, European Air Traffic Management Program, 2004)

La Figure 32 est extraite de ce document de recommandation et montre les différentes phases requises dans un programme de formation au contrôle de trafic aérien :

- La phase « Initial training » est la première phase et se divise en 2 étapes :
  - L'étape « Basic training » concerne l'acquisition des connaissances et compétences fondamentales.
  - L'étape « Rating training » concerne l'acquisition des connaissances et compétence dans une spécialité liée à la mission du personnel.

A l'issue de cette phase, une première certification ou licence, « Student licence », est accordée au personnel ayant atteint les objectifs requis en terme de connaissances et compétences.

- La phase « Unit training » est la seconde phase et se divise en 3 étapes :
  - L'étape « Transitionnal training » concerne l'acquisition des connaissances liées au site sur lequel le contrôleur aérien aura à opérer.
  - L'étape « Pre-On-The-Job Training » se concentre sur l'entraînement sur simulateur possédant les caractéristiques du site sur lequel le contrôleur aérien aura à opérer.
  - L'étape "On-The-Job Training" se concentre sur l'entraînement en situation réelle sur site avec un instructeur.

A l'issue de cette phase, les personnels ayant atteint les objectifs requis en termes de connaissances et compétences sont qualifiés et certifiés contrôleurs aérien (« ATCO License »).

- La phase « Continuation training » permet aux contrôleurs aériens déjà en fonction et qualifiés d'élargir leurs connaissances et compétences, mais aussi de les entretenir.
- La phase « Development training » permet aux contrôleurs aériens de se former à une autre carrière et à obtenir une certification pour exercer une autre fonction.

Cette formation est basée sur des objectifs à atteindre en termes de connaissances et compétences. Les élèves doivent remplir des objectifs prédéfinis pour passer d'une étape à l'autre. Plusieurs moyens peuvent être mis en œuvre pour former les élèves et leur permettre d'atteindre les objectifs : cours

magistraux, conférences, visites, étude de cas, exercices assistés par ordinateur (CBPE, Computer-based Practical Exercises), simulation,…

La Table 6 est extraite du document de recommandations « ATCO Basic Training – Training Plans » (EUROCONTROL, European Air Traffic Management Program, 2003) et présente un sous-ensemble d'objectifs pour la première phase du plan de formation. La première colonne contient la dénomination de l'objectif. La seconde colonne contient une ou plusieurs références au contenu de la formation lié à cet objectif le cas échéant. La troisième colonne, appelée « Level », contient permet de quantifier la manière dont l'élève doit atteindre l'objectif. Cette quantification s'effectue sur six niveaux, de 0 à 5, détaillés dans la Table 7. La quatrième colonne, « Type of training event », indique quel moyen de formation est mis en œuvre pour aider l'élève à atteindre l'objectif. CWBT correspond à « Computer/Web-based Training », ISimul correspond à « Individual Simulation », Les à « Lesson » et PTP à « Part-Task Practice » (pratique d'une tâche précise sur un simulateur ou une machine dédiée par exemple).

Table 6. Extrait de la table des objectifs et moyens de formation

Objectives	Training Content	L	Type of Training Event	Educational Material and References
				L = level
2.1.3 Perform communication effectively		3	CWBT ISimul (Sim) PTP	
3 ATC Clearances and Instructions				
3.1 Type and Content of ATC Clearances				
3.1.1 Define ATC clearance	ICAO Annex 2, Chapter 1	1	Les	
3.1.2 Describe the contents of an ATC clearance	ICAO Doc 4444, ICAO Annex 11	2	Les	
3.1.3 Issue appropriate ATC clearances		3	CBPE ISimul (Sim)	
3.2 ATC Instructions				
3.2.1 Define ATC instructions	ICAO Doc 4444, Part 1	1	Les	
3.2.2 Describe the contents of ATC instructions	ICAO Doc 4444, ICAO Annex 11	2	Les	
3.2.3 Issue appropriate ATC instructions		3	CBPE CWBT ISimul (Sim) PTP	

Table 7. Caractérisation des niveaux pour atteindre un objectif extrait de (EUROCONTROL, European Air Traffic Management Program, 2003)

Level 0	'To be aware of'.
Level 1	Requires a basic knowledge of the subject. It is the ability to remember essential points; the learner is expected to memorise data and to restore it.
Level 2	Requires an understanding of the subject sufficient to enable the learner to discuss intelligently. The individual is able to represent for himself or herself certain objects and events, and to act upon these objects and events.
Level 3	Requires a thorough knowledge of the subject and the ability to apply it with accuracy. The learner should be able to make use of his/her repertoire of knowledge to develop plans and activate them.
Level 4	The ability to establish a line within a unit of known applications following the correct chronology and the adequate method to resolve a problem situation. This involves the integration of known applications in a familiar situation.
Level 5	The ability to analyse new situations in order to elaborate and apply one or other relevant strategy to solve a complex problem. The defining feature is that the situation is qualitatively different to those previously met, requiring judgement and evaluation of options.

1.2 La formation à la fonction de pilote d'avions commerciaux

Les exigences portant sur la formation à la fonction de de pilote d'avions sont aussi régies par un organisme international, le JAA (Joint Aviation Authorities). Les recommandations issues du document (JAA, 2006) décrivent, entre autres :

- Les prérequis à la formation en termes d'âge minimum, de certificat médical et de connaissances.
- Les objectifs en termes de compétences et connaissances à atteindre par les élèves pour être qualifiés.
- Les moyens utilisés lors de la formation.

Comme pour le domaine du contrôle de trafic aérien, ce document comporte aussi un ensemble de tableaux de description des objectifs. Un extrait de grille d'évaluation des compétences génériques (non liées à un type d'avion particulier) est présenté dans la Table 8. Cette table décrit les manœuvres et procédures que les pilotes doivent être capable d'effectuer, avec quels moyens ils ont été formés (A pour avion, FS pour simulateur de vol, FTD et OTD pour tout autre matériel d'entraînement) et si les compétences sont considérés comme acquises (la lettre M dans la colonne correspondante indique que le test de cette procédure est obligatoire) et par quel instructeur (dernière colonne).

Table 8. Extrait de la grille d'évaluation des compétences d'un pilote d'avion (JAA, 2006)

	PRACTICAL TRAINING					ATPL/[MPL/]TYPE-RATING SKILL TEST/PROF CHECK	
Manoeuvres/Procedures (including Multi-Crew Cooperation)	OTD	FTD	FS	A	Instructor's initials when training completed	Chkd in FS A	Examiner's initials when test completed
SECTION 1							
1 Flight preparation							
1.1 Performance calculation	P						
1.2 Aeroplane ext. visual inspect.; location of each item and purpose of inspection	[P#]			P			
1.3 Cockpit inspection		P					
1.4 Use of checklist prior to starting engines, starting procedures, radio and navigation equipment check, selection and setting of navigation and communication frequencies	P----->	----->	----->	----->		M	
1.5 Taxiing in compliance with air traffic control or instructions of instructor			P----->	----->			
1.6 Before take-off checks		P----->	----->	----->		M	
SECTION 2							
2 Take-offs							
2.1 Normal take offs with different flap settings, including expedited take off			P----->	----->			
2.2* Instrument take-off; transition to instrument flight is required during rotation or immediately after becoming airborne			P----->	----->			
2.3 Cross wind take-off (A, if practicable)			P----->	----->			
2.4 Take-off at maximum take-off mass (actual or simulated maximum take-off mass)			P----->	----->			
2.5 Take-offs with simulated engine failure			P----->	----->			
2.5.1* shortly after reaching V_2,							

1.3 La formation à la fonction d'opérateur de commande et contrôle de missions satellitaires

Dans le domaine de l'industrie spatiale, l'organisme européen ECSS (European Cooperation for Space Standardisation) définit un ensemble d'exigences et de recommandations pour réglementer les activités spatiales. Parmi elles, le standard portant sur les systèmes sols et les opérations, ECSS-E-ST-70C (European Cooperation for Space Standardization, 2008) traite des opérations sur les systèmes de commande et contrôle, au sol, des missions spatiales.

Ce standard dicte des recommandations et exigences portant sur toutes les phases d'une mission spatiale, des phases amonts d'analyse de la mission jusqu'à la fin de l'exécution de la mission, en passant par l'analyse des tâches et opérations à effectuer sur les systèmes sols pour mener à bien la mission. Le standard exige de démarrer la formation des opérateurs une fois le système développé, vérifié et livré. A la fin de leur formation, les opérateurs doivent connaître de manière approfondie leur mission («The operations teams shall be assigned and trained such that they are familiar with the mission and the supporting facilities before the start of operational validation", (European Cooperation for Space Standardization, 2008)). Ce standard fournit aussi une liste d'étapes devant être effectuées lors de l'exécution du plan de formation, telles que les simulations et les répétitions de scénarios opérationnels.

2 Approches systématiques au développement des programmes de formation

L'approche systématique à la formation (« Systematic Approach to Training ») est une approche conçue pour le développement d'un programme de formation. Cette approche est largement utilisée depuis une trentaine d'années dans le domaine des systèmes critiques pour concevoir et développer des programmes de formation.

Cette approche a été proposée par (Branson, R.K., et al., 1975) pour concevoir et mettre en œuvre des programmes de formation des personnels militaires américains. Elle fournit un cadre de développement permettant de développer et d'améliorer les programmes de formation de manière itérative et incrémentale. Les travaux de (Reiser, 2001) présentent l'historique de cette approche et de ses différents courants.

Les caractéristiques principales de l'approche systématique à la formation sont (Branson & Grow, 1987) :

- La définition précise des besoins en formation. Cette caractéristique est de plus en plus appliquée par les approches existantes mais était plutôt novatrice il y a 30 ans, lorsque la plupart des approches s'appuyaient sur la diffusion générique de connaissances et de savoirs.
- Le développement et la mise en œuvre basés sur les résultats. Cette caractéristique est commune à un grand nombre d'approches de développement et de mise en œuvre de programmes de formation.

Le modèle sous-jacent du processus de développement utilisé par cette approche est le processus ADDIE (Analysis Design Development Implementation Evaluation), présenté dans la Figure 33.

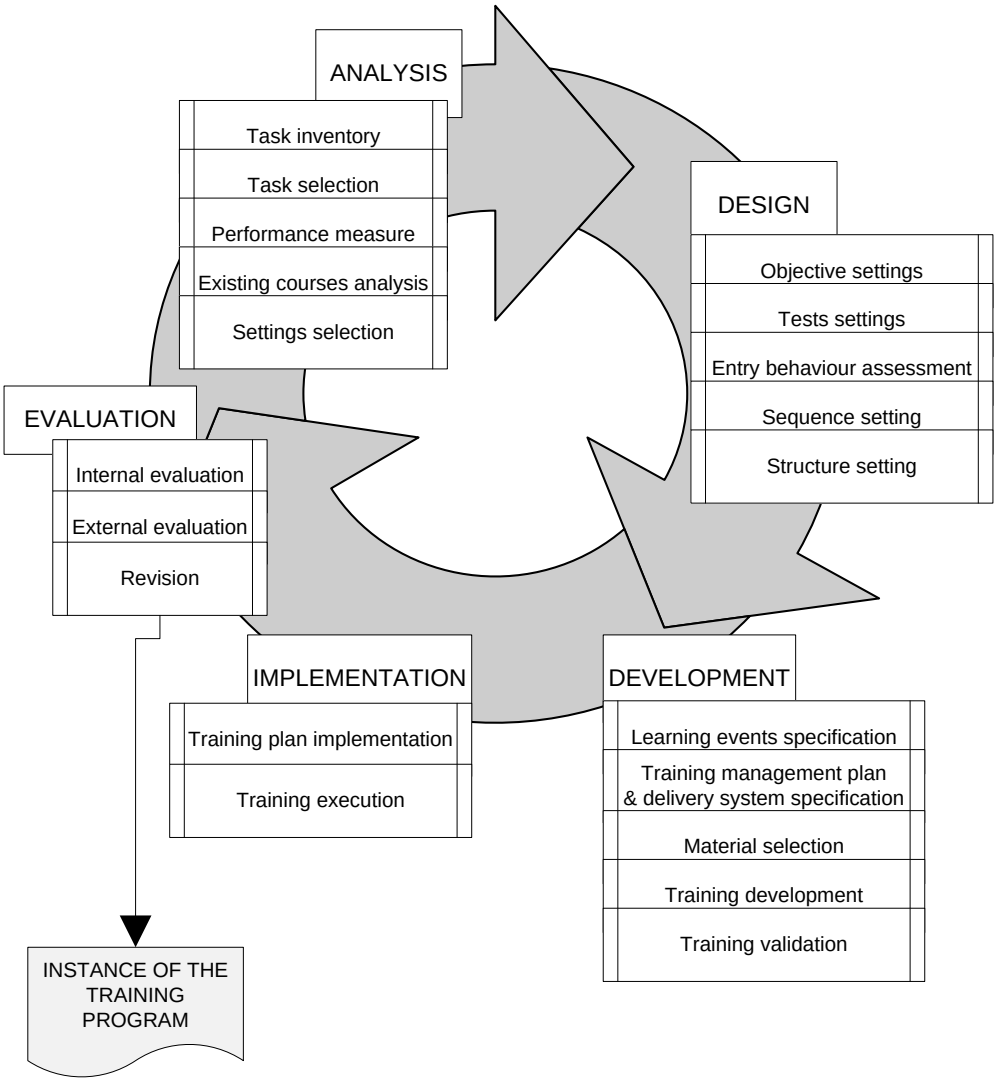


Figure 33. Processus de développement ADDIE

2.1 Description des phases génériques : « Analysis, Design, Development, Implementation, Evaluation »

Le processus ADDIE (Figure 33) se compose de cinq phases, reconduites cycliquement jusqu'à l'obtention d'une instance d'un programme de formation (U.S. Army Field Artillery School, 1984) :

- *Analysis* : La phase d'analyse permet de rassembler et d'analyser les besoins en formation, les objectifs de la formation ainsi que toutes les informations nécessaires pour le développement du programme de formation (mission, tâches, scénarios, contexte, moyens,…). Elle est composée des sous-phases suivantes :
    - *Task inventory* : Production de l'inventaire des tâches liées aux missions et au métier.
    - *Task selection* : Sélection des tâches qui devront être effectuées pour accomplir correctement les missions cibles.
    - *Performance measure* : Mesure des performances moyennes pour les tâches sélectionnées sur les systèmes opérés existants avec l'aide d'experts métier.

    - *Existing courses analysis* : Etude des programme de formation déjà développés pour le métier cible afin de réutiliser le maximum d'information pertinente déjà produite.
    - *Settings selection* : Choix des types et moyens de formation (cours magistraux, entraînement avec simulateur, formation assistée par ordinateur,…)
- *Design* : La phase de conception a pour but de créer et structurer le programme de formation et son contenu. Elle est composée des sous-phases suivantes :
    - *Objective settings* : Sélection des objectifs à remplir par la personne formée pour chaque tâche sélectionnée.
    - *Tests settings* : Sélection des moyens d'évaluation des compétences des personnes formées et des modalités de mesure des performances.
    - *Entry behaviour assessment* : Sélection des prérequis (compétences et capacités) pour suivre la formation.
    - *Sequence setting* : Ordonnancement des différentes activités prévues dans la formation, en fonction de leur complexité, du contexte, du personnel, et des types d'apprentissage prévus.
    - *Structure setting* : Décomposition des différentes activités prévues dans la formation, en fonction de leur complexité, du contexte, du personnel, et des types d'apprentissage prévus.
- *Development* : La phase de développement utilise les résultats des deux phases précédentes pour produire les matériaux concrets nécessaires à la formation.
    - *Learning events specification* : La spécification des moyens d'apprentissage utilisés pour atteindre les objectifs d'apprentissage.
    - *Training management plan & delivery system specification*: La spécification du projet de formation (ressources humaines et matérielles, lieux, emploi du temps,…)
    - *Material selection*: La sélection des supports pédagogiques utilisés.
    - *Training development*: Le développement des supports pédagogiques.
    - *Training validation*: La validation du projet de formation et des supports.
- *Implementation* : La phase de mise en œuvre du programme de formation se compose des sous-phases suivantes :
    - *Training plan implementation* : Production de tous les matériaux nécessaires à la formation, préparation des formateurs, mise en œuvre du projet de formation.
    - *Training execution* : Mise en œuvre de la formation.
- *Evaluation* : La phase d'évaluation permet d'évaluer les personnels formés mais aussi l'efficacité et l'efficience du programme de formation. Elle se compose ainsi des sous-phases suivantes :
    - *Internal evaluation* : Evaluation interne du programme de formation, pour déterminer si le programme de formation a permis d'atteindre les objectifs et dans quelle mesure.
    - *External evaluation* : Evaluation externe du programme de formation, pour déterminer si le personnels formés a atteint les objectifs fixés par la formation et dans quelle mesure.
    - *Revision* : Spécification des modifications à effectuer au programme de formation par rapport aux problèmes rencontrés lors des évaluations interne et externe.

L'évaluation d'un programme de formation est généralement effectuée sur quatre aspects (Kirkpatrick & Kirkpatrick, 2006) :

- L'évaluation des réactions, pour déterminer si la personne formée est satisfaite de la formation qu'elle vient de recevoir.

- L'évaluation des apprentissages, pour déterminer si les connaissances, compétences et comportements suite à la formation ont évolués vers les objectifs prédéterminés.
- L'évaluation du niveau de transfert, pour déterminer si les connaissances, compétences et comportements acquis sont utilisés au jour le jour par la personne formée dans sa mission.
- L'évaluation des résultats, pour déterminer si les connaissances, compétences et comportements acquis ont amélioré les résultats de l'organisation dans laquelle la personne formée accompli sa mission (diminution du nombre de de pannes ou d'accidents).

Les sous-phases du processus ADDIE présentées dans cette section sont celles principalement rencontrées dans les approches systématiques à la formation. Elles peuvent être raffinées en fonction du contexte et des différents courants de recherche académique. Ces variations sont présentées et discutées par (Clark, 2004).

2.2 L'analyse des tâches comme aspect central de l'approche

Les approches systématiques à la formation s'appuient principalement sur l'activité d'analyse de tâches du métier auquel le personnel doit être formé (Patrick, 1992) . L'analyse de tâches est au cœur de la phase d'analyse du processus ADDIE. Et elle est utilisée tout au long de la suite du processus pour construire le programme, pour évaluer le personnel formé et pour évaluer le programme de formation.

Les travaux de (Annett & Duncan, 1967) proposent une notation et une méthode d'analyse de tâches pour identifier les compétences à acquérir par les opérateurs d'un système : HTA (Hierarchical Task Analyis) (Anett, 2004). Cette notation, originellement conçue pour fournir un support aux approches systématiques à la formation, sera utilisée dans divers domaines d'application dans les décennies suivantes (cet aspect est discuté dans la section 2.2 du chapitre 2) et rendue graphique suite aux travaux de (Shepherd, 1985). L'analyse de tâches permet d'identifier et de spécifier les comportements observables à maîtriser par les opérateurs (Reiser, 1987).

De plus, les travaux de (Gagné R. , 1985) et (Gagné, Wager, Golas, & Keller, 2005) argumentent que l'identification des tâches et sous-tâches devant être correctement exécutées doit s'accompagner de l'identification :

- De la situation dans laquelle s'effectue la tâche.
- Des objets requis pour effectuer la tâche.
- Des actions effectuées pour accomplir la tâche.
- Des outils utilisés pour accomplir la tâche.

Cette identification permet de spécifier le plus exhaustivement possible les tâches et activités à accomplir, et fournit un support pour l'évaluation.

Pour conclure cette section, l'approche systématique au développement d'un programme de formation fournit plusieurs moyens :

- Une méthode pour s'assurer de la mise à jour et du maintien des compétences de l'équipe durant sa mission.
- L'analyse du métier et des tâches d'un opérateur.
- La spécification des objectifs en termes de performance opérateur et la mesure de ces performances.
- L'évaluation et l'optimisation continuelle du programme de formation pour s'adapter aux personnes formées, au système sur lequel elles doivent opérer et au contexte.

3 Mise en œuvre des approches et techniques de développement de programmes de formation

Les approches systématiques à la conception d'un programme de formation sont utilisées dans plusieurs domaines d'application des systèmes critiques, comme exemplifié dans la sous-section 3.1. Plusieurs outils logiciels sont aussi développés pour fournir un support à la mise en œuvre de programmes de formation à l'utilisation de systèmes critiques, comme décrits dans la sous-section 3.2.

3.1 Exemples d'application d'une approche systématique

3.1.1 Domaines aéronautique et aérospatial

Dans le domaine de l'aviation civile, la FAA (Federal Aviation Administration) fournit des directives aux compagnies aériennes pour la mise en place des programmes de formation des pilotes (Seamster, Boehm-Davis, Holt, & Schultz, 1998). L'approche décrite est aussi systématique et basée sur l'évaluation des performances et l'amélioration du programme en fonction des évaluations externes et internes du programme de formation (Federal Aviation Administration, 2011).

Dans le domaine spatial, un exemple est décrit dans les travaux de (Scott, 2009). Au sein de la NASA (National Aeronautics and Space Administration) aux États-Unis, il décrit la mise en place d'une approche systématique pour le développement et la mise en œuvre d'un programme de formation dans le cadre d'une mission spatiale. Il argumente, étude de cas à l'appui, que l'utilisation d'une approche systématique permet de développer des programmes de formation effectifs.

3.1.2 Domaine industriel de l'énergie

Dans le domaine industriel de l'énergie atomique, l'IAEA (International Atomic Energy Agency) fournit des directives (International Atomic Energy Agency, 2009) pour :

- Le développement d'un programme de formation.
- Le recrutement du personnel à former.
- L'évaluation du personnel formé.
- La qualification du personnel avant opération du système.

Ces directives préconisent l'utilisation d'une approche systématique au développement des programmes de formation et proposent un processus suivant les étapes de l'approche ADDIE (présentée dans la section précédente).

Dans le domaine industriel de l'énergie électrique, les travaux de (Neitzel, 2006) démontrent aussi l'efficacité des approches systématiques pour concevoir, développer et mettre en œuvre des programmes de formation. Il s'appuie sur l'étude de cas du développement d'un programme de formation pour les techniciens de maintenance des équipements électriques de centrales électriques.

3.2 Outils logiciels pour la mise en œuvre des programmes de formation

Les travaux de (Patrick, 1992), (Salas & Cannon-Bowers, 2001) et (Aguinis & Kraiger, 2009) présentent un état de l'art complet sur les méthodes et techniques existantes pour mettre en œuvre des programmes de formation, et distinguent plusieurs thèmes de recherche (qui correspondent à différentes phases de la conception d'un programme de formation). Leurs travaux distinguent le thème

de recherche sur l'analyse des besoins en formation (qui se rapproche de la phase d'analyse des approches systématiques), le thème de recherche sur les méthodes et stratégies de formation, dont les outils logiciels pour la formation, ainsi que le thème de recherche sur les méthodes d'évaluation.

3.2.1 Outils logiciels fournissant un support à l'analyse des besoins pour un programme de formation

Nous avons discuté de l'analyse des besoins en formation avec l'analyse des tâches liées au métier dans la sous-section 2.2 et plusieurs outils de modélisation de tâches, comme HTA (Annett & Duncan, 1967), ont été proposés dans ce but, comme décrit dans la section 2.2 du chapitre 2. Cependant, ces outils de modélisation ne sont pas adaptés à la gestion d'un nombre important de tâches.

D'autre part, certaines méthodes, n'ayant pas pour but d'être directement appliquées au domaine de la formation, peuvent être utile pour le développement des programmes de formation. C'est le cas des travaux de modélisation des performances humaines, comme ceux de (Gore, Hooey, Haan, Bakowski, & Mahlstedt, 2011). Ils peuvent aussi être utilisés en phase d'analyse du programme de formation, dans la sous-phase appelée « Performance measure » (décrite dans la sous-section 2.1), pour prédire les performances moyennes lorsque le training est conçu pour un nouveau système. En effet, dans ce cas, l'expert métier ne peut pas forcément extrapoler les performances moyennes sur un système qu'il ne connaît pas.

3.2.2 Outils logiciels fournissant un support au déroulement d'une formation

Plusieurs types d'outils logiciels, faisant partie de la catégorie « Computer-Based Training », peuvent être utilisés lors de la mise en œuvre du programme de formation (Bedwell & Salas, 2010). Dans la première section de ce chapitre, nous avons évoqué la formation assistée par ordinateur, l'utilisation de simulateurs et d'exercices pratiques assistés par ordinateur pour les programmes de formation des contrôleurs de trafic aérien et les pilotes d'avions. Ces outils ont pour vocation de :

- Permettre aux élèves de se former selon leur rythme et disponibilité.
- Réduire les coûts du programme de formation (en remplaçant l'instructeur).
- Confronter les élèves à des situations réelles avec les outils de simulation.
- S'adapter finement aux progrès de l'élève avec les outils appelés « Intelligent Tutoring Systems ».

Concernant la simulation, les travaux de (Kincaid & Westerlund, 2009) présentent un état des lieux sur son utilisation pour la formation et argumentent que les exigences et besoins pour la formation doivent être définis de manière précise et mis en relation avec l'environnement de simulation afin que l'environnement ne devienne pas plus matière à divertissement qu'à acquérir des connaissances et compétences.

De nouveaux types d'outils logiciels font actuellement l'objet de plusieurs travaux :

- Les environnements de formation virtuels pour la formation sont un axe de recherche particulier de la simulation et permettent de simuler des environnements réels de manière très détaillée. Tout comme les outils de simulation, ils fournissent un support à des entraînements qui n'auraient pas forcément été possibles dans le monde réel (lieux inaccessibles, scénarios trop dangereux). Par exemple, les travaux de (El-Chaar, Boer, Pedrazzoli, Mazzola, & Dal Maso, 2011) montrent que l'utilisation d'un environnement virtuel de formation pour les personnels d'une usine de fabrication de charpente permet de faciliter les explications du formateur pour les opérations complexes nécessitant l'utilisation de différents outils. Les

travaux de (Loftin & Kenney, 1994) argumentent que l'utilisation des environnements virtuels pour la formation permet d'augmenter l'efficacité du personnel dans ses fonctions.

- Les « Intelligent Tutoring Systems » (ITS) sont issus du domaine de recherche de l'intelligence artificielle et ont pour vocation de surveiller en temps réel les performances de l'élève et d'adapter aussi en temps réel le programme de formation ou la session de formation courante. Par exemple, les travaux de (Amokrane, Lourdeaux, Burkhardt, & Barthès, 2008) utilisent ce type d'outil logiciel pour fournir un support à un programme de formation pour opérateurs de systèmes critiques. Les travaux de (Fu, et al., 2006) se consacrent à un outil logiciel de formation basé sur le modèle cognitif de l'élève et évoluant en temps réel tout au long du programme de formation, afin de prendre en compte les compétences déjà acquises et celles restant à acquérir. Cependant, dans le cas des travaux de (Fu, et al., 2006), les modèles utilisés ne sont pas connectés au système lui-même et le support logiciel au programme de formation est utilisé de manière séparée du système.

3.2.3 Outils logiciels fournissant un support à l'évaluation

Comme expliqué dans la sous-section 2.1 de ce chapitre, la phase d'évaluation vise à évaluer les élèves suite à leur formation et/ou le programme de formation lui-même. Nous avons trouvé peu de références à des outils logiciels fournissant un support à l'évaluation des programmes de formation. Les travaux de (Hofer & Ruggiero, 1990) proposent une méthode pour intégrer l'évaluation des réactions et des apprentissages pour des sessions d'apprentissage du pilotage d'avion sur un simulateur. Plus récemment, les travaux de (Okazaki, Muromura, & Kaynano, 2011) s'intéressent à établir des moyens outillés pour l'évaluation des capacités motrices d'élèves pilotes dans la marine.

3.2.4 Outils logiciels fournissant un support à l'ensemble des étapes du développement d'un programme de formation

Certains travaux envisagent des techniques en support à la formation comme moyens intégrés à une approche plus générique. Ainsi, les travaux de (Feary, Sherry, Polson, & Palmer, 2000) proposent une approche intégrée pour la conception des systèmes de pilotage d'avions et la conception du programme de formation aux fonctions de pilote de ces systèmes. Ces travaux discutent de l'importance de la connaissance du modèle conceptuel du système dans l'esprit des pilotes. L'approche intégrée proposée est basée sur une description textuelle, dans une matrice, des scénarios d'utilisation et des comportements du système et ne fournit pas un support logiciel pour le déroulement de sessions de formation.

4 Conclusion

Dans ce chapitre, nous avons montré que les approches systématiques sont largement utilisées dans le domaine des systèmes critiques car elles permettent :

- D'analyser et de spécifier les tâches indispensables pour mener à bien une mission.
- De mettre à jour, évaluer, qualifier et maintenir les compétences et connaissances des utilisateurs de ce type de systèmes.

Cependant, les standards et directives couvrant les différents domaines d'application des systèmes critiques, mêmes si certains d'entre eux sont très précis sur les objectifs à atteindre, les critères d'évaluation, ainsi que les supports pédagogiques, n'indiquent pas de moyens précis pour mesurer les connaissances et compétences acquises.

D'autre part, plusieurs moyens de mettre en œuvre des programmes de formation sont actuellement investigués (comme décrits dans la section 3), mais ils répondent à des besoins ponctuels dans le programme de formation et ne proposent pas de solution globale pour fournir un support à chaque étape d'une approche systématique au développement du programme de formation.

Le chapitre 5 propose une approche de développement systématique d'un système interactif critique intégrant le développement du programme de formation associé afin de fournir un support aux phases d'analyse de tâches et d'évaluation des performances. Le chapitre 6 présente des moyens pour la mise en œuvre de l'approche proposée par cette thèse.

Chapitre 4– HAMSTERS : une notation et un outil logiciel pour la modélisation et la simulation des tâches utilisateur

Ce chapitre présente une notation outillée pour la modélisation et la simulation des tâches utilisateur. Cette notation remplit les exigences et besoins nécessaires (tels qu'exposés dans le la sous-section 2.2 du chapitre 2) pour fournir un support à la conception et au développement d'un système interactif critique simultanément utilisable, fiable et opérable.

La première version de cette notation a été conçue dans le cadre d'un projet réalisé par des étudiants lors de leur dernière année d'études de master IHM (Azzouzi, André, & Hoarau, 2009) et visait à créer un éditeur de modèles de tâches centré utilisateur de modèles. Notre contribution enrichit cette première version de la notation en y apportant des extensions ainsi qu'en proposant un outil logiciel pour fournir un support à l'approche proposée dans cette thèse.

HAMSTERS (Human-centered Assessment and Modeling to Support Task Engineering for Resilient Systems) utilise certains mécanismes des notations de modélisation de tâches existantes. Elle utilise les concepts de décomposition hiérarchique des buts en sous-buts et de raffinement du type de tâches, concepts présents dans un grand nombre de notations (comme décrit dans la section 2.2 du chapitre 2). Plus particulièrement, elle utilise les opérateurs temporels LOTOS (ISO, 1989), aussi utilisés par la notation CTT (Paterno, Mancini, & Meniconi, 1997) présentée dans la section 2.2 du chapitre 2. La notation HAMSTERS offre une représentation graphique et possède un outil logiciel associé HAMSTERS[4] permettant d'éditer, de classer et de simuler des modèles de tâches.

Les quatre premières sections de ce chapitre expliquent la manière dont la notation HAMSTERS satisfait aux besoins présentés dans la section 2.2 du chapitre 2 pour le développement de systèmes interactifs et de systèmes critiques et pour le développement de leurs programmes de formation associés :

- La section 1 explique comment HAMSTERS permet de différencier finement les types de tâches pour fournir un support à l'analyse de la répartition des tâches entre l'utilisateur et le système et pour fournir un support à la conception du programme de formation associé.
- La section 2 explique comment HAMSTERS permet de structurer un grand nombre d'activités pour fournir un support à la modélisation des tâches effectuées avec un système complexe.
- La section 3 explique comment HAMSTERS permet de décrire les flux de données entre le système et l'utilisateur afin de représenter les données et leur traitement.
- La section 4 explique comment HAMSTERS permet de modéliser les erreurs humaines afin de fournir un support à l'analyse des fautes et leur prévention lors de la conception des systèmes interactifs critiques.

La cinquième section présente le méta-modèle d'HAMSTERS afin de discuter des différences avec les autres notations. La dernière section décrit l'outil logiciel associé et explique comment il fournit un support à la validation des modèles et à l'exécution et la production de scénarios utilisateur.

[4] http://www.irit.fr/recherches/ICS/softwares/hamsters/index.html

Au cours de ce chapitre, nous utilisons l'exemple d'un utilisateur interagissant avec un distributeur de billet pour retirer de l'argent liquide. Cet exemple permet d'illustrer chaque concept présenté au fil des sections.

1 Types de tâches

Les types de tâches permettent de décrire les activités de l'utilisateur en fournissant des informations sur la nature de la tâche. Plus les types de tâches sont précis, plus il est aisé d'analyser les interactions entre le système et l'utilisateur ainsi que l'activité humaine. La première sous-section présente les types de tâches génériques, puis les seconde et troisième sous-sections présentent les raffinements des tâches interactives et des tâches utilisateurs. Les raffinements des tâches interactives et des tâches utilisateur seront précisément les éléments de la notation permettant :

- L'analyse de l'adéquation entre les tâches utilisateur et les fonctions du système.
- L'évaluation des performances de l'utilisateur.

1.1 Présentation générale

La notion de tâche est générique, elle peut être un but ou une action utilisateur. Ainsi, une tâche peut être un but et elle-même se décomposer en un sous-ensemble de tâches, mais elle peut aussi être une activité ou action atomique à effectuer dans le cadre d'un but de plus haut niveau d'abstraction. La Table 9 présente les types de tâches génériques fournis par la notation HAMSTERS et leur représentation.

Table 9. Types de tâches génériques de la notation HAMSTERS

Types de tâche	Représentation HAMSTERS
Tâche abstraite	 Tâche abstraite
Tâche (fonction) système	 Fonction système
Tâche utilisateur	 Tâche utilisateur
Tâche interactive	 Tâche abstraite interactive

Une **tâche abstraite** permet de décrire une tâche rassemblant des sous-tâches de types différents ou une tâche n'ayant pas encore de type défini.

Une **tâche (fonction) système** permet de représenter une fonction exécutée par le système.

Une **tâche utilisateur** permet de décrire une activité de l'utilisateur, ou une tâche rassemblant un ensemble d'activités utilisateur.

Une **tâche interactive** permet de décrire un passage d'information entre l'utilisateur et le système, ou une tâche rassemblant un ensemble de tâches interactives.

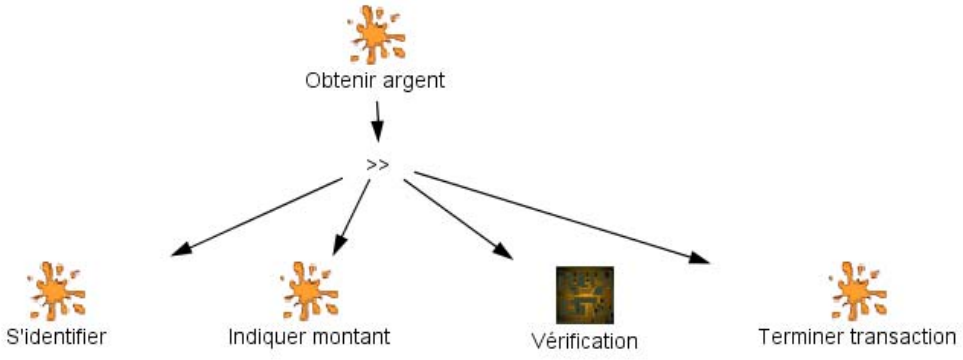


Figure 34. Exemple illustratif du distributeur de billets : modèle de tâche abstrait

La Figure 34 présente le modèle de tâches abstrait et peu détaillé de l'utilisation d'un distributeur de billets pour obtenir de l'argent liquide. La tâche abstraite « Obtenir argent » est la séquence des tâches abstraites « S'identifier » et « Indiquer montant », puis de la fonction système « Vérification » et enfin de la tâche abstraite « Terminer transaction ».

1.2 Raffinement des tâches interactives

Le raffinement des tâches interactives n'est proposé par aucune autre notation. Pourtant, il permet de caractériser la manière dont le flux d'information transite entre l'utilisateur et le système. Nous distinguons trois tâches interactives :

- Une **tâche interactive d'entrée** (représentée par le symbole de gauche sur la Figure 35) permet de décrire que l'utilisateur agit sur le système en lui fournissant un flux entrant d'information.
- Une **tâche interactive de sortie** (représentée par le symbole du milieu sur la Figure 35) permet de décrire que l'utilisateur reçoit un flux d'information sortant du système.
- Une **tâche interactive d'entrée/sortie** (représentée par le symbole de droite sur la Figure 35) permet de décrire que l'utilisateur envoie et reçoit simultanément des flux d'information avec le système.

Figure 35. Possibilités de raffinement d'une tâche interactive

Dans le cadre de l'activité de modélisation, le choix d'une tâche interactive raffinée permet de détailler précisément le types de flux d'information entre l'utilisateur et le système et de différencier :

- Les actions particulières de l'utilisateur sur le système.
- Les attentes du système envers l'utilisateur et les renseignements communiqués par le système à l'utilisateur.

La Figure 36 présente une nouvelle version du modèle de tâches de l'exemple illustratif du retrait d'argent. Dans cette version, le sous arbre de la tâche « S'identifier » est détaillé en une séquence de trois tâches interactives :

- La tâche interactive d'entrée « Insérer carte » permet de décrire le passage d'un objet de l'utilisateur vers le distributeur de billets.
- La tâche interactive de sortie « Demande du code » permet de décrire que le distributeur montre à l'utilisateur qu'il est en attente d'une information de la part de l'utilisateur.
- La tâche interactive d'entrée « Entrer le code » permet de décrire le passage d'une information de l'utilisateur vers le distributeur.

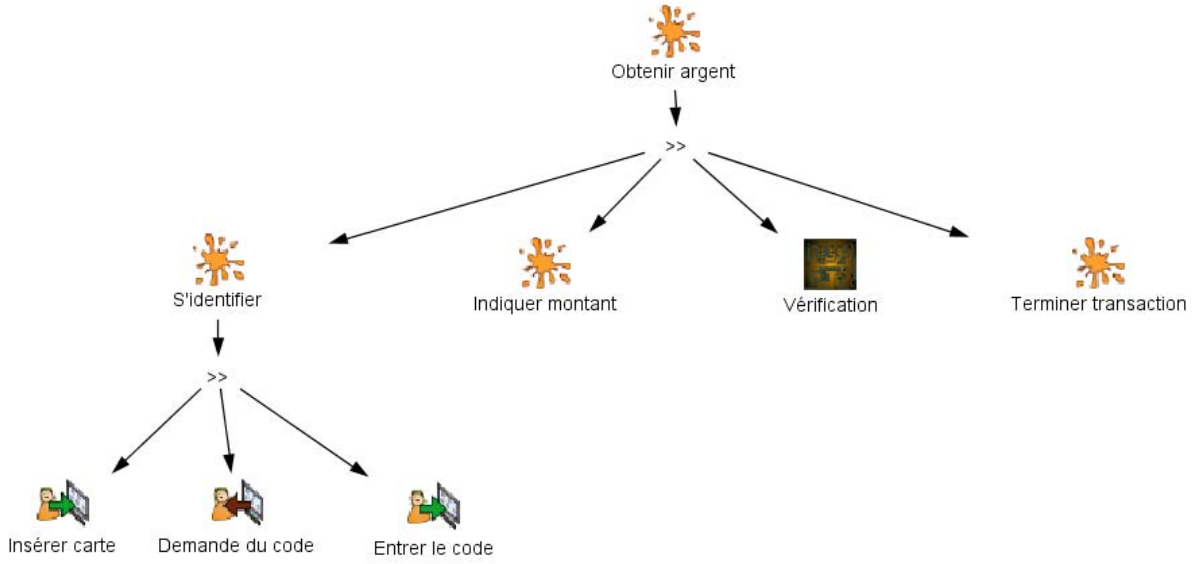


Figure 36. Exemple illustratif du distributeur de billets : raffinement de la tâche d'identification avec des sous-tâches interactives

Cette description fine des tâches interactives permet d'analyser l'adéquation entre le système et l'utilisateur. Par exemple, elle permet de mettre en valeur les actions répétitives de l'utilisateur sur le système et de concevoir le système différemment afin d'optimiser ces actions. Cet exemple a fait l'objet d'une publication et est détaillé dans (Martinie, Palanque, Barboni, & Ragosta, 2011).

1.3 Raffinement des tâches utilisateur

Le raffinement des tâches utilisateur permet de prendre en compte les différents aspects d'une activité menée par un utilisateur. Les tâches utilisateurs peuvent être de natures différentes selon la manière dont l'utilisateur traite l'information en provenance du monde extérieur.

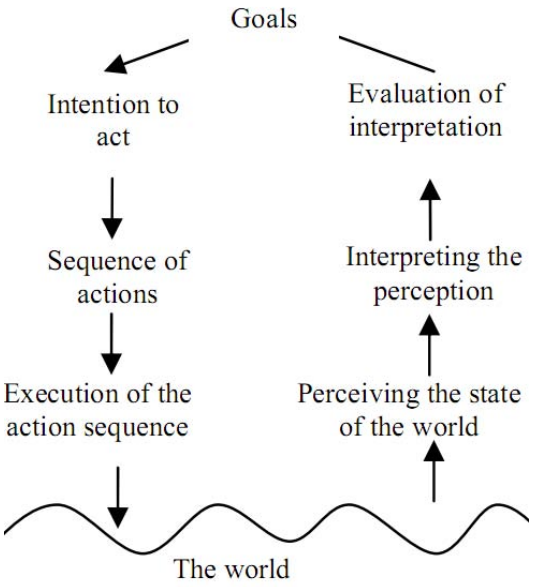


Figure 37. Les sept étapes du la théorie de l'action (Norman D. A., 2002)

La Figure 37 décrit les sept étapes de la théorie de l'action proposée par (Norman & Drapper, 1986). Chaque étape présente un traitement de l'information en provenance du monde extérieur par un utilisateur. La théorie de l'action montre que la réalisation d'une tâche met en jeu sept activités de types différents. L'utilisateur établit un but puis une intention, puis détermine une séquence d'action à accomplir et exécute cette séquence d'actions. Ensuite, il perçoit l'état actuel du système, puis interprète sa perception de l'état du système et évalue sa ou ses interprétations par rapport à son but. Cette description implique donc que l'utilisateur traite l'information de manière mentale, mais aussi de manière physique et sensorielle. Elle a pour but de mettre en valeur deux types de distances :

- La distance d'évaluation entre l'état du système et l'évaluation de l'état du système par l'utilisateur en fonction de ses buts.
- La distance d'exécution entre les buts de l'utilisateur et l'exécution d'une séquence d'action par l'utilisateur sur le système.

La description des tâches utilisateur de différents sous-types est donc nécessaire pour pouvoir : évaluer ces deux types de distances, essayer de les réduire, ou trouver des solutions aux étapes problématiques.

Les travaux de (Card, Newell, & Moran, 1983) proposent un modèle du processeur humain qui représente les différents types d'activités humaines selon les unités de traitement suivantes : processeur perceptif, processeur cognitif et processeur moteur. La notation HAMSTERS permet de décrire ces trois types d'activités humaines grâce aux éléments suivants :

- Une **tâche motrice** (représentée par le symbole de droite sur la Figure 38) permet de décrire un mouvement physique de la part de l'utilisateur (appuyer sur un bouton)
- Une **tâche perceptive** (représentée par le symbole de gauche sur la Figure 38) permet de décrire une action sensorielle de récupération d'informations extérieures (regarder un écran)
- Une **tâche cognitive** (représentée par le symbole de milieu sur la Figure 38) permet de décrire un traitement de l'information contextuelle.

Figure 38. Possibilités de raffinement d'une tâche utilisateur

La Figure 39 présente une nouvelle version du modèle de tâches de l'exemple illustratif du retrait d'argent liquide. Dans cette nouvelle version, la tâche « S'identifier » a été détaillée avec des tâches utilisateur :

- La tâche motrice « Saisir la carte » permet de décrire que l'utilisateur doit effectuer une activité motrice pour attraper sa carte.
- La tâche perceptive « Percevoir demande de code » permet de décrire que l'utilisateur doit percevoir que le système est en attente de la valeur du code de la part de l'utilisateur.
- La tâche cognitive « Se rappeler le code » permet de décrire que l'utilisateur doit effectuer une activité cognitive pour se rappeler son code avant de le fournir au système.
- La tâche motrice « Taper le code » permet de décrire que l'utilisateur doit effectuer une activité motrice pour taper le code.

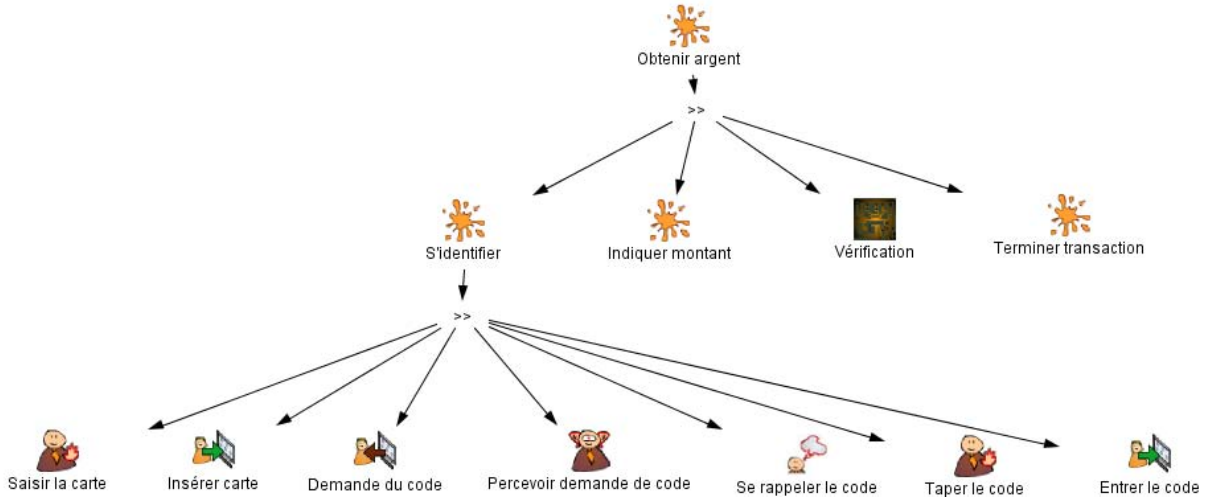


Figure 39. Exemple illustratif du distributeur de billets : raffinement de la tâche d'identification avec des sous-tâches interactives et utilisateur

Le type de tâche cognitive, même s'il est disponible dans certaines notations (décrites dans la section 2.2 du chapitre 2), n'est cependant jamais raffiné dans aucune notation. Or, il peut encore être raffiné pour permettre de déterminer l'allocation optimale de tâches et fonctions entre un utilisateur et le système à opérer. Les travaux de (Parasuraman, Sheridan, & Wickens, 2000) proposent un modèle humain simplifié du traitement de l'information afin de fournir un support aux choix d'allocation de tâches et fonctions entre un utilisateur et un système lors de la conception de systèmes informatiques. Ce modèle distingue deux étapes cognitives distinctes qui sont l'analyse et la décision. Cette distinction permet d'analyser les activités cognitives de l'utilisateur et de choisir, selon la criticité de l'activité, le contexte et la charge cognitive de l'utilisateur d'allouer une plusieurs activités au système.

HAMSTERS fournit les moyens de raffiner une tâche cognitive en deux sous-types :

- La **tâche d'analyse cognitive** (représentée par le symbole de gauche sur la Figure 40) permet de décrire que l'utilisateur analyse une situation.

- La **tâche de décision cognitive** (représentée par le symbole de droite sur la Figure 40) permet de décrire que l'utilisateur prend une décision en rapport avec une situation.

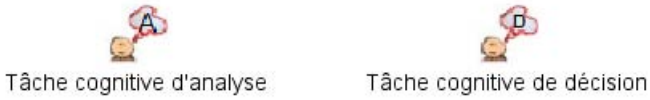


Figure 40. Possibilités de raffinement d'une tâche cognitive

La Figure 41 décrit le positionnement des sous-types de tâches utilisateurs par rapport aux quatre étapes du modèle humain simplifié du traitement de l'information de (Parasuraman, Sheridan, & Wickens, 2000).

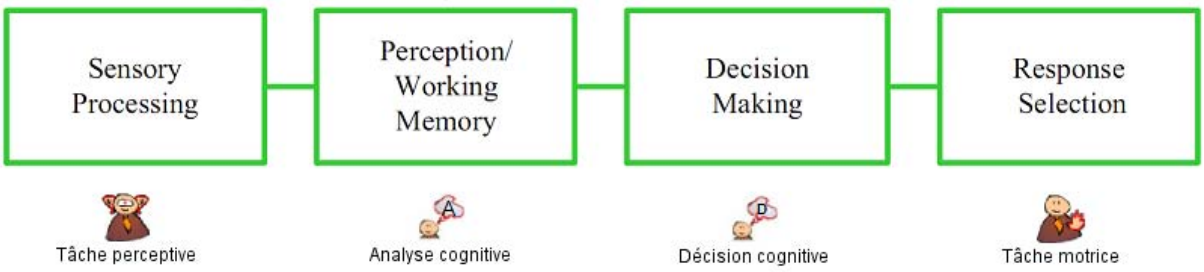


Figure 41. Répartition des sous-types de tâches utilisateur selon le modèle humain du traitement de l'information de Parasuraman

La Figure 42 présente une nouvelle version du modèle de tâches de l'exemple illustratif du retrait d'argent liquide. Dans cette nouvelle version, la tâche abstraite « S'identifier » a été repliée pour faciliter la lecture (symbole '+' en bas à droite de la tâche). La tâche « Indiquer montant » est détaillée en une séquence de tâches interactives et cognitives :

- La tâche interactive de sortie « Demande du montant souhaité » permet de décrire que le système attend l'information du montant de la part de l'utilisateur.
- La tâche perceptive « Percevoir demande de montant » permet de décrire que l'utilisateur doit percevoir que le système est en attente de l'information du montant souhaité de la part de l'utilisateur.
- La tâche cognitive d'analyse « Evaluation du montant nécessaire » permet de décrire que l'utilisateur doit analyser et évaluer le montant dont il a besoin.
- La tâche cognitive de décision « Choix de la valeur à entrer » permet de décrire que l'utilisateur doit effectuer un choix en fonction des différentes possibilités évaluées et de son but principal.
- La tâche motrice « Taper montant » permet de décrire que l'utilisateur doit effectuer une activité motrice pour entrer le montant souhaité.
- La tâche interactive d'entrée « Entrer montant » permet de décrire que l'utilisateur transmet désormais l'information issue de son choix au système.

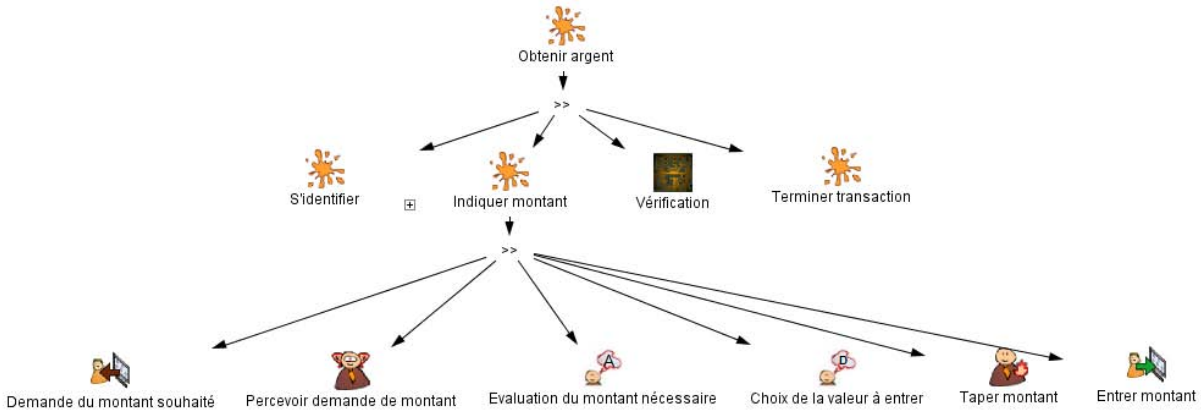


Figure 42. Exemple illustratif du distributeur de billets : raffinement de la tâche de demande du montant

Cette description fine des tâches utilisateur permet d'analyser l'adéquation entre les fonctions du système et les tâches utilisateur. Elle fournit un support à l'analyse et à la conception de systèmes interactifs partiellement automatisés. Cette contribution a fait l'objet d'une publication et est détaillée dans (Martinie, Palanque, Barboni, & Ragosta, 2011).

Le raffinement des tâches est nécessaire dans le cadre du développement d'un système interactif critique mais aussi dans le cadre du développement de son programme de formation. En effet, la description détaillée des tâches interactives et des tâches utilisateur fournit un support à plusieurs phases du développement du programme de formation :

- La phase d'analyse, ou étape « Analysis » dans le processus ADDIE (telle que décrite dans la section 2 du chapitre 3), nécessite un support à l'activité de description des tâches à accomplir dans le cadre de la mission de l'utilisateur opérant le système ainsi qu'un support à la sélection des tâches pour lesquelles l'utilisateur sera formé.
- La phase de conception, ou étape « Design » dans le processus ADDIE (telle que décrite dans la section 2 du chapitre 3), nécessite un support à l'activité de spécification des capacités et compétences pré requises pour effectuer la formation.

2 Structuration des modèles de tâches

Cette section présente les différents mécanismes proposés dans la notation HAMSTERS pour structurer les modèles de tâches de systèmes complexes. Dans la première sous-section nous expliquons les relations temporelles qualitatives mises en œuvre lors de la modélisation. Puis, dans la seconde sous-section, nous expliquons les relations temporelles quantitatives permettant de fournir un support à l'estimation et à l'évaluation des performances d'un utilisateur. Enfin, dans la troisième sous-section, nous expliquons les éléments de la notation permettant de factoriser la taille des modèles de tâches et ainsi de gérer la complexité des tâches.

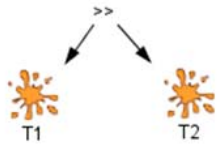
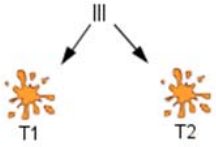
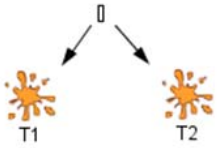
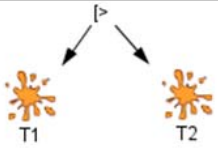
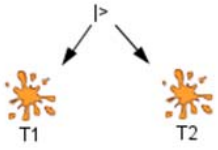
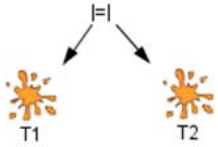
2.1 Relations temporelles qualitatives

Les relations temporelles qualitatives sont décrites grâce aux opérateurs ou structures de contrôle du langage informatique formel LOTOS (ISO, 1989). Les mécanismes de composition d'opérateurs ainsi que de description de l'optionalité et/ou la répétition d'une tâche permettent de faciliter l'ordonnancement des tâches.

2.1.1 Opérateurs d'ordonnancement temporel

La Table 10 présente l'ensemble des opérateurs permettant de décrire les relations temporelles qualitatives entre les tâches.

Table 10. Opérateurs d'ordonnancement temporel utilisés par la notation HAMSTERS

Type d'opérateur	Symbole	Utilisation
Enable	>>	
Concurrent	|||	
Choice	[]	
Disable	[>	
Suspend-resume	|>	
Order independent	|=|	

L'opérateur « Enable » permet de décrire que les tâches T1 et T2 se déroulent en séquence l'une après l'autre.

L'opérateur « Concurrent » permet de décrire que les tâches T1 et T2 peuvent se dérouler simultanément.

L'opérateur « Choice » permet de décrire que l'utilisateur effectuera la tâche T1 ou la tâche T2, mais que le choix d'une tâche entraînera la désactivation de l'autre.

L'opérateur « Disable » permet de décrire que le démarrage de la tâche T2 entraîne l'arrêt définitif de la tâche T2.

L'opérateur « Suspend-resume » permet de décrire que le démarrage de la tâche T2 entraîne l'arrêt temporaire de la tâche T1 ; la tâche T1 peut être redémarrée à tout moment puis interrompue à nouveau par la tâche T2, tant que la tâche T1 n'est pas terminée.

L'opérateur « Order independant » permet de décrire que l'utilisateur peut choisir s'il effectuera la tâche T1 ou la tâche T2 en premier. Cet opérateur indique aussi que la tâche choisie pour être exécutée en premier sera terminée avant de passer à la suivante.
La Table 10 représente la mise en œuvre des opérateurs pour deux tâches. La plupart des opérateurs peuvent être utilisés pour ordonnancer temporellement plus de deux tâches. Seuls les opérateurs « Disable » et « Suspend-resume » fournissent une relation temporelle qualitative entre deux sous-branches.

2.1.2 Composition d'opérateurs d'ordonnancement temporel

La notation HAMSTERS permet d'utiliser la composition d'opérateur. Cette technique fait partie des mécanismes de structuration permettant de réduire la taille des modèles.
La Figure 43 présente une nouvelle version du modèle de tâches de l'exemple illustratif du retrait d'argent liquide. Dans cette nouvelle version, la description de la fin de transaction a été détaillée en :

- Une tâche interactive de sortie permettant de décrire que le système est en attente d'une opération de la part de l'utilisateur.
- Un opérateur de choix indiquant deux sous-tâches possibles pour terminer la transaction :
    - L'utilisateur peut décider d'annuler sa transaction et retirer sa carte sans retirer d'argent.
    - L'utilisateur peut décider de confirmer son retrait, récupérer son argent et sa carte.

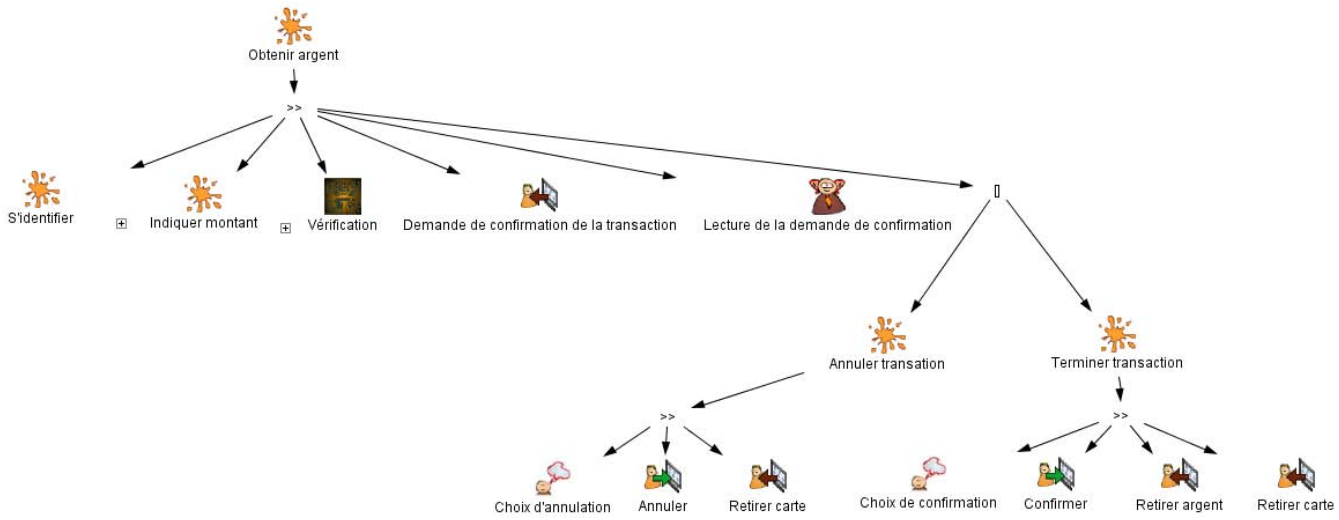


Figure 43. Exemple illustratif du distributeur de billets : composition d'opérateurs

Cette nouvelle version du modèle de tâche permet de montrer la composition d'opérateurs de différents types comme mécanisme de structuration des modèles de tâches.

2.1.3 Répétition et optionalité

La notation HAMSTERS permet de décrire qu'une tâche peut être optionnelle, itérative ou les deux simultanément, sachant que :

- Une tâche optionnelle est une tâche qui ne sera pas forcément exécutée, selon le contexte ou selon une décision arbitraire de la part de l'utilisateur.
- Une tâche itérative peut être exécutée une ou plusieurs fois jusqu'à ce qu'elle soit interrompue ou suspendue par une autre tâche.

Figure 44. Représentation graphique d'une tâche abstraite (optionnelle, itérative ou les deux simultanément)

La Figure 44 décrit la manière dont la nature itérative, optionnelle ou simultanément itérative et optionnelle d'une tâche abstraite et représentée. Même s'il est ici présenté sur un type de tâche abstraite, cet élément de notation s'applique à tous les types de tâches.

Les mécanismes de description de la répétition sont un premier moyen de structurer les modèles en évitant de décrire plusieurs fois en séquence une même tâche. Les mécanismes de description de l'optionalité d'une tâche permettent d'alléger la structure d'un modèle en évitant l'utilisation d'un opérateur de choix (« Choice »). La Figure 45 présente deux manières de décrire une tâche optionnelle, celle avec l'opérateur « Choice » et celle avec un seul opérateur et le symbole de l'optionalité apposé à la tâche concernée.

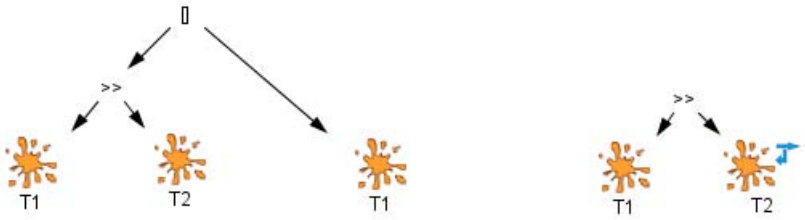


Figure 45. Deux descriptions possibles pour une tâche optionnelle

Les Figure 46 et Figure 47 présentent deux nouvelles versions du modèle de tâches de l'exemple illustratif du retrait d'argent liquide. Dans ces nouvelles versions, un exemple de modélisation de tâche itérative et un exemple de modélisation de tâche optionnelle sont exposés.

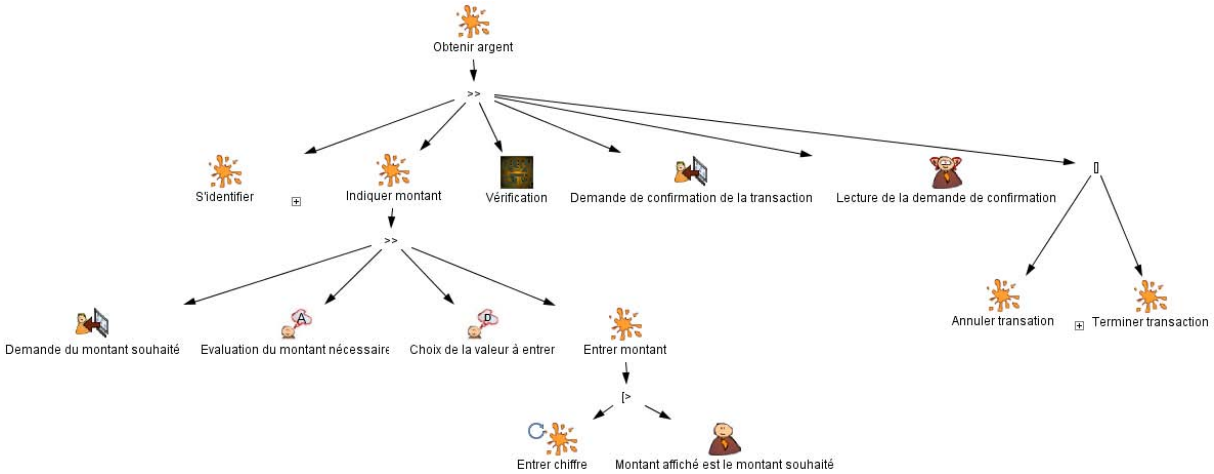


Figure 46. Exemple illustratif du distributeur de billets : Tâche itérative et opérateur d'interruption

La Figure 46 permet de montrer la description d'une tâche itérative interactive d'entrée de saisie d'un chiffre qui se répète tant que l'utilisateur n'a pas détecté que le montant saisi correspond au montant souhaité. L'association d'une tâche itérative à l'un des deux opérateurs « Disable » et « Suspend-resume » est un patron de modélisation fréquemment mis en œuvre (Paterno, 2004) car il permet de décrire une tâche se répétant un nombre de fois non déterminé tout en y association une condition d'arrêt.

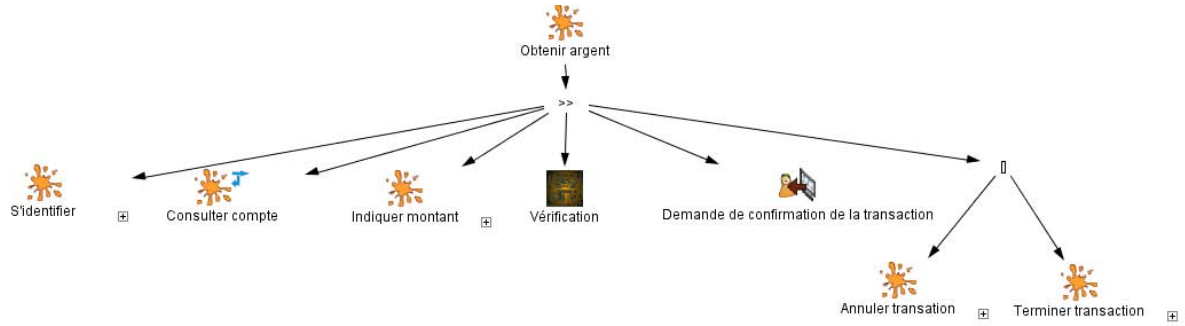


Figure 47. Exemple illustratif du distributeur de billets : Tâche optionnelle

La Figure 47 permet de montrer la description d'une tâche facultative parmi un ensemble de tâches ordonnées temporellement. La tâche abstraite « Consulter compte » sera effectuée ou non selon le choix de l'utilisateur.

2.2 Relations temporelles quantitatives

La notation HAMSTERS comprend les attributs temporels suivants, pouvant être associés à tous les types de tâches :

- Durée minimum d'exécution : une tâche peut s'effectuer en une durée minimale.
- Durée maximum d'exécution : une tâche peut s'effectuer en une durée maximale.

Ces attributs permettent de déterminer les temps minimum requis d'une tâche à l'autre, d'un sous-but à un autre sous-but et ainsi de vérifier leur compatibilité. Ils fournissent un support à :

- L'analyse de l'adéquation temporelle entre les actions de l'utilisateur et le traitement de l'information par le système.
- La validation du système conçu et l'évaluation des performances de l'utilisateur selon les scénarios d'utilisation.
- La spécification des objectifs de performance à remplir par les utilisateurs à l'issue du déroulement de la formation.
- L'évaluation des performances des opérateurs ayant suivi la formation pour leur qualification ou non qualification à l'opération du système.

2.3 Structuration de tâches complexes et nombreuses

Dans le cas de systèmes interactifs complexes, la modélisation des tâches utilisateur nécessite des moyens supplémentaires (cette problématique a été décrite dans la section 2.2 du chapitre 2).

La notation HAMSTERS fournit les éléments de notation *subroutine* et *copy task*. Ces termes font référence aux éléments de programmation structurée employés dans par un grand nombre de langages informatiques. Ils fournissent un support à modularité et à la réutilisabilité des modèles de tâche.

Le mécanisme de *copy task* (inspiré d'un mécanisme de programmation du langage COBOL) permet de décrire une tâche et d'utiliser cette description plusieurs fois dans un même modèle et dans différents modèles. Afin de signifier qu'une ou plusieurs tâches sont les mêmes qu'une autre, la notation HAMSTERS utilise la représentation visuelle présentée dans la Table 11, selon chaque type de tâche.

Table 11. Représentation des symboles associés aux tâches utilisant le mécanisme *copy* task.

Types de tâche	Représentation du mécanisme *copy task*
Tâche abstraite	 Tâche abstraite copy
Tâche (fonction) système	 Tâche système copy
Tâches utilisateur	 Tâche utilisateur copy  Tâche perceptive copy  Tâche motrice copy  Tâche cognitive copy  Tâche d'analyse copy  Tâche de décision copy
Tâches interactives	 Tâche d'interaction copy  Tâche interactive d'entrée copy  Tâche interactive de sortie copy  Tâche interactive d'entrée/sortie copy

Le mécanisme de *subroutine* permet de définir un ensemble d'activités (décomposées hiérarchiquement et ordonnées temporellement) qu'un utilisateur effectue plusieurs fois et parfois dans différents contextes, i.e. avec différents flux de données. Les éléments suivant permettent de caractériser une *subroutine* :

- Son nom.
- Une représentation visuelle spécifique qui est l'icône des tâches abstraites (en effet, les sous-tâches lui appartenant peuvent être de différents types).
- Des ports d'entrée et sortie spécifiques permettant de symboliser la nécessité ou non de flux de données entrants et/ou sortant.

La Table 12 décrit les différents types de *subroutines* selon leur capacité à recevoir et fournir des paramètres d'entrée/sortie.

Table 12. Types de *subroutines* dans la notation HAMSTERS

Type de subroutine	Représentation avec la notation HAMSTERS
Subroutine sans aucun flux de données entrant et sortant	
Subroutine nécessitant un (ou plusieurs) flux de données entrant(s)	
Subroutine fournissant un (ou plusieurs) flux de données sortant(s)	
Subroutine nécessitant et fournissant un (ou plusieurs) flux de données entrant(s) et sortant(s).	

Une *subroutine* peut être utilisée une ou plusieurs fois, ceci dans un ou plusieurs modèles de tâches. Une *subroutine* sans aucun flux entrant et/ou sortant s'apparente à une tâche abstraite utilisant le mécanisme de *copy task*.

L'exemple illustratif du retrait d'argent n'étant pas une tâche complexe, un exemple illustratif de modélisation de différents types de *subroutines* est proposé dans l'étude de cas exposée au chapitre 7 sur la supervision d'une mission satellitaire. La sous-section 3.3 de ce chapitre détaille l'utilisation de ce mécanisme avec les flux de données. Les mécanismes de structuration *copy task* et *subroutine* ont fait l'objet d'une publication qui détaille son utilisation (Martinie, Palanque, & Winckler, 2011).

3 Représentation des données et de leur traitement

Le concept de données ou d'objets permet de décrire les éléments externes ou internes à l'utilisateur et qu'il manipule pour accomplir un but. Ainsi, il permet de représenter dans les modèles de tâches :

- Un objet physique passant d'un utilisateur à un autre ou d'un utilisateur à un système.
- Un objet logique ou une connaissance nécessaire à l'accomplissement d'une tâche, qu'elle soit interne ou externe à l'utilisateur (« Knowledge in the head » versus « Knowledge in the world », (Norman D. A., 2002)).

Dans la suite de cette section, nous distinguons la représentation du flux de données de la représentation des objets, puis détaillons leur influence sur la structure de contrôle des modèles de tâches.

3.1 Flux de données

Afin de décrire les flux de données, HAMSTERS contient plusieurs éléments de notation (représentés sur la Figure 48) :

- Un port de sortie sur la droite de la tâche abstraite T1 (rectangle orange) permet de décrire que la tâche peut fournir un flux de données sortant.
- Un port d'entrée sur la gauche de la tâche abstraite T2 (rectangle orange) permet de décrire que la tâche peut réceptionner un flux de données entrant.
- Une flèche entre le port de sortie de T1 et le port d'entrée de T2 permet de décrire que la tâche T1 fait effectivement passer un flux de données à la tâche T2.

Figure 48. Représentation d'un flux d'information entre deux tâches

La Figure 48 utilise le type de tâche abstraite pour l'exemple mais la description d'un flux d'information est possible entre chaque type de tâches.

3.2 Objets

Afin de décrire les objets et leurs relations avec les tâches et les flux de données, HAMSTERS contient plusieurs éléments de notation (représentés sur la Figure 49) :

- Des flèches uni ou bidirectionnelles entre une tâche et un ou plusieurs objets/données (schéma de gauche sur la Figure 49) permettent de décrire si :
  - la valeur d'un/une objet/donnée est nécessaire à l'accomplissement d'une tâche (la tâche T1 nécessite la valeur de l'objet « objet1 »).
  - la valeur d'un/une objet/donnée est modifiée par l'accomplissement d'une tâche (la tâche T1 modifie la valeur de l'objet « objet2 »).
  - la valeur d'un/une objet/donnée est nécessaire et modifiée lors de l'accomplissement d'une tâche (la tâche T1 nécessite la valeur de l'objet « objet3 » et la modifie).
- Une flèche (ou plusieurs selon le nombre d'objets) entre un/une objet/donnée et la représentation d'un flux d'information permettant de décrire le/la (ou les) objet/donnée(s) transitant par ce flux (l'objet « objet4 » est transmis à la tâche T3 à la fin de l'exécution de la tâche T2).

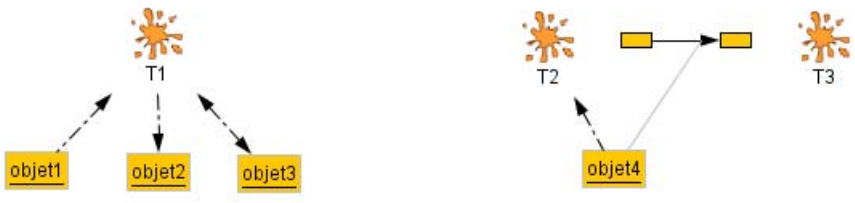


Figure 49. Représentation des relations entre objets, tâches et flux d'information

La Figure 49 utilise le type de tâche abstraite pour l'exemple mais la description des relations entre objets, tâches et d'information est possible pour chaque type de tâches. De plus, pour une même tâche, autant de ports d'entrée et sorties sont nécessaires que de flux de données.

La Figure 50 présente une nouvelle version du modèle de tâches de l'exemple illustratif du retrait d'argent liquide. Dans cette nouvelle version, la donnée « code » est nécessaire à l'utilisateur (flèche allant de la représentation de la donnée vers la tâche « Se rappeler le code ») et va transiter par les flux d'information entre la tâche cognitive « Se rappeler le code » et la tâche motrice « Taper le code », puis entre la tâche motrice « Taper le code » et la tâche interactive d'entrée « Entrer le code ».

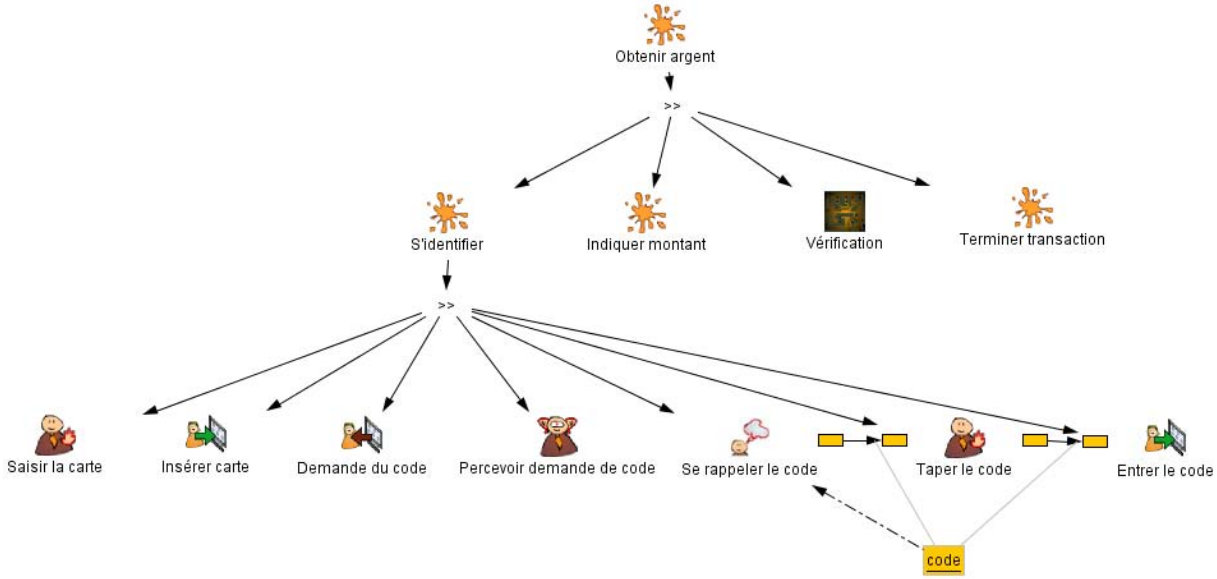


Figure 50. Exemple illustratif du distributeur de billets : saisie du code d'identification

3.3 Conditions et assignations

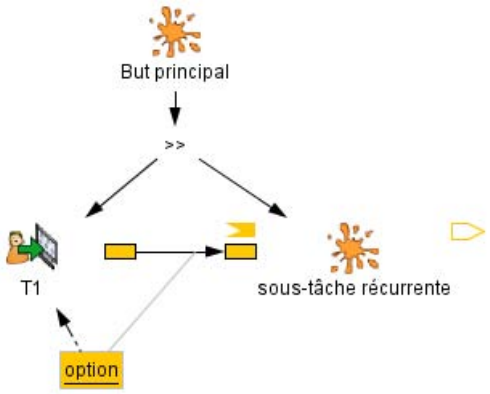
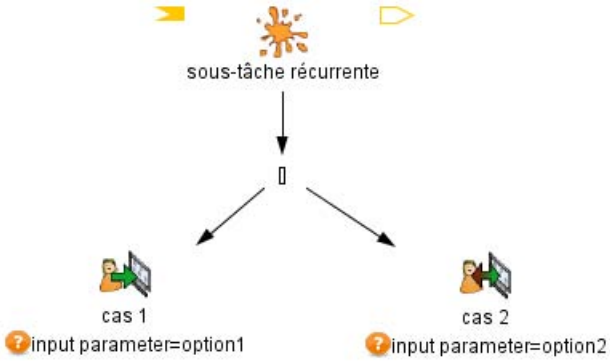
Une condition (« **Condition** »), représentée par le symbole , et associée à une tâche, permet de décrire (comme son nom l'indique) sous quelle condition, portant sur une donnée ou un objet, la tâche peut être effectuée.

Une assignation (« **Assignment** »), représentée par le symbole , et associé à une tâche, permet de décrire le changement de valeur d'un objet à la suite de l'exécution d'une tâche et ainsi fournir un moyen de conditionner l'exécution d'une tâche suivante.

Ces mécanismes permettent d'influencer la structure de contrôle des modèles de tâches et de configurer la séquence de tâches effectuées selon le contexte et la valeur des objets/données. Des exemples de configuration sont proposés dans les tableaux suivants ou ces mécanismes sont associés aux mécanismes de *subroutines*.

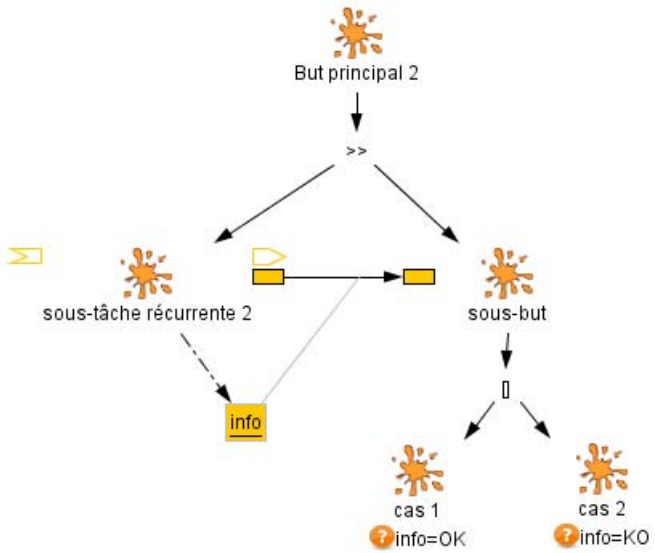
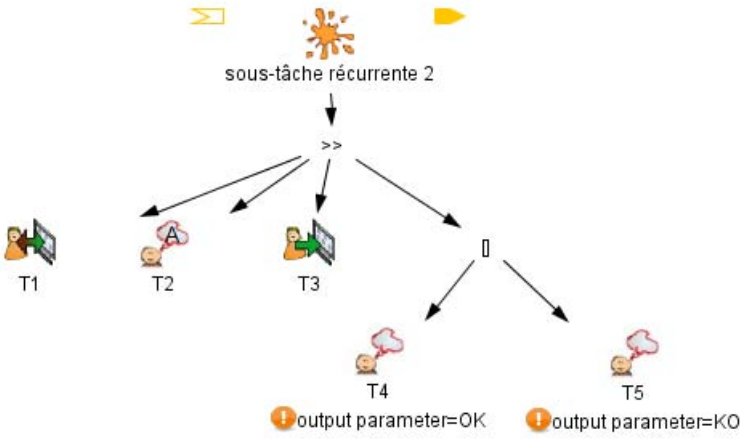
La Table 13 décrit la représentation d'un modèle de tâche appelé modèle principal utilisant une référence à une *subroutine* (modèle du haut) et la représentation de la *subroutine* employée par le modèle de tâche principal (modèle du bas). Le modèle de tâche principal contient la représentation du flux de données entre une tâche interactive d'entrée du modèle de tâche principal et la *subroutine* nécessitant un flux de données entrant (paramètre d'entrée représenté par le symbole). La valeur de la donnée « option » conditionne l'exécution d'une des branches de la *subroutine*.

Table 13. Représentation d'un modèle de tâche et d'une subroutine nécessitant un flux d'information entrant

Exemple de modèle principal avec les éléments de notation de subroutine et de flux de données	
Exemple de subroutine utilisée par le modèle principal avec les éléments de notation de description des conditions	

La Table 14 décrit la représentation d'un modèle de tâche appelé modèle principal utilisant une référence à une *subroutine* (modèle du haut) et la représentation d'une *subroutine* employée par le modèle de tâches principal (modèle du bas). Le modèle de tâches principal contient la représentation du flux de données entre la *subroutine* fournissant un flux de données sortant (paramètre de sortie représenté par le symbole) et une tâche abstraite du modèle de tâche principal. La valeur de la donnée en provenance de la *subroutine* conditionne l'exécution d'une des branches de la tâche abstraite « sous-but » du modèle principal.

Table 14. Représentation d'un modèle de tâche et d'une subroutine fournissant un flux d'information sortant

Exemple de modèle principal avec les éléments de notation de description de subroutine, de flux de données et de condition	
Exemple de subroutine utilisée par le modèle principale avec les éléments de notation de description d'assignation	

Tous ces mécanismes sont utilisés dans l'étude de cas du Chapitre 1.

4 Erreurs humaines

Les modèles de tâches permettent de décrire les différentes activités que l'utilisateur accomplit avec le système pour mener à bien sa mission. Cependant, la modélisation des tâches peut aussi être utilisée pour analyser les déviations possibles du comportement de l'utilisateur pendant ses activités et ainsi prévenir les fautes d'utilisation au moment de la conception du système.

Les approches de conception et de développement de systèmes critiques prennent en compte les problématiques de fautes et d'erreurs humaines (ces problématiques sont décrites dans la section 2 du chapitre 1 et les approches de développement sont décrites dans la section 1 du chapitre 2). Les travaux de (Palanque & Basnyat, 2004) proposent l'utilisation des techniques de modélisation des tâches pour analyser systématiquement les erreurs d'utilisation potentielles.

Les éléments de la notation HAMSTERS présentés dans les sections précédentes fournissent un support à :

- La description des erreurs et fautes humaines.
- L'analyse des déviations possibles lors de l'utilisation du système.
- La mise en place de barrières pour prévenir les fautes.

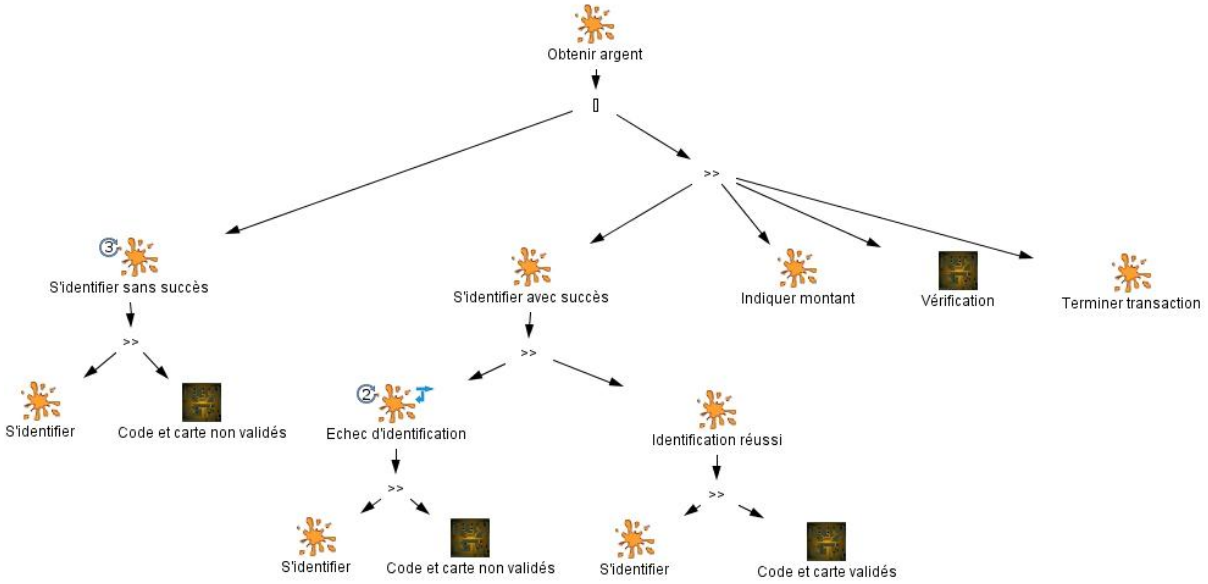


Figure 51. Exemple illustratif du distributeur de billets : modélisation des erreurs

La Figure 51 présente une nouvelle version du modèle de tâches de l'exemple illustratif du retrait d'argent liquide. Dans cette nouvelle version, le modèle de tâche identifie les déviations possibles lors de l'utilisation du distributeur de billets. Les erreurs humaines et l'utilisation malveillante du système sont prises en compte. Ce modèle décrit l'échec possible de l'opération d'identification La partie gauche du modèle de tâches, sous la branche « S'identifier sans succès » décrit que si l'utilisateur essaie de s'identifier trois fois sans y parvenir, le système ne donnera pas suite à la transaction. La partie centrale du modèle de tâches, sous la branche « S'identifier avec succès » décrit que l'utilisateur peut potentiellement échouer deux fois puis réussir à s'identifier ou peut réussi à s'identifier dès la première tentative. Ce modèle de tâche décrit aussi la barrière de sûreté mise en place pour éviter une usurpation d'identité et l'utilisation frauduleuse d'une carte : le nombre de tentatives d'identification n'est pas infini mais limité à trois.

5 Méta-modèle de la notation HAMSTERS

Les travaux de (Limbourg & Vanderdonckt, 2004) montrent que l'établissement du méta-modèle d'une notation permet de mieux comprendre sa nature et ses spécificités. La Figure 52 présente le diagramme entité association de la notation HAMSTERS.

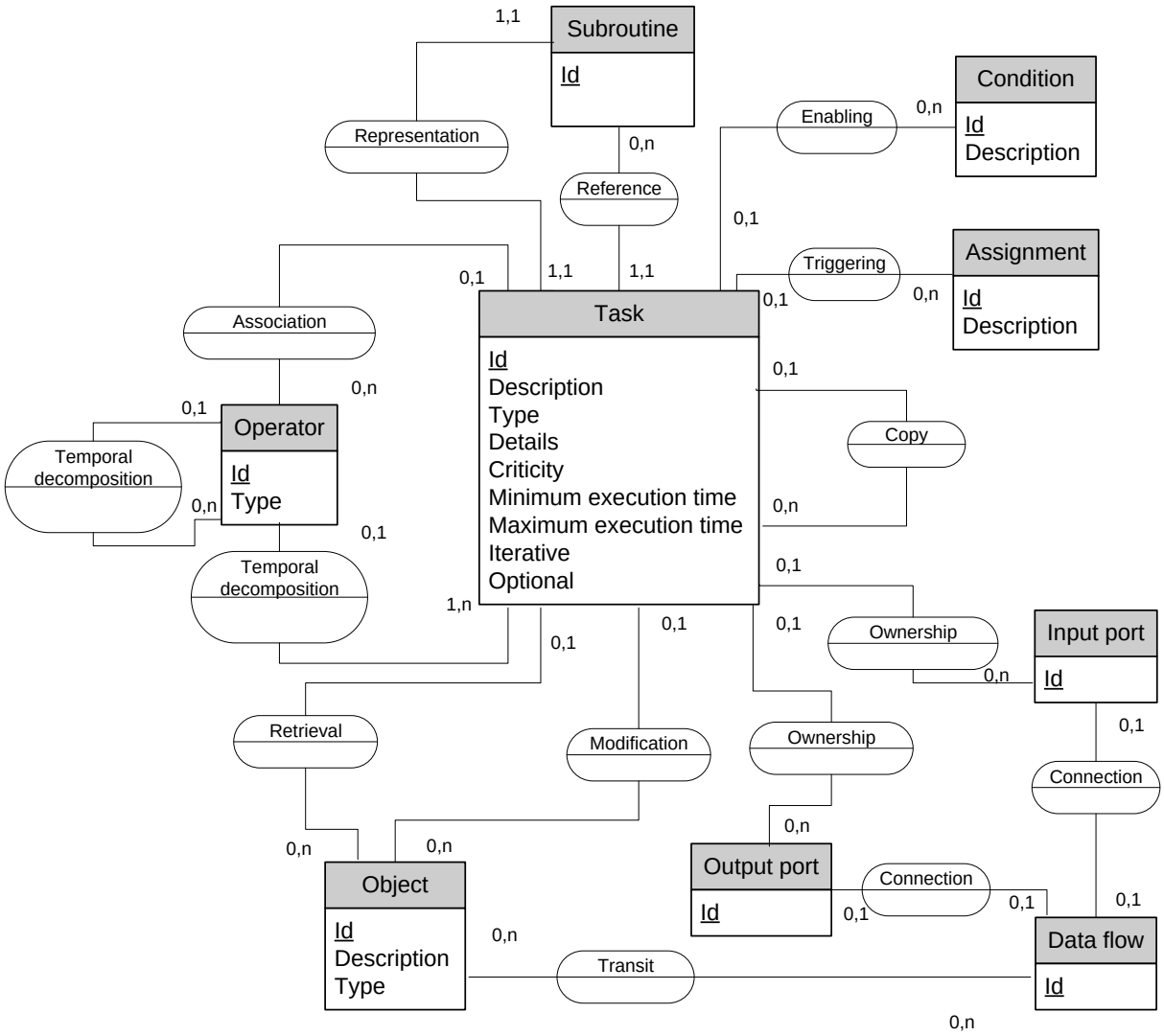


Figure 52. Diagramme Entité Association de la notation HAMSTERS

La Figure 52 permet de montrer précisément les relations entre les différents éléments de notation décrits dans les paragraphes précédents :

- Une tâche n'est associée à aucun ou est associée à un ou plusieurs opérateurs d'ordonnancement temporel.
- Un opérateur d'ordonnancement temporel est associé la décomposition temporelle d'une ou plusieurs tâches et n'est associé à aucun ou est associé à un ou plusieurs opérateurs d'ordonnancement temporel (ce dernier point permet la factorisation d'opérateurs d'ordonnancement temporel).
- Une tâche peut nécessiter la consultation ou modifier la valeur d'aucun, un ou plusieurs objets.
- Un port d'entrée peut appartenir à une tâche et une tâche peut posséder aucun, un ou plusieurs ports d'entrée.
- Un port de sortie peut appartenir à une tâche et une tâche peut posséder aucun, un ou plusieurs ports de sortie.
- Un flux de données est lié à un et un seul port de sortie et à un et un seul port d'entrée.
- Un objet est lié à un ou plusieurs flux d'information et un flux d'information peut permettre à un ou plusieurs objets de transiter entre un port d'entrée et un port de sortie.

- Aucune ou plusieurs tâches peuvent être la copie (*copy task*) d'aucune ou d'une et une seule tâche.
- Une tâche peut être une référence à aucune ou à une et une seule *subroutine* et une *subroutine* peut être référencée par aucune, une ou plusieurs tâches.
- Une *subroutine* peut être représentée par une et une seule tâche.
- Une tâche peut être effectuée sous aucune, une ou plusieurs conditions portant sur un ou plusieurs paramètres d'entrée d'une *subroutine*.
- Une tâche effectuée peut mener à l'assignation d'une valeur à un ou plusieurs paramètres de sortie d'une *subroutine*.

Par rapport aux méta-modèles des notations présentés par dans les travaux de (Limbourg Q. , Vanderdonckt, Michotter, Bouillon, & Lopez-Jaquero, 2004), la représentation du méta-modèle de HAMSTERS permet de mettre en évidence les nouvelles possibilités de modélisation :

- Les éléments de la notation concernant les flux de données (« Input port », « Output port », « Data flow »). fournissent la possibilité de décrire les flux de données de manière plus fine que les autres notations.
- Les éléments de la notation appelés « Copy task » et « Subroutine ». Ils influencent la modularité et la réutilisabilité des modèles de tâches en fournissant la possibilité pour une tâche de faire référence à une autre tâche (*copy task*) ou à un autre modèle de tâche (*subroutine*).
- Les relations « Association » et « Temporal decomposition » entre les éléments de notation « Task » et « Operator ». Ils influencent l'association entres les opérateurs et les tâches ainsi qu'entre les opérateurs eux-mêmes et fournissent un support la composition d'opérateurs.

Par rapport aux méta-modèles des notations présentées dans les travaux de (Limbourg & Vanderdonckt, 2004), il est important de remarquer que la notation HAMSTERS ne fournit cependant pas d'éléments de notation permettant la description des rôles des utilisateurs. Une extension de la notation est envisagée et discutée comme perspective à cette thèse.

6 Outil logiciel d'édition et de simulation HAMSTERS

L'outil logiciel HAMSTERS permet d'éditer et de simuler des modèles de tâches en utilisant les différents éléments de la notation HAMSTERS présentés dans les sections précédentes de ce chapitre.

La Figure 37 décrit les sept étapes de la théorie de l'action de Norman. A la manière dont le concepteur d'un système interactif utilise cette représentation pour analyser l'activité de l'utilisateur, cette représentation a été utilisée pour concevoir l'outil d'édition et de simulation de modèles de tâches HAMSTERS afin qu'il soit centré utilisateur. Une telle démarche a déjà été effectuée pour concevoir l'outil d'édition de réseaux de Petri PetShop afin qu'il soit centré utilisateur et est présentée dans les travaux de (Barboni, Bastide, Lacaze, Navarre, & Palanque, 2003). L'outil de modélisation des tâches doit faciliter la perception et l'évaluation des tâches que le concepteur modélise. Ainsi, les différents éléments de notation ont été choisis pour faciliter la perception et l'évaluation des tâches modélisées. La fonctionnalité de simulation des modèles de tâches permet aussi de contribuer à l'évaluation d'un modèle édité et à sa validation.

La Figure 53 présente une copie d'écran de l'environnement logiciel HAMSTERS. Elle montre la perspective d'édition de modèles de tâche, comprenant :

- Un navigateur de projets avec ses modèles de tâches et scénarios associés (pastille 1).

- Un cadre central permettant d'éditer et consulter plusieurs modèles de tâches à la fois (pastille 2).
- Une palette d'édition permettant de glisser/déposer les éléments de la notation (pastille 3).
- Une fenêtre d'édition des propriétés liées à un élément de la notation (pastille 4).
- Un navigateur de structure du modèle de tâche couramment édité (pastille 5).

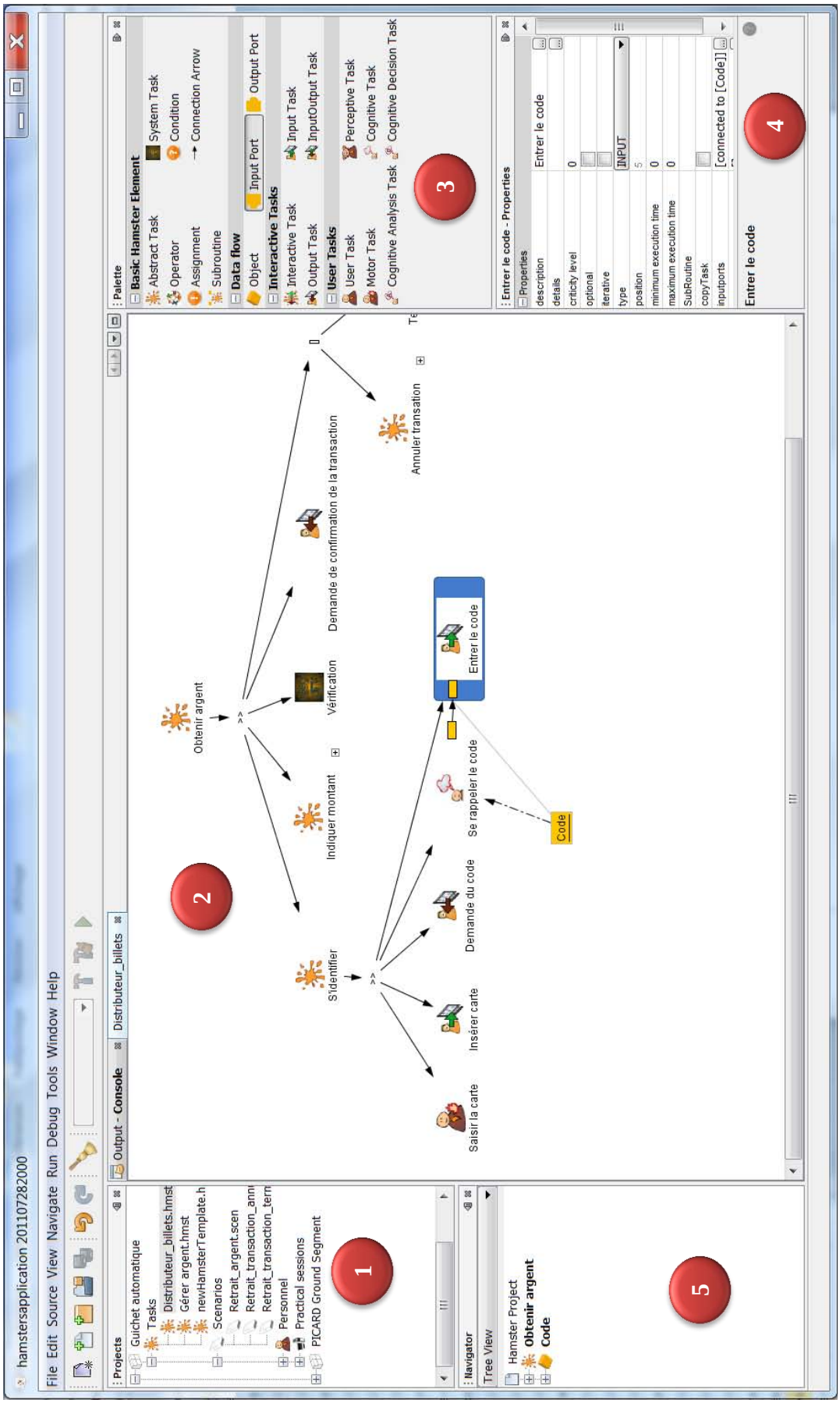


Figure 53. Copie d'écran de l'environnement logiciel HAMSTERS

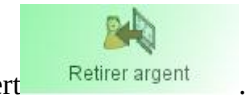
L'outil logiciel HAMSTERS permet de déplier/replier des sous arbres de tâches pour faciliter la visualisation, pour ce faire le symbole '+' est apposé en bas à droite d'une tâche pliée (Figure 54).

Figure 54. Représentation graphique d'une tâche repliée

L'outil logiciel HAMSTERS fournit une fenêtre graphique de simulation permettant de contrôler et commander l'exécution des tâches.

La Figure 55 montre la perspective de simulation de l'environnement HAMSTERS qui n'est pas très éloignée visuellement de la perspective d'édition. En effet, seuls les panneaux d'édition de droite ont été remplacés par un panneau de contrôle de la simulation contenant la liste des tâches pouvant être effectuées (liste dans la partie supérieure, pastille 1) et la liste des tâches déjà effectuées dans leur ordre d'exécution avec le modèle auquel elles appartiennent (liste dans la partie inférieure, pastille 3). La zone centrale (pastille 2) permet au concepteur de saisir des données contextuelles sur l'exécution d'une tâche. Par exemple, dans le cas de cette copie d'écran, le concepteur peut saisir la valeur de la donnée associée à l'exécution de la tâche. Cette zone permet aussi au concepteur d'indiquer si une tâche utilisateur s'est déroulée correctement. Cette fonctionnalité permet de fournir un support à l'analyse des erreurs humaines aussi lors de l'exécution de scénarios d'utilisation.

Dans la partie centrale de la copie d'écran, celle contenant le modèle de tâches, de nouvelles signalétiques permettent au concepteur de percevoir rapidement :

- les tâches déjà effectuées, avec l'apposition sur leur droite du symbole suivant .
- les tâches disponibles, surlignées en vert 

.

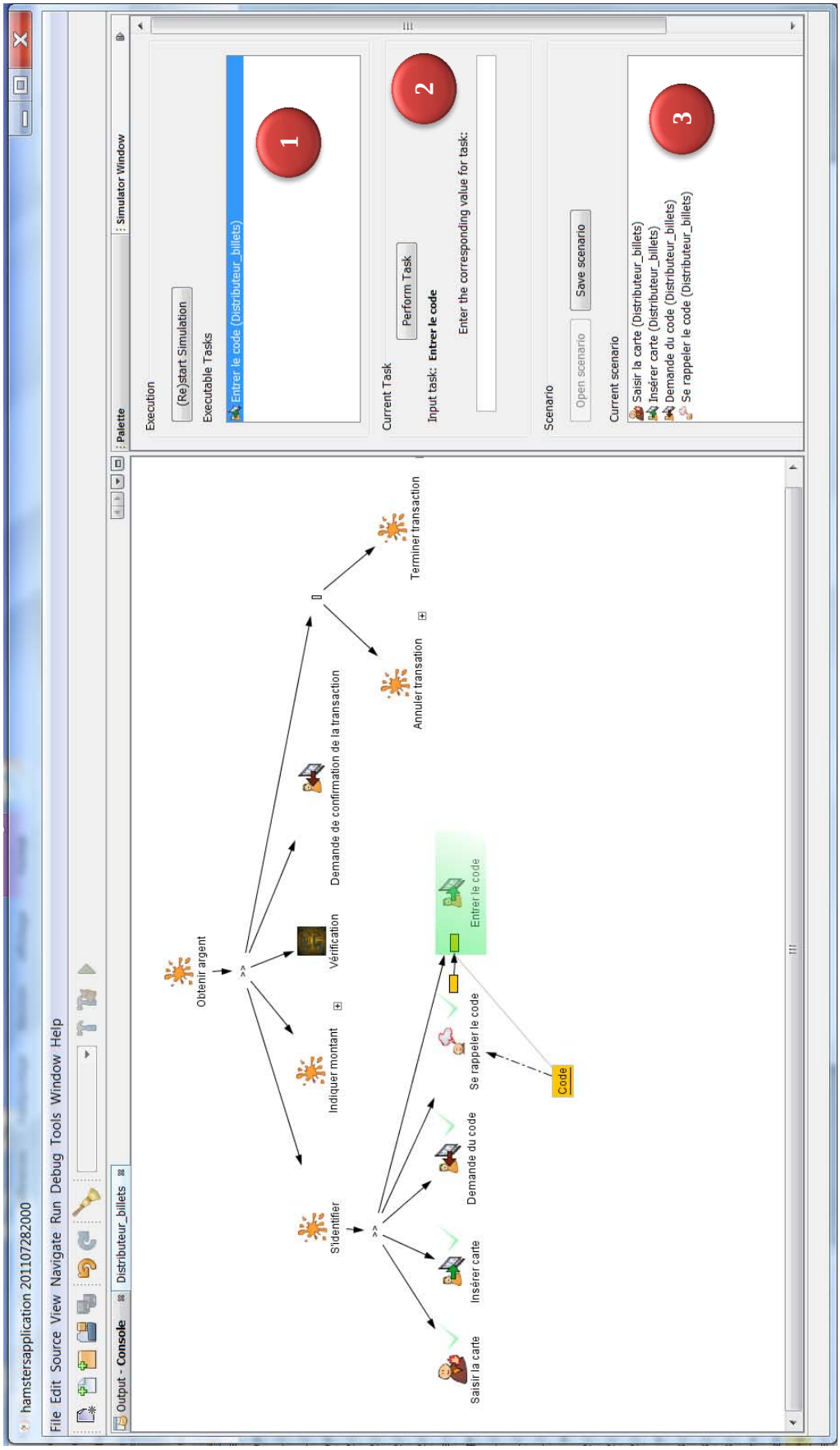


Figure 55. Copie d'écran d'une simulation de modèles de tâches avec l'outil logiciel HAMSTERS (exemple de tâches effectuées et actives)

La Figure 56 montre la perspective de simulation de l'outil logiciel HAMSTERS où l'état d'exécution du modèle montre qu'un sous arbre du modèle a été désactivé par un choix d'exécution.

Dans la partie centrale de la copie d'écran, celle contenant le modèle de tâches, de nouvelles signalétiques permettent au concepteur de percevoir rapidement :

- les tâches ayant été désactivées par des choix effectués antérieurement par l'utilisateur, surlignées en rouge 

.

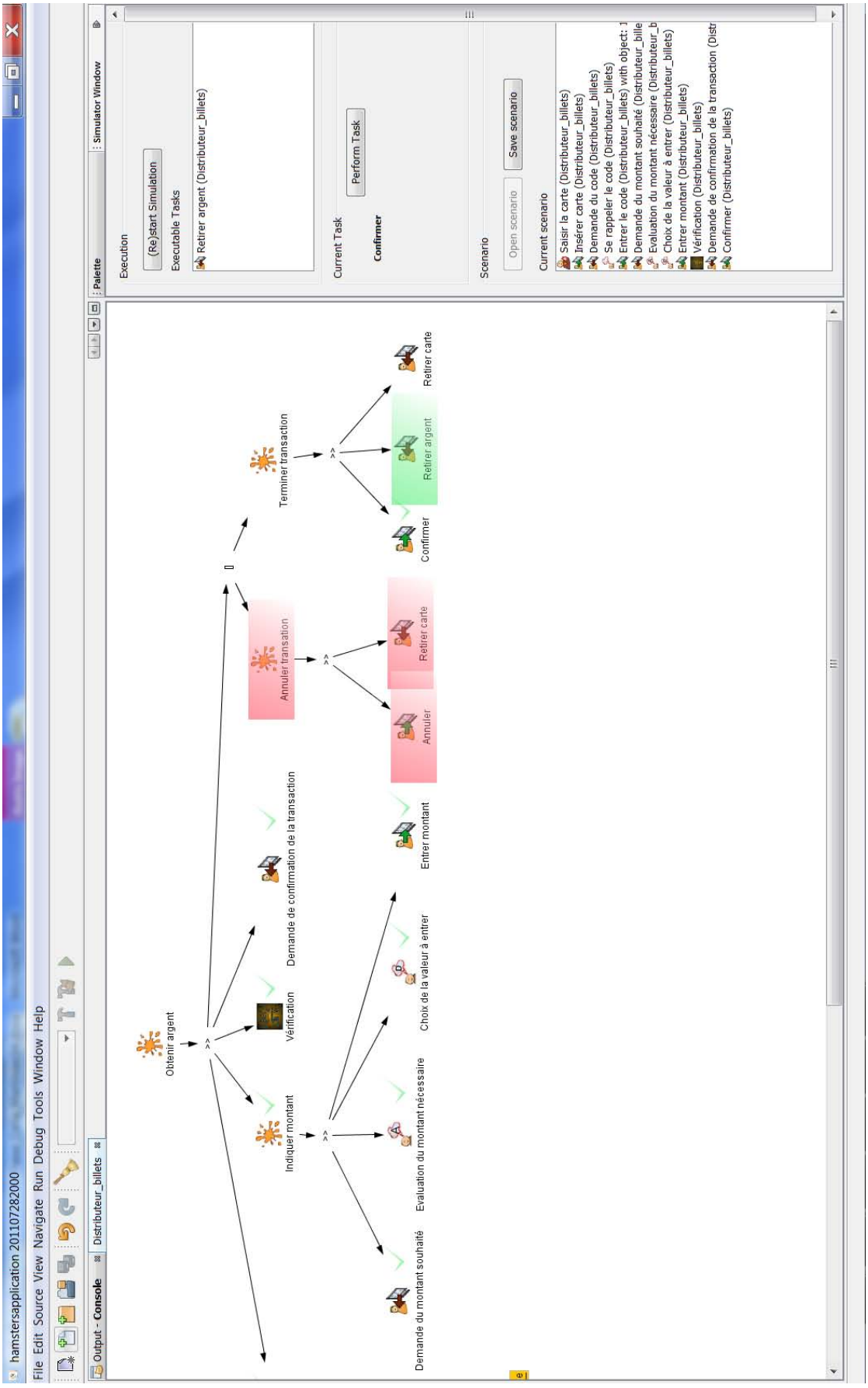


Figure 56. Copie d'écran d'une simulation de modèles de tâches avec l'outil logiciel HAMSTERS (exemple de tâches désactivées)

7 Conclusion

Ce chapitre nous a permis de montrer comment la notation HAMSTERS fournit un support à la conception de systèmes interactifs complexes et de systèmes interactifs critiques, en satisfaisant chaque point de la liste des besoins établie dans la section 2 du chapitre 2. HAMSTERS est une notation outillée permettant de :

- Structurer et simuler une grande quantité d'activités utilisateur liées à un système informatique.
- Analyser l'adéquation entre les tâches utilisateur et les fonctions du système.
- Spécifier les tâches utilisateurs nécessaires au métier de l'opérateur du système et sélectionner les tâches pertinentes pour le programme de formation.
- Prévoir, évaluer et analyser les performances de l'utilisateur opérant un système interactif.
- Analyser les erreurs et fautes humaines potentielles afin d'établir des barrières lors de la conception du système.

De plus, nous avons montré que cette notation outillée est actuellement la seule à fournir un support à la conception de systèmes interactifs complexes et de systèmes interactifs critiques.

Cette notation outillée est nécessaire à la mise en œuvre de l'approche de conception et développement proposée par cette thèse dans le chapitre 5. La mise en œuvre de cette notation est décrite dans le chapitre 6 ainsi que dans l'étude de cas proposée au chapitre 7.

Chapitre 5- Une approche de développement pour les systèmes interactifs critiques

Dans ce chapitre, nous présentons une des contributions apportées par cette thèse, à savoir une approche de développement pour les systèmes interactifs critiques basée sur un processus et des techniques de mise en œuvre (ces dernières sont décrites de manière théorique dans ce chapitre). L'originalité de ce processus de développement est qu'il comprend une phase intégrée de développement d'un programme de formation, alors que, comme détaillé dans l'état des connaissances scientifiques au chapitre 2, le développement d'un programme de formation est généralement effectué séparément du développement du système interactif critique et lorsqu'il est déjà déployé.

Tout d'abord, nous présentons une vue d'ensemble du processus de développement (exposée dans la section 1) puis détaillons en particulier les phases du processus impactées par notre contribution :

- Conception du système interactif critique (phase décrite dans la section 2)
- Développement du programme de formation associé (phase décrite dans la section 3)
- Traçabilité des choix et exigences tout au long du processus (phase décrite dans la section 4)

Enfin, la dernière section est consacrée à une discussion sur les bénéfices et limitations de cette approche.

1 Présentation générale du processus de développement

Cette section présente une vue d'ensemble du processus de développement proposé par cette thèse. La première partie de cette section (sous-section 1.1) propose une définition des mots clés de la terminologie utilisée pour décrire ce processus. Ces mots clés sont régulièrement utilisés dans la suite du mémoire. La seconde partie de cette section (sous-section 1.2) survole le processus de développement.

1.1 Terminologie utilisée pour décrire le processus de développement

Nous utiliserons le terme générique *système interactif critique* ou *système* pour désigner la partie interactive du système à développer (pouvant être un ensemble ou un sous-ensemble des éléments suivants : interface utilisateur, application logicielle, objet matériel utilisé pour interagir avec le système, pilotes de périphériques d'interaction avec l'utilisateur).

Le terme *phase* employé dans de chapitre fait référence à un ensemble distinct d'étapes visant à :

- Produire des éléments importants pour le système interactif critique.
- Déclencher une autre phase ou contribuer à une autre phase.

Le terme *modélisation* employé dans ce chapitre est générique et ne fait pas référence à une notation particulière. Ce terme peut être utilisé pour décrire :

- L'*activité de modélisation*. Cette activité permet de filtrer un certain nombre d'éléments du sujet que l'on veut décrire (flux, comportement, architecture, opérations,…) et de mettre en valeur les éléments nécessaires pour certaines étapes du processus de développement. Ainsi, à chaque étape, un ou plusieurs types de modélisation différents peuvent être utiles.

- Les *modèles produits*. Ce sont des représentations issues de l'activité de modélisation. Ils peuvent être nécessaires en entrée d'une étape et/ou d'une phase de développement. Ils peuvent être produits en sortie d'une étape et/ou d'une phase de développement.

1.2 Survol du processus de développement

Dans cette section, nous allons proposer une vision abstraite du processus de développement qui a été défini dans le cadre de cette thèse. Cette vision abstraite donne une vue d'ensemble sur le processus et permet d'en comprendre le cheminement sans en connaître les détails.

Le processus de développement de systèmes interactifs critiques proposé par cette thèse intègre en particulier les phases (issues des domaines de l'IHM et des systèmes critiques) suivantes :

- Analyse et modélisation des tâches utilisateurs comme référence dans les différentes phases du processus et entre les différentes phases du processus.
- Modélisation formelle et prototypage très haute-fidélité du système.
- Evaluation quantitative des performances de l'utilisateur.
- Développement d'un programme de formation à l'utilisation d'un système critique.
- Traçabilité des exigences et besoins tout au long du processus de conception.

Les trois premières phases sont principalement issues du domaine de l'IHM. La quatrième phase et la cinquième phase sont issues du domaine des systèmes critiques.

La phase de développement d'un programme de formation permet de s'assurer qu'un utilisateur sera compétent pour exploiter un système critique. Cette phase ne fait généralement pas partie des processus de développement de systèmes interactifs critiques. Les programmes de formation, dans le cas où le personnel doit être qualifié au même moment où le système est opérationnel, est généralement conçu et mis en œuvre à partir de données et simulateurs de systèmes existants (le cas échéant). Sinon, les programmes de formation sont conçus et mis en œuvre, dans le meilleur des cas quand les contraintes temporelles ne l'empêchent pas, à partir du système nouvellement développé et déployé en s'appuyant sur : un manuel utilisateur (le cas échéant), un simulateur (le cas échéant), les spécifications du système. Intégrer la phase de conception du programme de formation au processus de développement du système interactif critique permet de concevoir un programme de formation avant le déploiement du système basé sur :

- L'analyse des activités de l'utilisateur correspondant au système développé.
- La spécification formelle du système conçu.
- Le prototype très haute-fidélité du système conçu.

La phase de traçabilité des exigences par rapport au choix de conception permet de structurer et mettre en évidence les choix effectués ainsi que la mesure dans laquelle ils satisfont aux exigences et besoins pour le développement du système interactif critique. Ces informations peuvent être utiles pour la compréhension des choix effectués et sont nécessaires pour la certification dans certains domaines d'application.

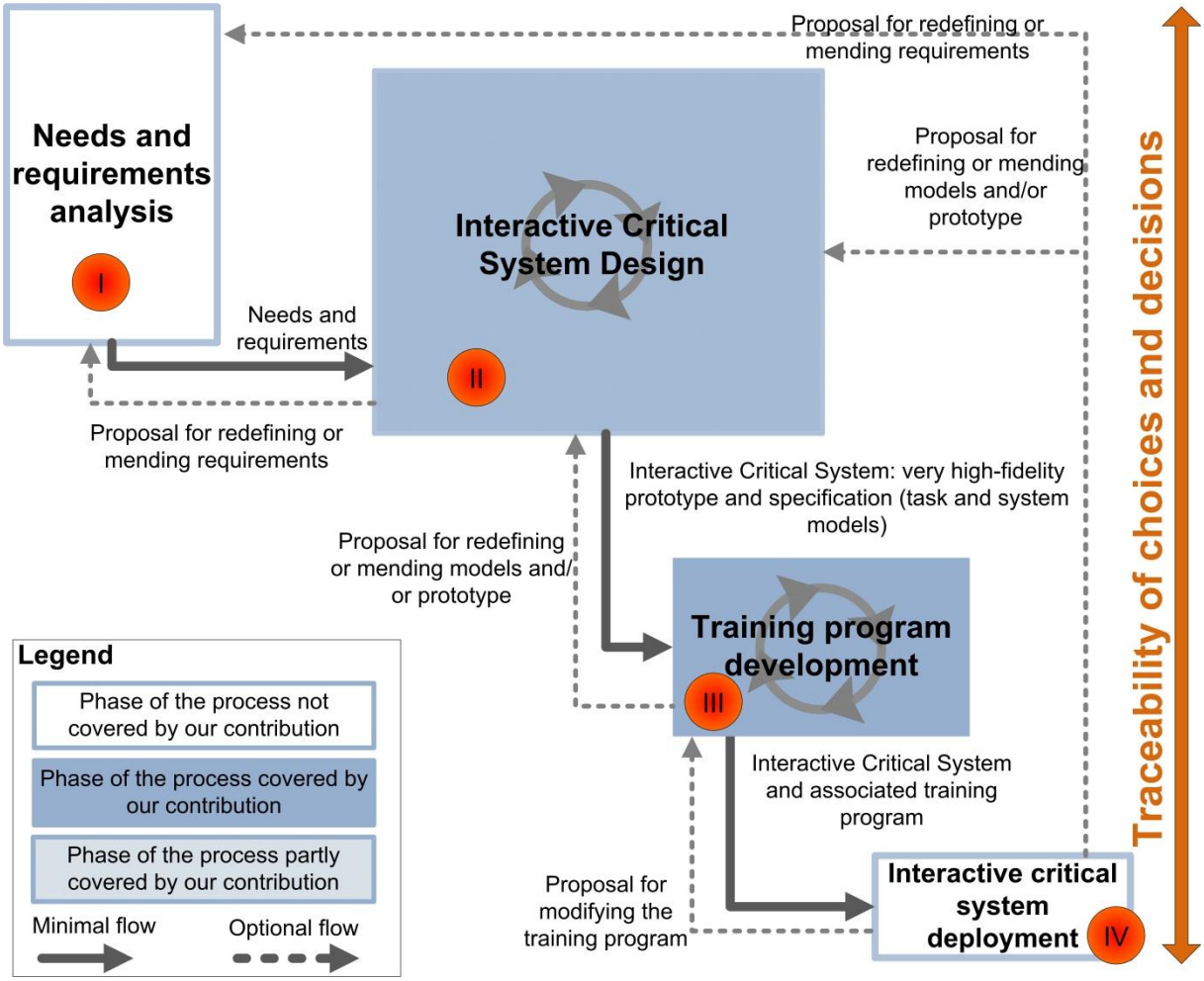


Figure 57. Diagramme général du processus de développement d'un système interactif critique proposé

La Figure 57 décrit le processus de développement d'un système interactif critique qui débute par la phase d'analyse des besoins et exigences (I) (intitulée « Needs and requirements analysis »). Cette première phase consiste à analyser les besoins des utilisateurs et les exigences liées à l'opérabilité et à la fiabilité du système interactif critique conçu. Elle produit un ensemble de besoins et d'exigences qui seront le point de départ de la phase de conception du système interactif critique. La phase de conception du système interactif critique (II) (intitulée « Interactive Critical System Design » et détaillée dans la section 2), consiste à étudier la liste des besoins et exigences et à produire un prototype très haute-fidélité du système et sa spécification selon les propriétés d'utilisabilité et de fiabilité requises. Ces éléments sont alors transférés vers la phase de développement du programme de formation (III) (intitulée « Training program development » et détaillée dans la section 3). Ils servent à analyser les besoins en formations pour les personnes allant opérer le système et à concevoir, développer, implémenter et évaluer un programme de formation adapté.

Lorsque le programme de formation est validé, le processus de développement du système interactif critique est poursuivi vers la phase de déploiement (IV) en aval (intitulée « Interactive critical system deployment »).

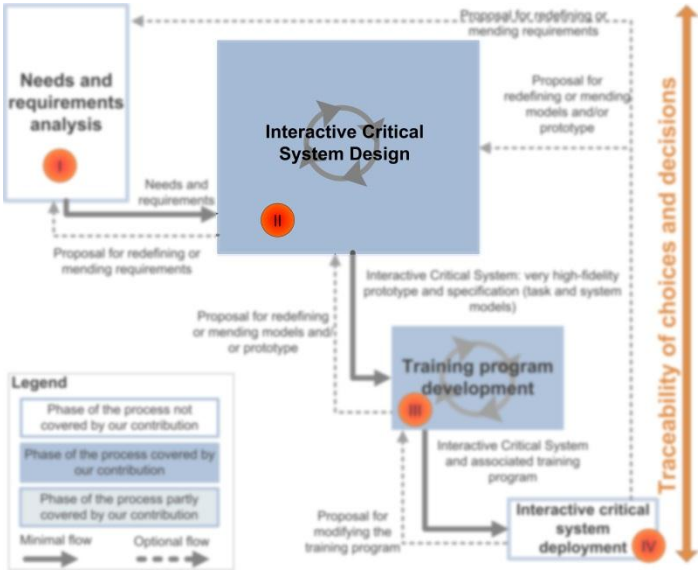
Chaque phase du processus peut fournir un retour aux phases précédentes si un problème d'utilisabilité ou de fiabilité non détecté auparavant a émergé. Ces problèmes peuvent remettre plus ou moins en

cause les éléments produits dans les phases amont et engendrer des modifications à ces éléments ou des changements plus importants (flèches de retour en pointillé « Proposal for redefining or mending… » sur la Figure 57).

De plus, ce processus fournit un support pour la traçabilité des choix de développement effectués tout au long du processus (détaillé dans la section 4).

Tout comme les processus de développement de systèmes interactifs et les processus de développement des systèmes critiques, ce processus de développement nécessite la participation de différents corps de métier, comme des concepteurs graphiques, des ergonomes, des experts en utilisabilité, des ingénieurs de conception et développement logiciel, des formateurs et consultants en formation.

2 Phase de conception des systèmes interactifs critiques



La Figure 58 décrit en détail la phase de conception du système interactif critique (cette phase correspond à la partie III de la Figure 57, miniaturisée ci-contre). Elle contient trois sous-phases itératives principales :

- Analyse et modélisation des tâches utilisateur (pastille 3 sur la Figure 58), détaillée dans la section 2.1
- Phase itérative de prototypage basse-fidélité du système (pastilles 2 3 4 5 sur la Figure 58), détaillée dans la section 2.2
- Phase itérative de modélisation formelle et prototypage très haute-fidélité du système (pastilles 6 7 8 9 10 sur la Figure 58), détaillée dans la section 2.3

Le nombre d'itérations de ces sous-phases est conditionné par l'étape d'évaluation quantitative des performances de l'utilisateur (pastille 10 sur la Figure 58). Cette étape, détaillée dans la section 2.4, est déterminante car si les résultats mesurés correspondent aux objectifs d'utilisabilité et de fiabilité, les modèles et prototype pourront être utilisés dans les phases suivantes.

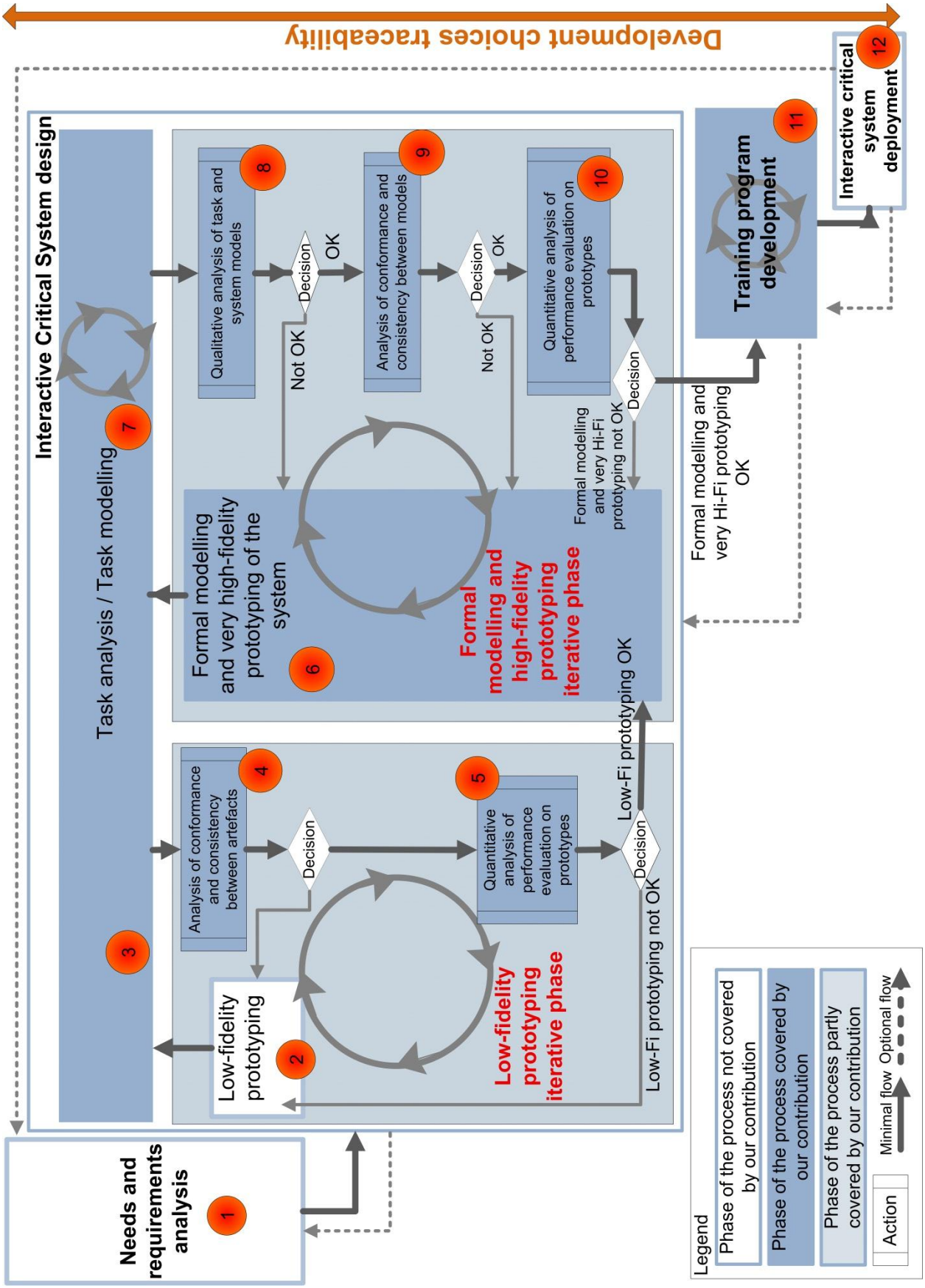
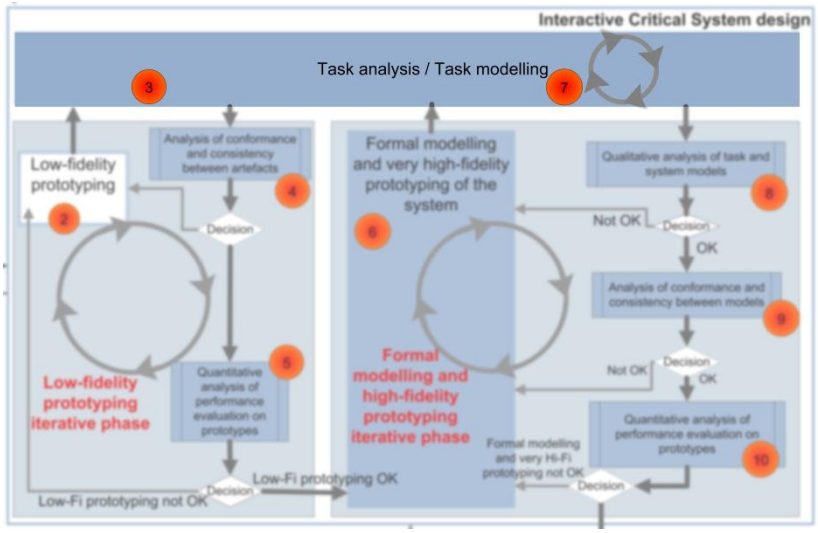


Figure 58. Diagramme détaillé du processus de développement d'un système interactif critique proposé

2.1 Analyse et modélisation des tâches

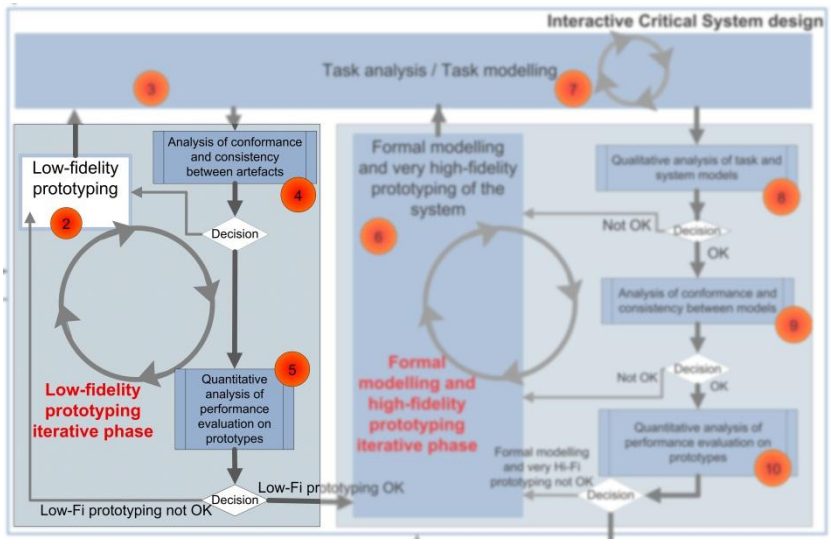


La phase d'analyse et modélisation des tâches 3 (Figure 58, miniaturisée ci-contre) exploite les besoins des utilisateurs et les exigences liées aux propriétés d'utilisabilité et de fiabilité.

L'importance de l'analyse et de la modélisation des tâches dans la conception des systèmes interactifs a été soulignée dans le chapitre 2. Dans le cadre de notre processus de développement, elle est une phase primordiale du processus :

- Chaque itération d'une sous-phase contient une étape d'analyse et modélisation des tâches, qui permet de vérifier la conformité et la cohérence entre les tâches utilisateurs et le système conçu.
- Les évaluations qualitatives et quantitatives des performances des utilisateurs sur les prototypes basse-fidélité 5 et haute-fidélité 10 sont basées sur cette analyse et modélisation, afin de s'assurer que les buts de l'utilisateur sont atteignables et que les actions effectuées par l'utilisateur ne mettront pas en péril l'intégrité de l'utilisateur, l'intégrité de ses pairs et celle du système.
- La phase de développement du programme de formation 11 est basée sur l'analyse des tâches à accomplir pour opérer le système.

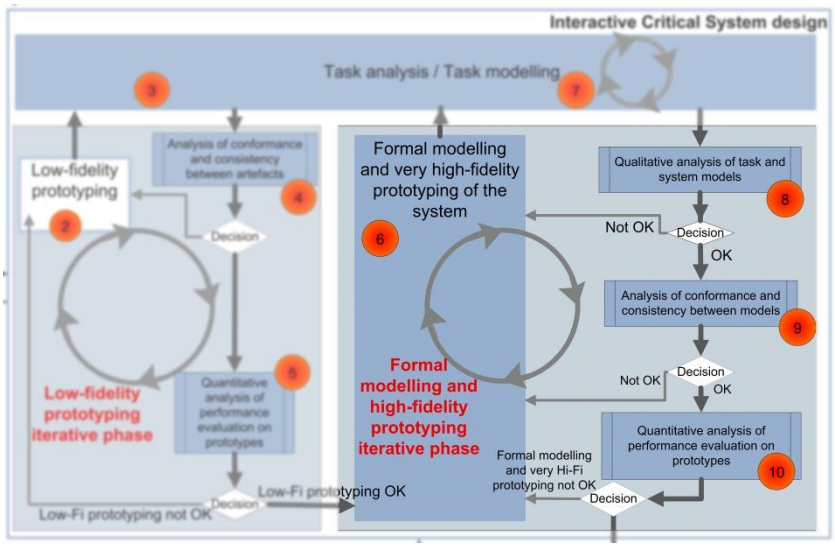
2.2 Phase itérative de prototypage basse-fidélité



La phase itérative de prototypage basse-fidélité 2 3 4 5 (Figure 58, miniaturisée ci-contre), en s'appuyant sur les besoins utilisateurs et exigences liées aux propriétés d'utilisabilité et de fiabilité, permet de concevoir plusieurs prototypes basse-fidélité 2 du système. Durant cette phase, plusieurs ébauches d'interfaces utilisateurs sont produites avant de concevoir et développer toutes les fonctionnalités sous-jacentes du système interactif critique. Chacun de ces

prototypes est ensuite confronté aux modèles de tâches 3 afin de déterminer qualitativement 4 s'ils sont conformes et cohérents avec les tâches de l'utilisateur. Tant que les prototypes basse-fidélité ne sont pas conformes aux tâches des utilisateurs, cette étape de prototypage est reconduite. Lorsqu'ils sont conformes et cohérents, une analyse quantitative des performances des utilisateurs sur ces prototypes est effectuée 5. Si les résultats correspondent aux besoins des utilisateurs et aux exigences de propriétés d'utilisabilité et de fiabilité, les éléments retenus des prototypes basse-fidélité sont ensuite transmis vers l'étape suivante de modélisation formelle et de prototypage très haute-fidélité.

2.3 Phase itérative de modélisation formelle et prototypage très haute-fidélité du système interactif critique



La phase itérative de modélisation formelle et de prototypage haute-fidélité 6 7 8 9 10 (Figure 58, miniaturisée ci-contre) a été conçue en fonction de l'association des deux éléments formant le système interactif critique :

- les parties du système interagissant avec les sens des utilisateurs. Ils peuvent avoir différents aspects : affichage graphique, clavier, souris, synthèse vocale, reconnaissance vocale, objets interactifs (WIMP ou post-WIMP, liste non exhaustive)
- la modélisation formelle du comportement du système interactif critique. Elle permet la vérification des propriétés de fiabilité requises pour le système.

Ce prototypage est qualifié de *très* haute-fidélité dans le sens où il s'exécute sur une plateforme logicielle et matérielle et contient toutes les fonctionnalités requises du système interactif critique à déployer. De plus, les éléments retenus à l'issue de cette phase formeront le système interactif critique déployé ou une partie de ces éléments seront contenus dans la version déployée.

Sur la Figure 58, le démarrage de cette phase est déclenché lorsque les éléments de prototypage basse-fidélité ont été validés lors de la phase précédente. Elle débute par le prototypage très haute-fidélité et la construction des modèles du système 6. Lorsqu'une première version est établie, le prototype et les modèles sous-jacents du comportement du système interactif critique sont confrontés aux modèles de tâches 7. Une analyse qualitative des modèles de tâches et du système est effectuée afin de s'assurer qu'ils vérifient les propriétés d'utilisabilité et de fiabilité requises pour le système 8. Si toutes les propriétés exigées ne sont pas vérifiées dans une partie du système, une nouvelle itération de la phase de modélisation formelle et prototypage est amorcée. Si les modèles de tâches et du système vérifient les propriétés, ils sont ensuite mis en correspondance 9 afin de s'assurer que les fonctions du système sont conformes et cohérentes avec les tâches de l'utilisateur. Si une non-conformité ou une

incohérence est détectée, une nouvelle itération de la phase de modélisation formelle et prototypage est amorcée. Si une version conforme aux besoins utilisateurs et cohérente avec les tâches utilisateurs du prototype haute-fidélité et des modèles de spécification formelle est disponible, une analyse quantitative des performances utilisateurs est effectuée sur cette version[10]. Cette étape complète les étapes précédentes d'analyse qualitative en validant les propriétés d'utilisabilité et de fiabilité du système. Enfin, lorsqu'une version vérifiée et validée du système est disponible, elle est alors transférée vers le processus de développement du programme de formation, ainsi que vers les phases aval de déploiement du système. Sinon, une nouvelle itération de spécification formelle et de prototypage haute-fidélité du système est amorcée.

2.4 Analyse quantitative des performances de l'utilisateur

Cette étape permet de déterminer quels modèles des tâches utilisateur, modèles du système et prototypes pourront être utilisés en aval du processus de développement, suite à une évaluation quantitative des performances de l'utilisateur avec ces artefacts.

Notre processus de développement permet d'utiliser deux types d'évaluation de l'utilisabilité du système. La prochaine section (2.4.1) expose les étapes permettant d'analyser quantitativement les performances de l'utilisateur avec une approche non formelle et avec un prototype basse fidélité. La section 2.4.2 expose les étapes permettant d'analyser quantitativement les performances de l'utilisateur avec une approche formelle et un prototype très haute-fidélité.

Dans les deux cas, nous distinguons les rôles de deux parties impliquées dans le développement d'un système interactif : le concepteur du système et l'expert en utilisabilité. Les activités de l'expert en utilisabilité sont nombreuses et variées et consistent à :

- Définir des hypothèses.
- Définir des scénarios d'utilisation de l'application.
- Identifier les métriques utilisées pour mesurer et évaluer les performances par rapport aux hypothèses.
- Utiliser des outils pour collecter les données sur l'utilisation de l'application.
- Calculer les taux succès et échecs lors de l'exécution des tâches utilisateurs.
- Calculer les performances de l'utilisateur.
- Identifier et choisir les utilisateurs.
- Effectuer plusieurs sessions de tests utilisateurs.
- Analyser les résultats.
- Modifier les tests d'utilisabilité si besoin.
- Fournir des recommandations pour améliorer l'application conçue.

2.4.1 Evaluation de l'utilisabilité non formelle et un prototype basse fidélité

La Figure 59 décrit de manière générale la séquence des étapes couramment empruntées pour évaluer l'utilisabilité d'un prototype très haute-fidélité d'un système interactif grâce aux tests utilisateur.

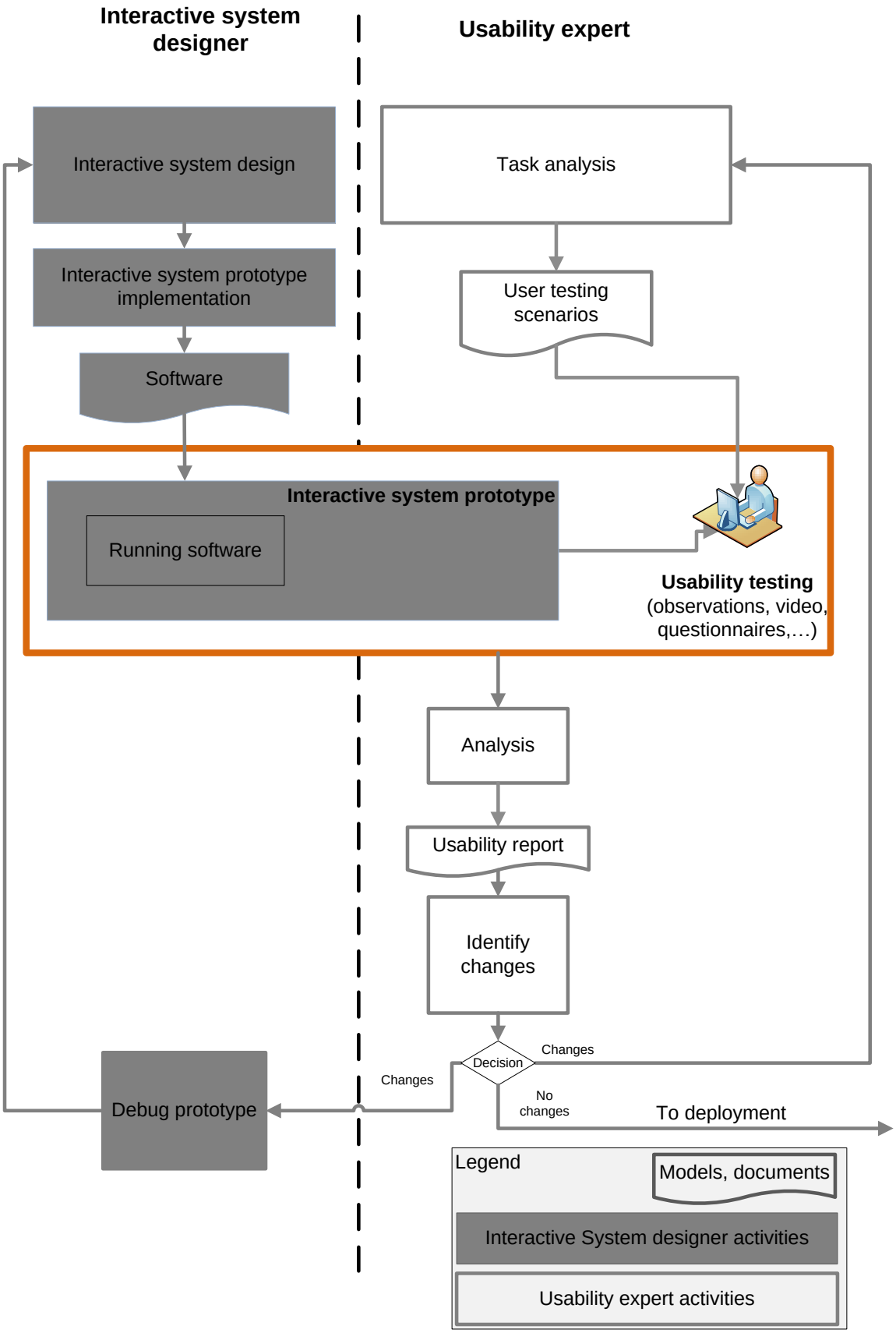


Figure 59. Diagramme de flux de l'évaluation de l'utilisabilité d'une technique d'interaction

La partie gauche de la Figure 59 décrit les différentes étapes effectuées par le concepteur du système interactif avant la phase de tests utilisateurs, de la conception du système au développement du prototype fonctionnel s'exécutant sur une configuration matérielle. De la même manière, l'expert en utilisabilité, sur la partie droite de la Figure 59 analyse les tâches de l'utilisateur et prépare les scénarios de test avant la phase de tests utilisateur. Suite aux tests, dans le cas où des modifications sont nécessaires, l'expert en utilisabilité analyse les éléments recueillis et identifie les modifications à apporter au système et aux prochains scénarios de test. Une fois ces modifications identifiées, elles seront aussi transmises au concepteur du système qui essaiera de trouver dans son prototype où se situent les problèmes correspondants et comment les résoudre.

2.4.2 Evaluation de l'utilisabilité avec une approche formelle et un prototype très haute-fidélité

Les travaux de (Bernhaupt, Navarre, Palanque, & Winckler, 2007) et de (Ladry J.-F. , 2010) proposent un processus d'évaluation de l'utilisabilité de l'application et des techniques d'interaction basé sur la modélisation formelle. Notre contribution approfondit l'étude du processus d'évaluation d'une technique d'interaction et propose une nouvelle version pouvant être utilisée pour évaluer un système interactif.

La complexité et le nombre des activités menées par l'expert en utilisabilité (comme évoqué au début de la section 2.4) engendrent plusieurs difficultés au cours de certaines étapes de l'évaluation de l'utilisabilité :

- Les étapes d'identification et de résolution d'un problème d'interaction dans le prototype peuvent être très compliquées et fastidieuses. En effet, le concepteur doit investiguer à quelle partie du code source de l'application correspond le problème d'utilisabilité rencontré.
- Dans le cas où l'application permet d'enregistrer les évènements applicatifs s'étant déroulés pendant la session de test, le concepteur doit trouver le lien entre ces évènements et les évènements déclenchés par l'utilisateur.
- L'expert en utilisabilité et le concepteur de l'application doivent s'assurer que l'application sera capable de rejouer exactement le même scénario de manière déterministe.

L'approche de modélisation formelle et de prototypage très haute-fidélité nous permet de proposer une nouvelle version du processus d'évaluation. La Figure 60 décrit les nouvelles étapes proposées pour compléter celles empruntées lors de l'évaluation de l'utilisabilité du prototype basse fidélité du système interactif :

- Le prototype utilisé pour l'évaluation est le prototype très haute-fidélité et il dispose d'une fonctionnalité de traçage et d'enregistrement des évènements applicatifs.
- Les étapes d'analyse et d'identification des problèmes sont menées conjointement par le concepteur et par l'expert en utilisabilité.

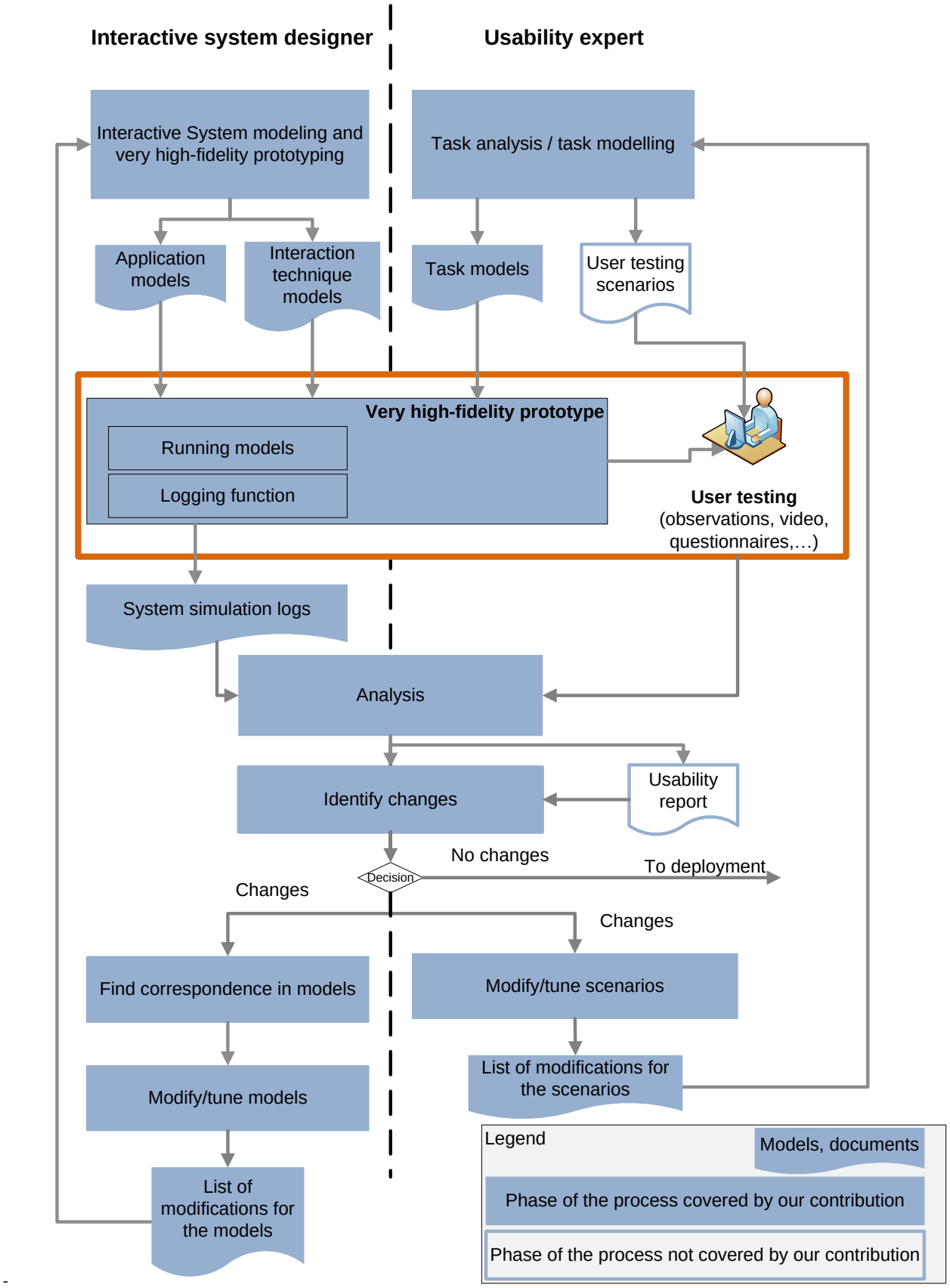


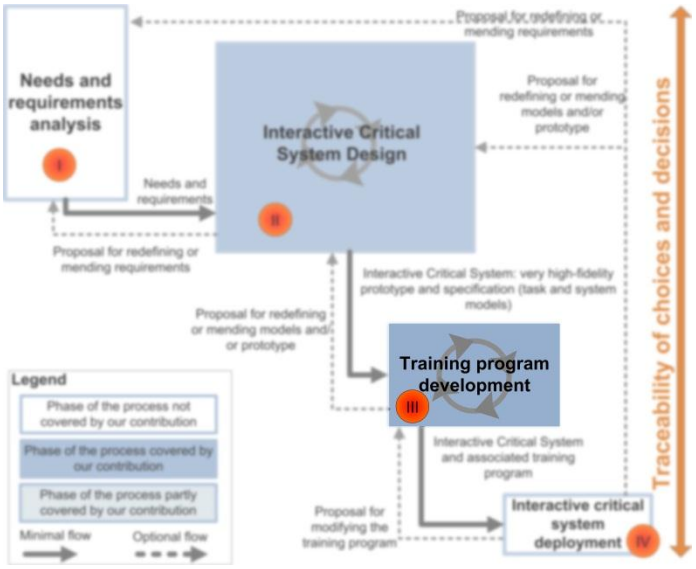
Figure 60. Diagramme de flux de l'évaluation de l'utilisabilité d'un système interactif complémenté par notre approche

L'approche de modélisation formelle et prototypage très haute-fidélité amène des solutions aux problèmes évoqués précédemment :

- Les étapes d'identification et de résolution d'un problème d'interaction peuvent être facilitées par l'association logique des modèles de comportement du système à l'interface du système. La mise en œuvre de cette association dans un environnement d'exécution du prototype permet d'acquérir des traces détaillées de l'historique des évènements d'interaction durant la session de tests utilisateurs.
- Le concepteur peut trouver plus rapidement le lien entre les évènements applicatifs internes et les évènements déclenchés par l'utilisateur grâce aux modèles.
- L'utilisation des modèles permet à l'expert en utilisabilité et au concepteur de l'application de vérifier que le système interactif sera capable de rejouer exactement le même scénario de manière déterministe.

Un environnement de mise en œuvre de la phase de conception d'un système interactif est présenté dans la section Chapitre 62.2 du chapitre suivant.

3 Phase de développement du programme de formation associé



Cette section présente la phase de développement du programme de formation (intitulée « Training program development », pastille III sur la Figure 57 miniaturisée ci-contre). Durant le processus de développement du système interactif critique, la phase de conception II décrite dans la section précédente produit une version du système par rapport aux propriétés d'utilisabilité et de fiabilité requises. Cette version est alors transférée vers la phase de développement du programme de formation III.

Nous avons précédemment expliqué que les phases de développement d'une approche systématique à la formation (*Analysis Design Development Implementation Evaluation*) étaient largement utilisées pour la conception de programmes de formation à l'utilisation des systèmes critiques (tel que décrit dans la section 2.2 du chapitre 2) et qu'il était basé sur l'analyse des tâches prescrites pour l'utilisation du système. L'analyse des tâches est la clé de voute du programme de formation car elle permet de :

- Définir les objectifs du programme de formation.
- Identifier et sélectionner les tâches que les personnes formées devront maîtriser à l'issue de la formation.
- Mesurer les performances avant et après le programme de formation.
- Définir les connaissances et compétences requises avant de démarrer la formation.

Ainsi, les modèles de tâches produits lors de la phase de conception du système interactif critique (Figure 57) sont des données nécessaires pour le développement du programme de formation. De plus, la spécification du système ainsi que le prototype très haute-fidélité permet de concevoir un programme de formation au plus proche du système sur lequel les utilisateurs devront opérer.

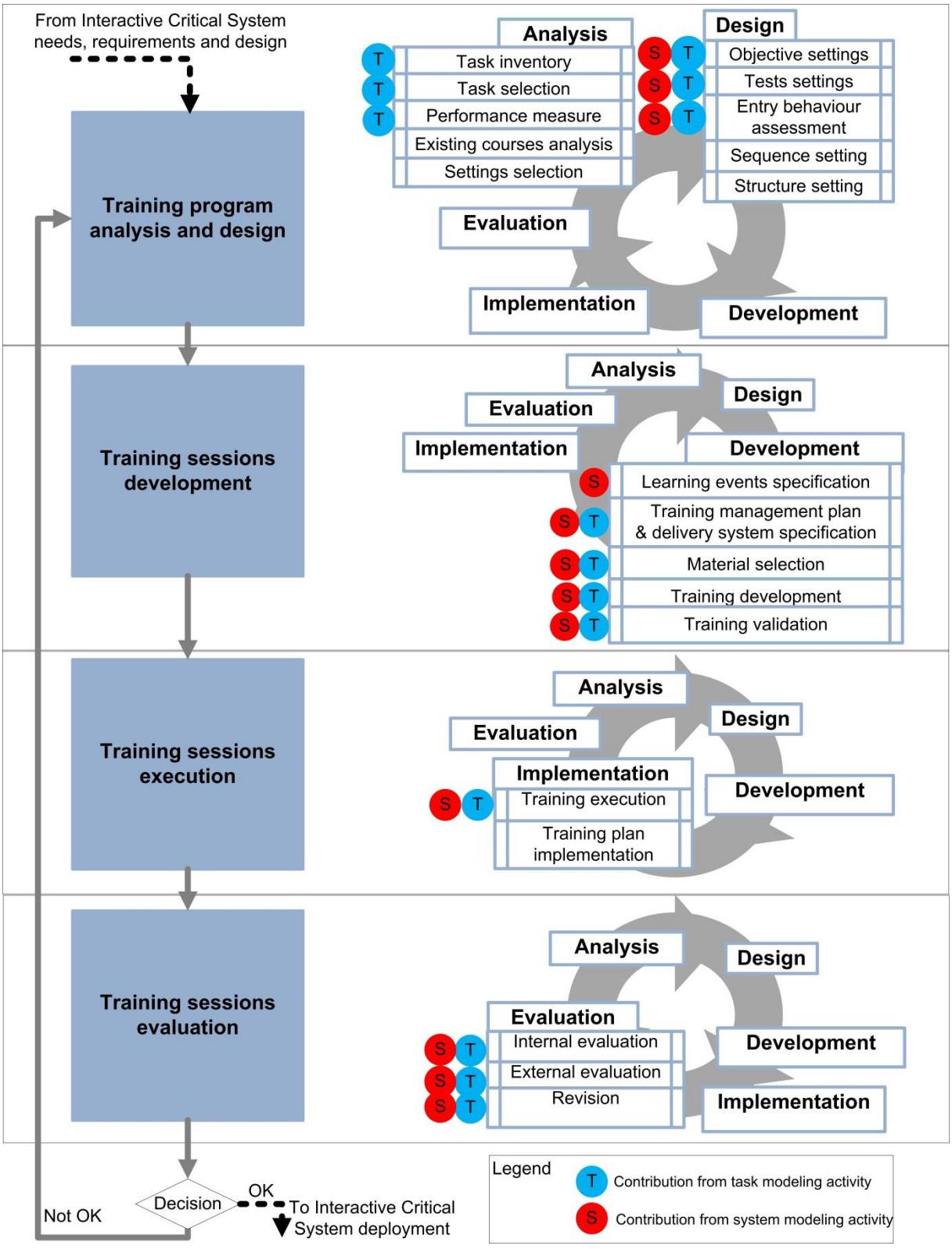


Figure 61. Processus de développement du programme de formation associé

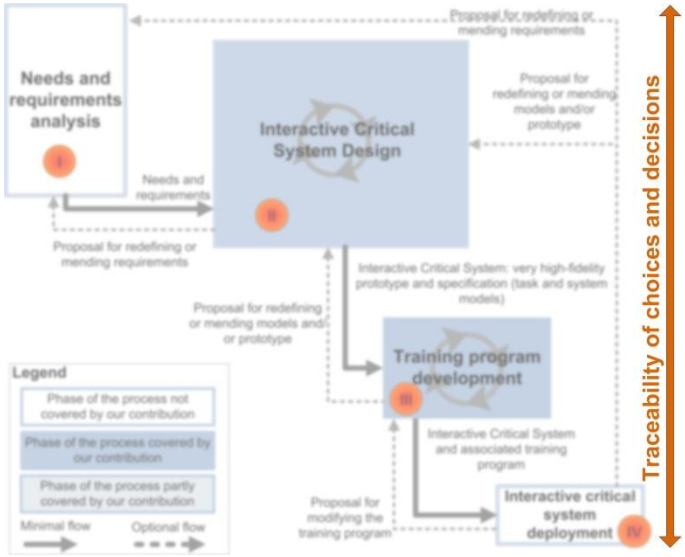
La Figure 61 détaille quatre étapes, parmi celles requises pour développer un programme de formation, pour lesquelles nous apportons une contribution (blocs rectangulaires sur partie gauche). La partie droite de la Figure 61 montre les sous-phases correspondantes dans le processus ADDIE. Chaque sous-phase est complémentée par l'activité de modélisation des tâches T et/ou par l'activité de modélisation du système S :

- Durant la phase d'analyse et de conception du programme de formation (première étape), nous proposons d'utiliser les modèles produits par la phase de conception du système interactif critique. Lors de la phase d'analyse, les modèles de tâche permettent d'inventorier et de sélectionner les tâches à apprendre et définir les critères de performance requis. Lors de la phase de de conception, les modèles produits sont utilisés pour définir les objectifs du programme de formation, en termes de tâches à savoir exécuter. Ils sont aussi utilisés pour concevoir les scénarios d'entraînement, définir les paramètres d'évaluation du personnel à former et déterminer les prérequis nécessaires pour suivre la formation. Les modèles permettent aussi de définir quels niveaux de performances peuvent et devraient être atteints.
- Durant la phase de développement d'une session de formation (seconde étape), nous proposons d'utiliser les modèles de tâches et du système ainsi que le prototype haute-fidélité pour déterminer comment s'effectuera la formation sur le plan pratique. Ils permettent d'aider à sélectionner les cas d'étude utilisés pour faciliter l'apprentissage et à développer les scénarios d'utilisations auxquels sera confronté le personnel en formation. Cette phase vise aussi à produire le matériel nécessaire au déroulement des sessions de formation.
- Pendant la phase d'exécution des sessions de formation (troisième étape), nous proposons d'utiliser tous les éléments produits par les étapes précédentes pour fournir au personnel en formation un entraînement sur prototype très haute-fidélité avec les scénarios conçus grâce aux modèles de tâches conformes au système et selon les objectifs de la formation.
- La phase d'évaluation des sessions de formation (quatrième étape) permet d'évaluer les résultats du personnel ainsi que l'efficacité de cette partie de la formation. S'il apparaît que tout ou partie de ces sessions de formation ne permettent pas au personnel d'atteindre les objectifs nécessaires pour opérer le système selon les propriétés de fiabilité requises, alors une nouvelle itération de ce processus sera initiée.

Nos propositions basées sur les modèles de tâches et du système pour compléter les approches systématiques à la formation afin d'améliorer la fiabilité des opérateurs sont aussi décrites dans nos travaux initiaux sur l'approche de développement proposée dans cette thèse (Martinie, Palanque, Navarre, & Winckler, 2010).

Un environnement de mise en œuvre pour la préparation des sessions de formations, leur déroulement ainsi que leur évaluation est détaillé dans la section 4 du chapitre 6.

4 Traçabilité des besoins et exigences tout au long du processus de développement



La traçabilité des choix de conception ainsi que l'exploration de leurs possibles est un aspect important tout au long du processus de développement d'un système critique. Certains organismes exigent que la traçabilité des choix de conception ainsi que la traçabilité des exigences liées à ces choix de conception soient fournis pour passer en phase de certification du système critique développé (comme détaillé dans la sous-section 1.3.3 du chapitre 2).

Le processus de développement décrit dans ce chapitre intègre cet aspect de traçabilité. Nous proposons de lier logiquement les modèles issus de la phase de conception du système interactif critiques aux modèles de données contenant les choix de conception et la couverture des exigences par rapport à ces choix de conception. Un environnement de mise en œuvre de cette proposition est détaillé dans la section 2.4 du chapitre 6.

5 Avantages et limites de l'approche

Cette section permet de mettre en valeur les avantages et limites de l'approche proposée et en particulier du processus de développement sous-jacent.

5.1 Avantages liés à l'application du processus proposé

Le processus de développement proposé intègre les phases clé des processus de développement des systèmes interactifs et des systèmes critiques afin de combler les manques de chacun de ses types de processus pour le développement d'un système interactif critique :

- Ce processus permet d'analyser de manière détaillée la compatibilité, la cohérence et la conformité des tâches de l'utilisateur avec les interactions et fonctionnalités du système. La modélisation détaillée des tâches et du système associée aux étapes d'évaluations qualitative et quantitative permettent de s'approcher, pour chaque but ou sous-but de l'utilisateur, de la combinaison optimale des tâches qui ont intérêt à être effectuées par l'utilisateur et des fonctions qui ont intérêt à être exécutées par le système.
- La phase de développement du programme de formation en tant qu'étape intégrée du programme de développement du système interactif critique est un concept important de notre contribution. Ce processus apporte un cadre de développement et des techniques pour garantir les propriétés d'utilisabilité et de fiabilité du système mais aussi pour renforcer la fiabilité du personnel opérant le système. De plus, cette approche permet de mettre à profit l'activité de modélisation des tâches (effectuée lors de la conception du système) pour la phase de développement du programme de formation mais aussi de permettre au programme de formation d'être au plus proche du système à opérer.

- La traçabilité des besoins et exigences sur les différentes options de conception tout au long du processus de développement en fonction des propriétés d'utilisabilité et de fiabilité requises permettent de surveiller et contrôler de manière continue la conformité du système conçu.

5.2 Limites liées à l'application du processus proposé

Ce processus, explicitement lié aux systèmes interactifs critiques, est plus lourd à mettre en œuvre et à gérer qu'un processus de développement de système interactif classique (comme les processus décrits dans la section 1 du chapitre 2). En particulier, la phase de modélisation formelle implique deux conséquences immédiates :

- Elle requiert une équipe de conception pluridisciplinaire dont certains membres doivent être aguerris aux méthodes formelles.
- Plus le système interactif critique sera complexe, plus le nombre de modèles sera élevé et plus leurs imbrications seront compliquées.

Ces deux conséquences amènent en chaîne à une troisième qui est celle du temps et des coûts de développement, devant théoriquement être augmentés. Cependant, à ce jour, la modélisation formelle du système est le seul moyen de garantir les propriétés à vérifier pour un système critique et elle devient nécessaire pour gérer complexité croissante des systèmes interactifs critiques. De plus, pour palier à ce problème, les solutions suivantes peuvent être envisagées :

- La capitalisation d'une partie des modèles peut être envisagée d'un système à un autre.
- L'étude des parties du processus où la modélisation formelle ne serait pas indispensable pour garantir les propriétés requises.

6 Conclusion

Ce chapitre décrit de manière théorique le processus de développement pour les systèmes interactifs critiques et/ou complexes proposé dans le cadre de l'approche présentée dans cette thèse. La Figure 57 décrit les principales étapes de ce processus et la Figure 58 détaille les sous-phases de la conception du système interactif critique. On y remarque que la phase d'analyse et de modélisation des tâches est incluse dans la conception du système interactif critique. Elle peut aussi être démarrée en amont au moment de l'analyse des besoins et exigences. De la même manière, des étapes de chaque phase du processus de développement proposé peuvent être démarrées dès que des éléments nécessaires sont disponibles. Par exemple, certaines étapes du développement du programme de formation comme l'analyse des tâches peuvent aussi démarrer avant qu'une version satisfaisante des modèles formels et du prototype très haute-fidélité ne soit fourni. Ce type de mise en œuvre du processus est schématisé par la Figure 62.

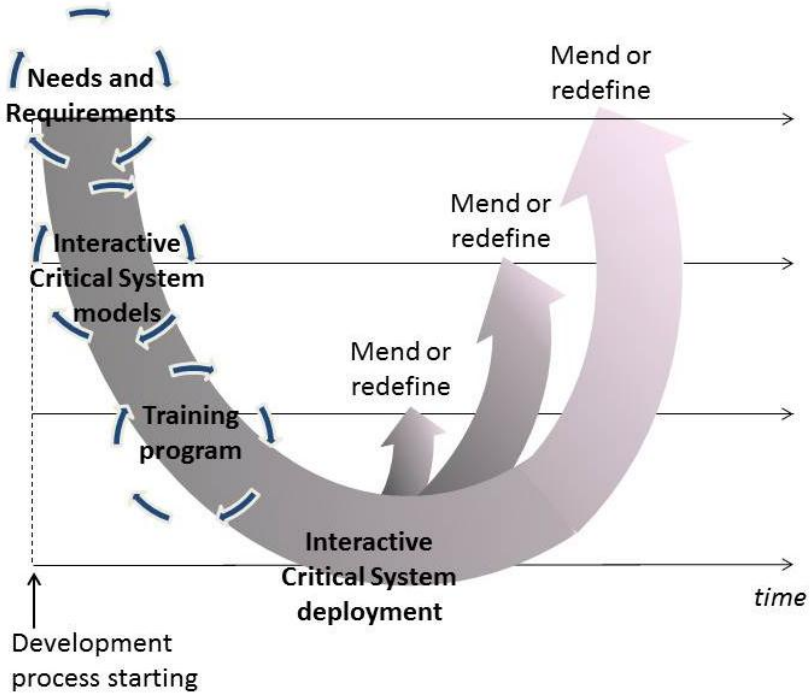


Figure 62. Cycle de vie des éléments produits lors de la mise en œuvre des différentes phases du processus de développement du système interactif critique

La Figure 62 montre que chaque phase peut théoriquement démarrer au même moment (le temps consacré à chaque phase est représenté par une flèche horizontale pour chaque phase). Tout au long du processus, des propositions pour corriger ou redéfinir les différents artefacts du processus (modèles, programme de formation,…) peuvent remonter vers la phase correspondante. Ces propositions seront cependant moins nombreuses par rapport au nombre d'artefacts rendus disponibles d'une phase à l'autre (flèches remontantes en gris de plus en plus clair). En effet, chaque phase amont veillant à produire des éléments remplissant les propriétés d'utilisabilité, de fiabilité et d'opérabilité, peu de problèmes doivent théoriquement surgir dans les phases aval. L'approche présentée a fait l'objet de plusieurs travaux, dont (Martinie, Palanque, Navarre, & Winckler, 2010) et (Martinie & Palanque, 2011).

Pour conclure ce chapitre, nous souhaitons attirer l'attention sur le cas de systèmes interactifs critiques pour la sécurité de ses utilisateurs. Les travaux de (Basnyat, 2006) ont montré qu'il était nécessaire d'ajouter des étapes supplémentaires dans les phases itératives de modélisation. Notre processus de développement s'intègre à cette approche et nous recommandons (Navarre, Palanque, Martinie, & Steere, 2011) de :

- Complémenter la modélisation des tâches utilisateurs avec des étapes d'analyse d'erreurs humaines.
- Complémenter la modélisation du système avec des étapes analyses de risques et de modélisation de barrières de sécurité.
- Ajouter une phase de modélisation de la sécurité avec notamment l'analyse des rapports d'accidents et incidents.

Dans le chapitre suivant, nous complémentons ce processus avec des techniques et outils de mise en œuvre pour fournir un support complet à notre approche de développement.

Chapitre 6- Mise en œuvre de l'approche de développement pour les systèmes interactifs critiques

Dans ce chapitre, nous présentons les techniques, notations et outils fournissant un support à la mise en œuvre de l'approche de développement d'u système interactif critique et de son programme de formation associé proposée dans cette thèse et exposée au chapitre 5. Chaque section de ce chapitre, à partir de la section 2, détaille la mise en œuvre d'une phase de l'approche proposée dans cette thèse. Les deux phases de l'approches nommées *Analyse des besoins et exigences* et *Phase itérative de prototypage basse-fidélité,* font partie de notre contribution en tant qu'étape indispensable pour notre approche. Cependant, notre contribution ne s'attache pas à fournir de moyens particuliers pour leur réalisation (excepté HAMSTERS en support à l'analyse de tâches utilisateur). Elles ne seront donc pas détaillées autant que les autres phases. Nos contributions pour la mise en œuvre de l'approche proposée dans cette thèse s'attache à fournir les moyens pour :

- La modélisation des tâches utilisateur d'un système interactif complexe en s'appuyant sur la notation outillée HAMSTERS (sa mise en œuvre exemplifiée est présentée dans la sous-section 2.2.1).
- La conception, le développement et la validation d'un prototype très haute-fidélité du système interactif en s'appuyant sur l'utilisation synergique des modèles du système et des modèles de tâches (sa mise en œuvre exemplifiée est présentée dans la sous-section 2.2.3).
- L'évaluation des performances utilisateur sur le système conçu en s'appuyant sur l'utilisation d'une technique outillée d'analyse quantitative (sa mise en œuvre exemplifiée est présentée dans la sous-section 2.2.4).
- Le développement d'un programme de formation associé au système en s'appuyant sur l'utilisation synergique des modèles du système, des modèles de tâches et du contenu des sessions de formation effectuées avec le prototype très haute-fidélité du système (sa mise en œuvre exemplifiée est présentée dans la sous-section 2.3).
- La traçabilité des besoins et exigences tout au long des différentes phases du processus de développement en s'appuyant sur la notation outillée DREAMER (sa mise en œuvre exemplifiée est présentée dans la sous-section 2.4).

La première section de ce chapitre est dédiée à la présentation de l'environnement logiciel utilisé pour mettre en œuvre l'approche proposée dans cette thèse. Cette section nous permet de décrire les fonctionnalités requises pour cette mise en œuvre et notre contribution à cet environnement et aux outils logiciels qui le composent pour intégrer ces fonctionnalités (présentée dans la sous-section 1.1). Pour la phase de traçabilité, cette section nous permet aussi de mettre en valeur notre contribution à la notation de conception rationalisée TEAM (présentée dans la sous-section 1.2).

1 Présentation de l'environnement de mise en œuvre

Cette première section présente les différentes parties de l'environnement de mise en œuvre du processus de développement exposé dans le chapitre 5. Tout au long de cette section, nous allons introduire ces parties et les raisons pour lesquelles elles ont été créées. Dans le Chapitre 1, nous avons expliqué de manière abstraite comment, durant la phase de conception d'un système interactif critique, les activités de modélisation des tâches utilisateur et les activités de modélisation formelle du système s'imbriquaient afin de permettre la vérification et la validation de la conformité entre les activités et actions de l'utilisateur et les fonctions et interactions proposées par le système. Ces phases de

conception étant fortement itératives, des options de conceptions vont apparaître tout au long du processus. En conséquence, l'environnement de mise en œuvre fournit deux fonctions principales qui correspondent à deux parties :

- Environnement logiciel intégré de modélisation des tâches, de modélisation formelle du système et de prototypage très haute-fidélité pour leur exploitation synergique. Il est détaillé dans la sous-section 1.1.
- Environnement logiciel de conception rationnalisé afin de structurer et gérer la complexité des choix de conception et tracer l'analyse des besoins et exigences par rapport à ces choix tout au long du processus de conception. Cet environnement est basé sur la notation TEAM. La notation et l'environnement sont détaillés dans la section 1.2.

1.1 Environnement logiciel intégré d'exploitation synergique des modèles

Cette section décrit les exigences requises pour fournir un support à la mise en œuvre de l'approche proposée par cette thèse ainsi que les différents éléments logiciels permettant de la mettre en œuvre. Cet environnement d'exploitation synergique des modèles a été élaboré conjointement avec Eric Barboni, David Navarre et Philippe Palanque dans la cadre du projet de recherche Tortuga[5] (CNES R-S08/BS-0003-029).

1.1.1 Exigences sur l'environnement d'exploitation synergique

Dans la section 2.2 du chapitre 2, nous avons indiqué les exigences pour la sélection des notations et outils de modélisation. A l'issu du chapitre 2, les notations et outils retenus pour cette exploitation synergique sont :

- L'outil logiciel de spécification et prototypage PetShop pour la modélisation du comportement du système (dont la description détaillée a été effectuée dans le Chapitre 2)
- L'outil logiciel d'édition et de simulation HAMSTERS pour la modélisation des tâches utilisateur (dont la description détaillée a été effectuée dans le Chapitre 4).

Afin de permettre la mise en œuvre du processus de développement, l'environnement doit fournir, suivant deux modes, les fonctionnalités suivantes au concepteur :

- Mode édition :
  - Edition des modèles de tâches.
  - Edition des modèles du prototype très haute-fidélité.
  - Edition des scénarios d'utilisation.
  - Mise en correspondance des tâches utilisateur avec les évènements d'entrée/sortie du prototype très haute-fidélité.
  - Edition des plans de sessions de formation.
  - Edition des profils des personnels formés.

- Mode exécution :
  - Simulation des modèles de tâches.
  - Exécution du prototype très haute-fidélité et de ses modèles sous-jacents.
  - Pour chaque type de modèle, visualisation à tout moment des parties des modèles couramment exécutées.

[5] http://www.irit.fr/recherches/ICS/projects/tortuga/index.htm

- Co-exécution simultanée des modèles de tâches, du prototype très haute-fidélité et de ses modèles sous-jacents.
- Exécution des scénarios d'utilisation d'une session de formation.

Le paragraphe suivant indique l'architecture fonctionnelle générale de l'environnement conçue pour répondre à ces exigences.

1.1.2 Intégration des outils logiciels HAMSTERS et PetShop

Par rapport aux outils logiciels HAMSTERS et PetShop et afin de permettre l'exécution liée et simultanée d'ensembles de modèles des deux types et du prototype haute-fidélité correspondant, dans un environnement intégré, des briques logicielles supplémentaires sont nécessaires (Navarre, Palanque, Barboni, & Mistrzyk, 2007). La Figure 63 montre la décomposition de l'architecture fonctionnelle en différents modules :

- Le module HAMSTERS permettant l'édition des modèles de tâches et le module associé de simulation (ces deux modules forment l'outil logiciel HAMSTERS).
- Le module PetShop permettant l'édition des modèles de comportement du système et le module associé d'exécution du prototype très haute-fidélité et d'interprétation des modèles sous-jacents (ces deux modules forment l'outil logiciel PetShop).
- Les modules synergiques entre HAMSTERS et PetShop : « Correspondence editor » et « Simulation controller ».
- Le module d'édition, d'exécution et d'évaluation des sessions de formation: « Training module ».

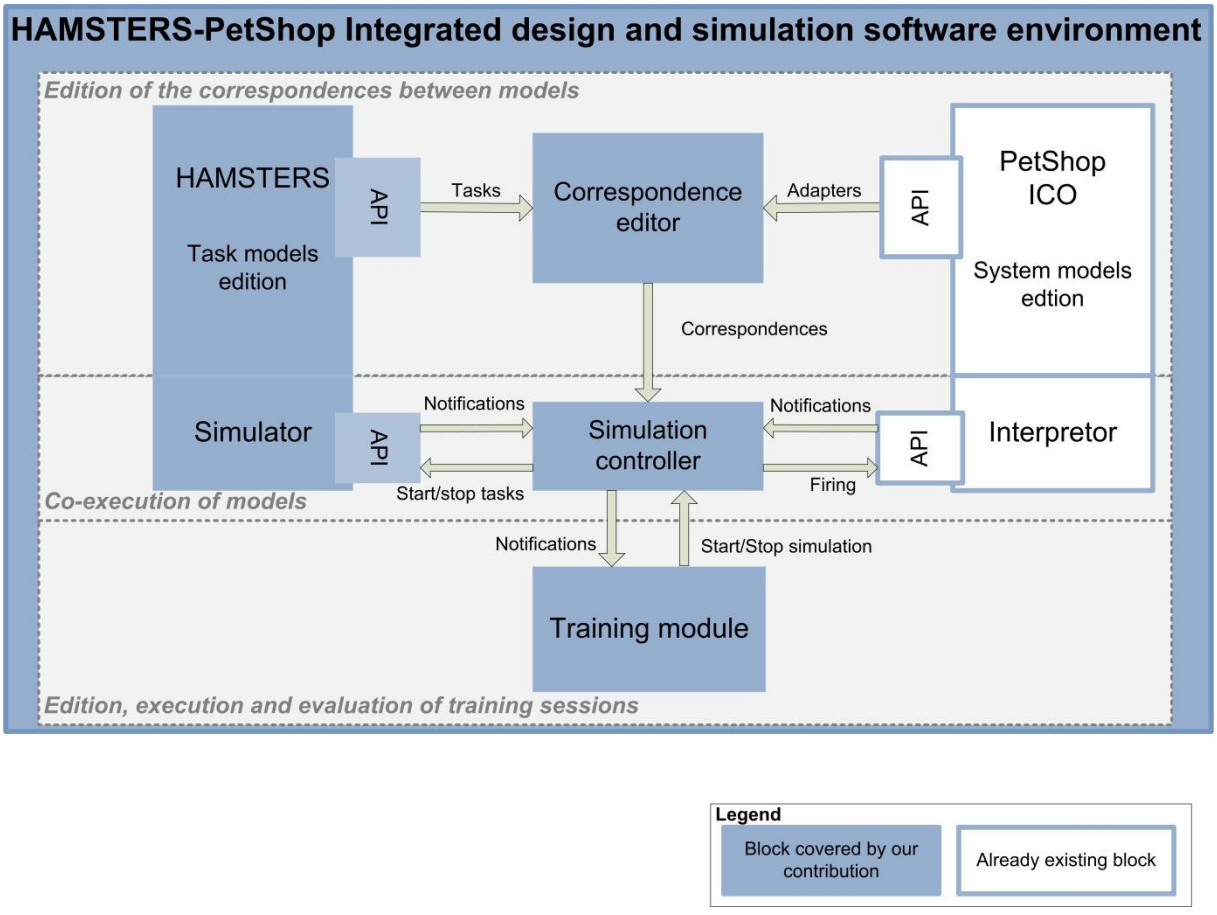


Figure 63. Architecture fonctionnelle de l'environnement d'exploitation synergique des modèles

Deux modules permettent d'exploiter de manière synergique les modèles de tâches et du système : le module « Correspondence Editor » permet d'éditer les correspondances entre deux modèles de types différents ; le module « Simulation controller » permet de co-simuler ou co-exécuter plusieurs modèles de différents types.

Le premier module « Correspondence Editor » possède deux fonctionnalités clés pour la mise en œuvre synergiques des modèles de tâches et du système :

- A partir des spécifications des tâches, il extrait l'ensemble des tâches interactives d'entrée et de sortie. Cet ensemble représente les manipulations qui peuvent être effectuées par l'utilisateur sur le système et les signaux issus du système portés à l'attention de l'utilisateur.
- A partir des spécifications ICO, il extrait les fonctions d'activation et de rendu. Ces fonctions représentent l'ensemble des évènements échangés entre les modèles de comportement du système et son prototype haute-fidélité.

Le principe d'édition des correspondances entre deux modèles est de lier, comme décrit sur la Figure 64:

- Une tâche interactive d'entrée (à partir du modèle de tâches HAMSTERS) avec une fonction d'activation (à partir du modèle ICO du système).
- Une fonction de rendu (à partir du modèle ICO du système) avec une tâche interactive de sortie (à partir du modèle de tâche HAMSTERS).

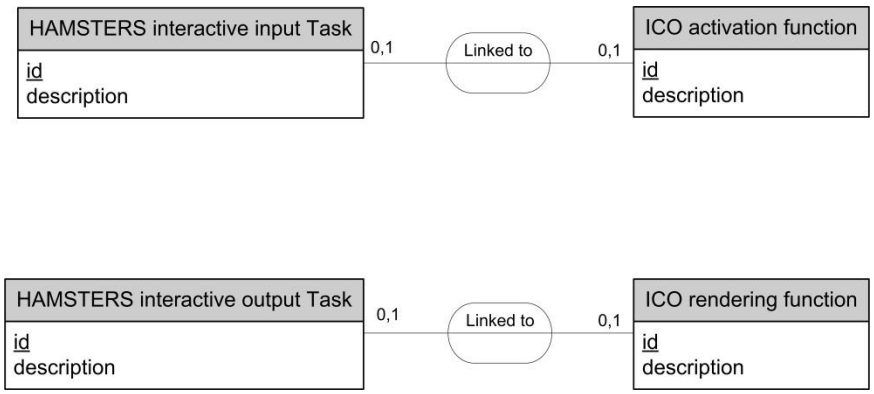


Figure 64. Métamodèle du « Correspondence editor »

La mise place cette correspondance, au moment de l'édition des modèles, permet de détecter :

- Les incohérences entre les tâches de l'utilisateur et les événements gérés et générés par le système.
- Les tâches interactives qui ne sont pas prises en charge par le système.
- Le rendu d'information non utile ou néfaste pour l'exécution des tâches.

Le second module « Simulation Controller » permet de contrôler l'exécution du prototype haute-fidélité et de ses modèles sous-jacents de types différents. Il possède trois modes d'exécution dans le sens où son exécution peut être dirigée par :

- Les interactions menées par l'utilisateur avec le prototype.
- Les interactions menées par l'utilisateur avec la simulation des modèles de tâches.
- Les interactions menées par l'utilisateur avec un scénario préenregistré de séquence d'actions issues des modèles de tâches.

Ce module utilise les données de correspondance entre les modèles et aiguille les évènements entre les moteurs d'exécution de PetShop et HAMSTERS.

1.2 Notation TEAM et environnement logiciel de conception rationalisée DREAM

La notation TEAM (Traceability, Exploration and Analysis Method) et son environnement d'édition logiciel DREAM (Design Rationale Environment for Argumentation and Modeling) ont été proposés par (Lacaze, Palanque, Barboni, Bastide, & Navarre, 2006) pour promouvoir et fournir un cadre conceptuel à l'exploration systématique des choix de conception tout au long du processus de développement d'un système interactif critique (Palanque & Lacaze, 2007). Ils ont pour but d'aider les ingénieurs et concepteurs de systèmes interactifs critiques dans leurs choix de conception en fournissant une structure d'aide à la décision ainsi qu'un moyen de capitaliser leurs connaissances sur les choix effectués précédemment et le cas échéant de décider ou non de réutiliser un choix de conception en fonction de changements dans les critères.

1.1.1 Description de la notation existante

La notation TEAM est une extension de la notation de conception rationnalisée QOC (Question Option Criteria) de (MacLean, Young, Bellotti, & Moran, 1991). Elle permet de décrire les différentes options en réponse à une question sur la conception d'un système ainsi que les différents critères utilisés pour sélectionner l'option retenue. La notation TEAM, grâce aux éléments décrit sur la Figure 65 permet de structurer et modéliser les interrogations et choix suivants (effectués lors de réunions de conception par exemple) :

- Les questions émises.
- Les options de conceptions qui ont été investiguées et celles qui ont été sélectionnées.
- Les évaluations effectuées pour les différentes options.
- L'ensemble des critères qui ont été utilisés pour évaluer les options considérées.
- L'ensemble des facteurs qui ont été pris en compte et la manière dont ils sont liés aux critères.
- Les modèles de tâches correspondants aux options.
- Les scénarios issus des modèles de tâches utilisés pour quantifier, pour chaque option, le pourcentage de remplissage d'un critère.
- Les arguments ayant influencé la conception.

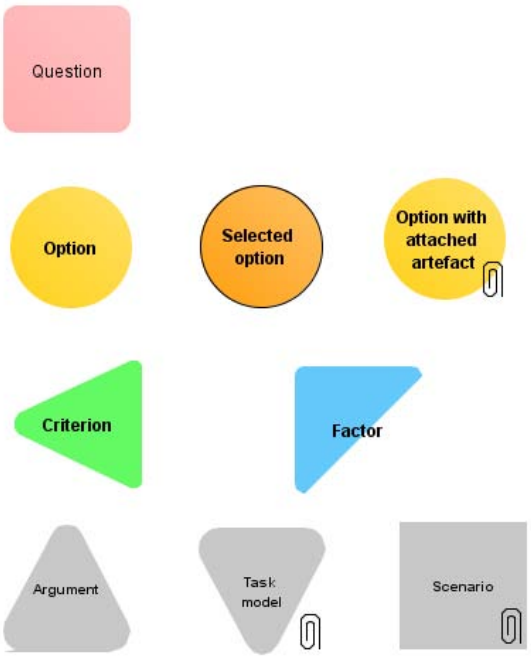


Figure 65. Eléments de la notation TEAM

La Figure 67 est un méta-modèle de la notation TEAM et décrit les relations entre chaque type d'entité de la notation. La Figure 66 montre l'environnement d'édition de diagramme TEAM.

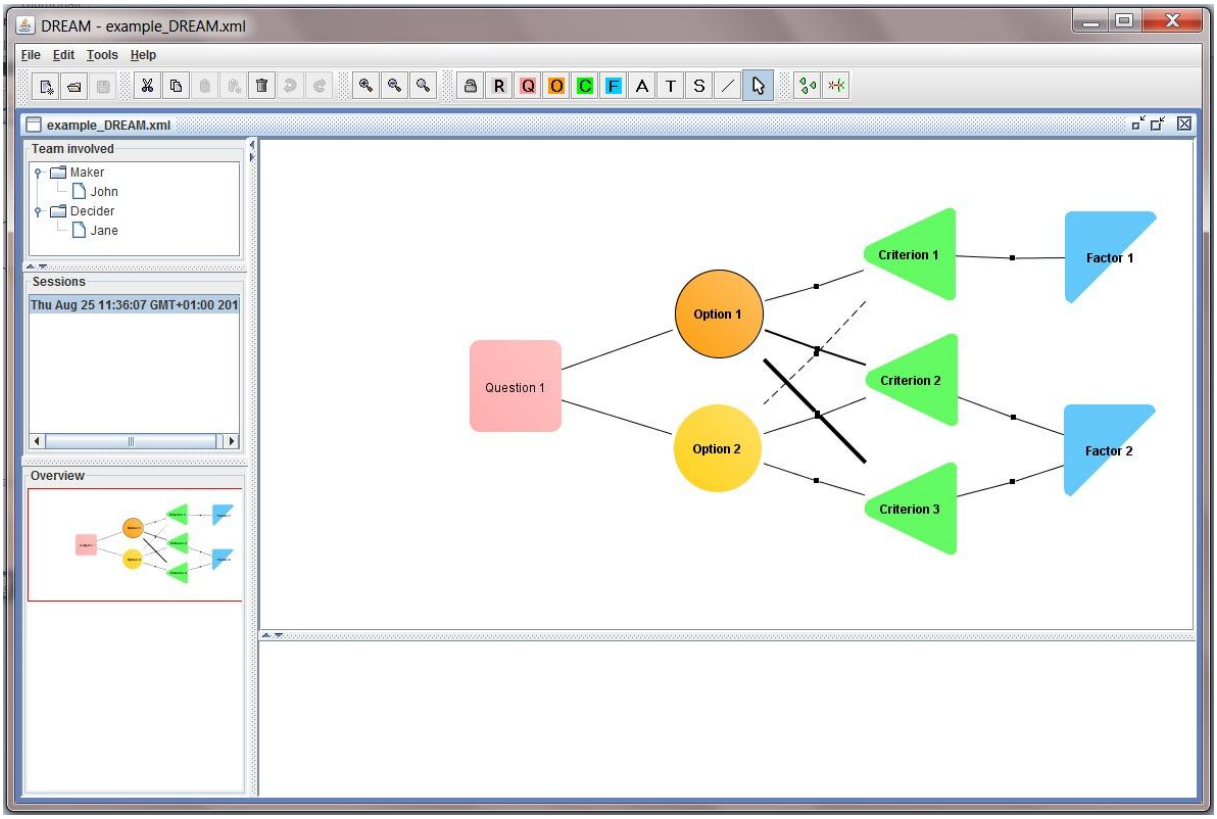


Figure 66. Capture d'écran de l'environnement DREAM

Cette notation et son environnement associé sont le produit d'une thèse et sont décrits de manière complète et détaillée dans (Lacaze, 2005).

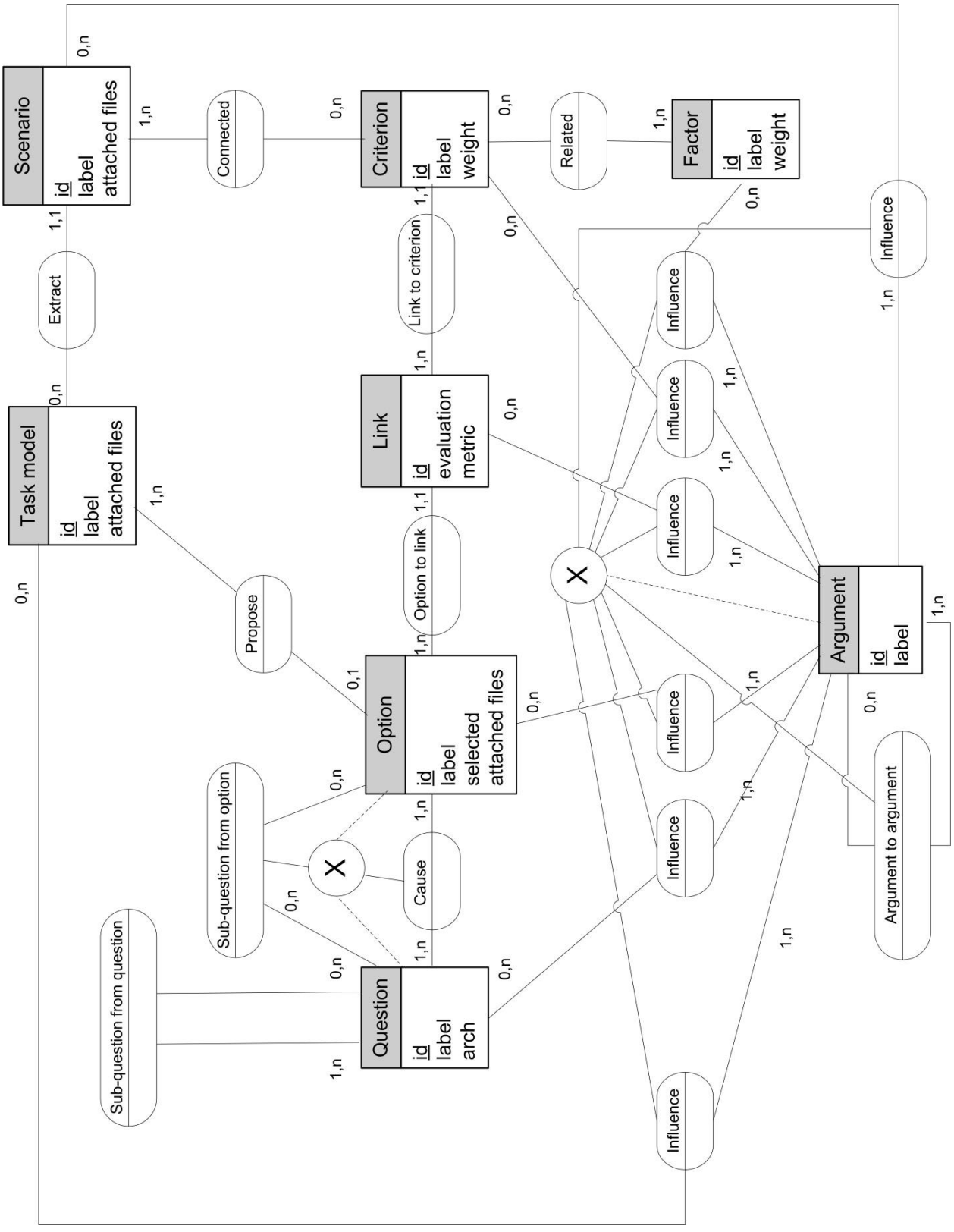


Figure 67. Diagramme d'entité relations des éléments de la notation TEAM

1.2.1 Extensions apportées à la notation et à l'environnement d'édition associé

Dans sa version initiale, la notation TEAM ne fournit pas d'éléments de représentation :

- Des exigences et besoins fonctionnels et non-fonctionnels liés aux choix de conception. La notation et l'environnement ne permettent pas de déterminer de manière directe le taux de couverture des besoins et exigences d'une option.
- De décomposition des facteurs en sous-facteurs. Les facteurs peuvent parfois être regroupés sous un facteur de plus haut niveau et la notation et l'environnement ne permettent pas une décomposition précise de ces éléments.

Nous proposons de modifier la notation pour ajouter ces possibilités de représentation. La Figure 68 décrit les éléments de la notation TEAM étendue. Il y apparaît notamment un nouvel élément « Requirements » pour décrire les besoins et exigences (bloc rectangulaire en haut de la Figure 68).

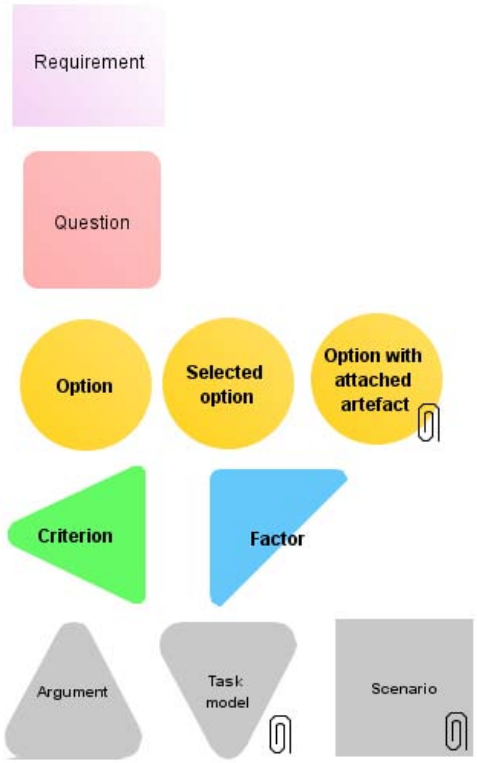


Figure 68. Eléments de la notation TEAM étendue

La Figure 69 décrit le nouveau diagramme d'entités relations des éléments de la notation TEAM étendue pour permettre de :

- Décrire les besoins et exigences (entité « Requirements » sur le diagramme).
- Décomposer les besoins ou exigences en besoins et exigences de plus bas niveau (relation « Divided » sur le diagramme).
- Décrire les questions de conception soulevées par un besoin ou une exigence (relation « Raise » sur le diagramme).
- Décrire les besoins ou exigences satisfaits par une option de conception (relation « Satisfy » sur le diagramme).
- Décomposer un facteur en sous-facteurs (relation « Decomposed » sur le diagramme)

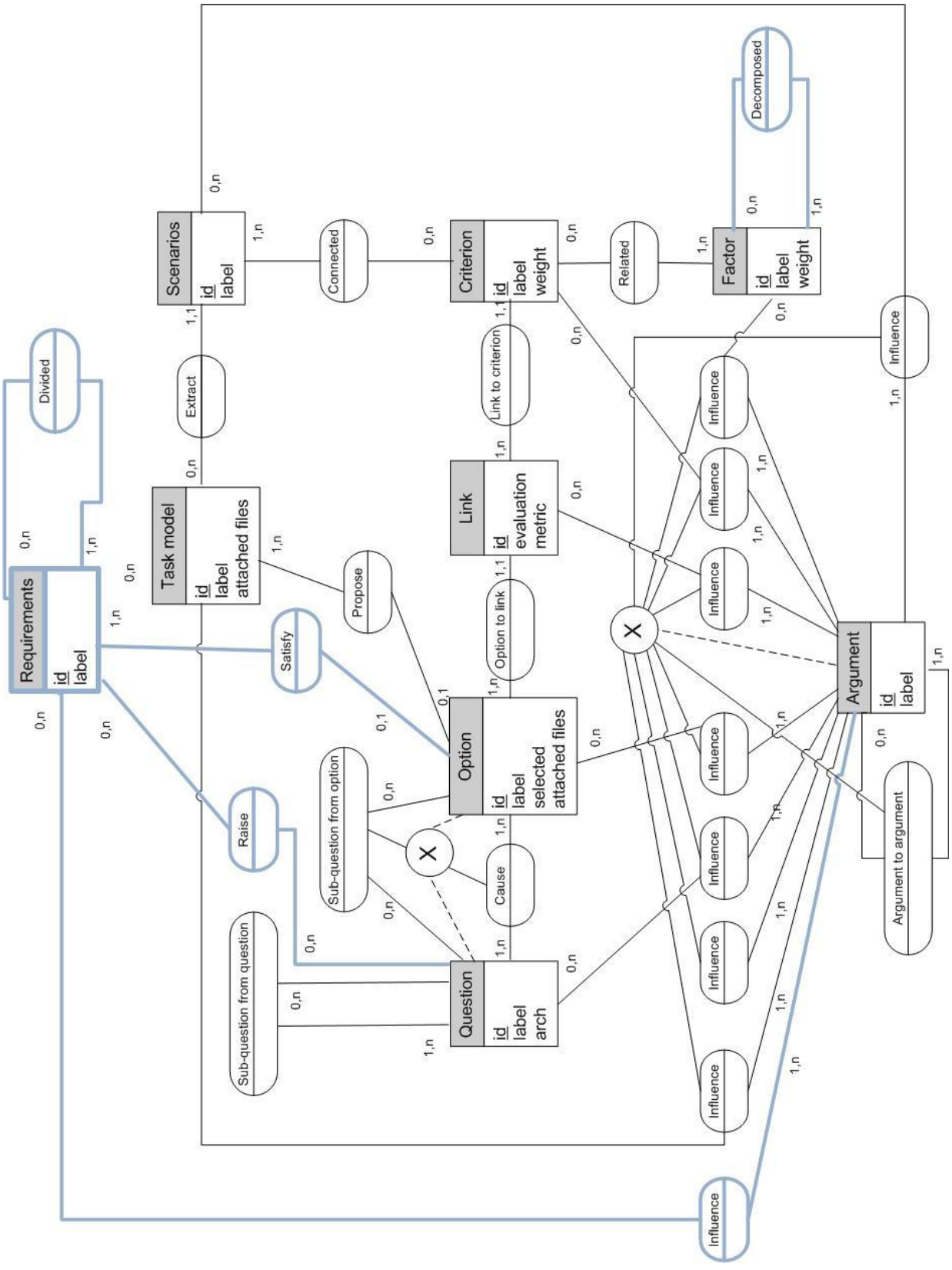


Figure 69. Diagramme d'entités relations des éléments de la notation TEAM étendue

1.3 Architecture fonctionnelle globale de l'environnement de mise en œuvre

Pour conclure cette présentation de l'environnement de mise en œuvre, la Figure 70 offre une vision de l'architecture fonctionnelle globale.

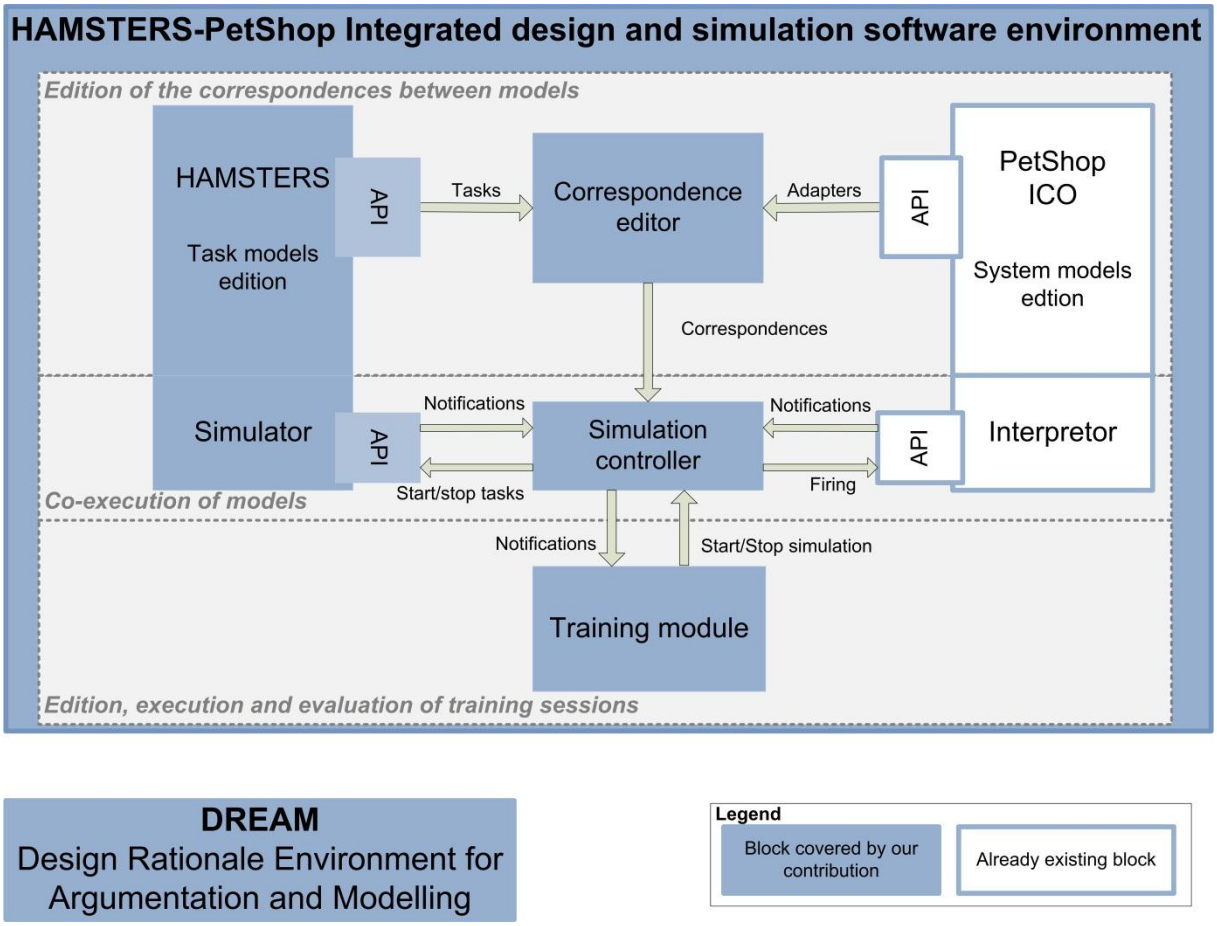
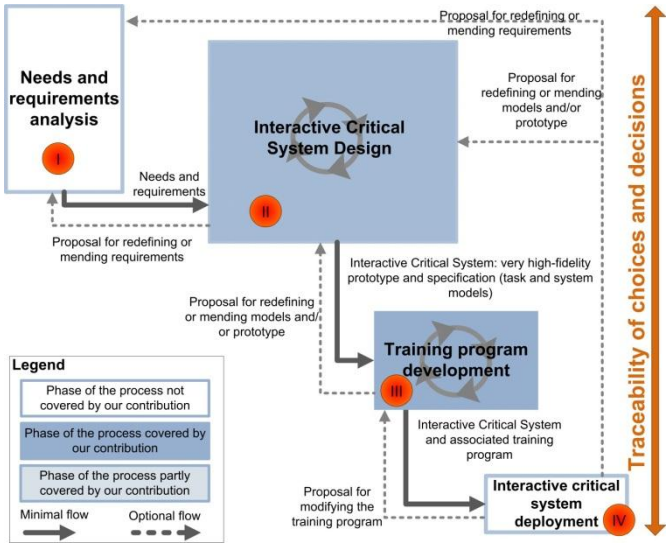


Figure 70. Architecture fonctionnelle de l'environnement de mise en œuvre du processus de développement d'un système interactif critique

2 Mise en œuvre des phases de l'approche de développement pour les systèmes interactifs critiques



Pour illustrer la mise en œuvre des phases de notre approche, nous utilisons des exemples portant sur les applications interactives de cockpit d'avions commerciaux.

Dans la phase de conception et développement de l'application et la phase de développement du programme de formation associé, nous utilisons l'exemple d'une application interactive développée pour les cockpits d'avions commerciaux civils, appelée WXR (pour Weather Radar System). Seule la sous-phase d'analyse qualitative des performances utilisateur ne porte pas directement sur cet exemple. En effet, l'exemple choisi permet de mettre en valeur la mise en œuvre de cette phase pour une technique d'interaction multimodale et peut s'appliquer à l'utilisation combinée des KCCU (Keyboard and Cursor Control Unit) par les pilote et co-pilote dans un cockpit. Dans la phase de traçabilité des exigences, nous utilisons l'exemple illustratif des widgets utilisés pour développer les applications interactives des cockpits.

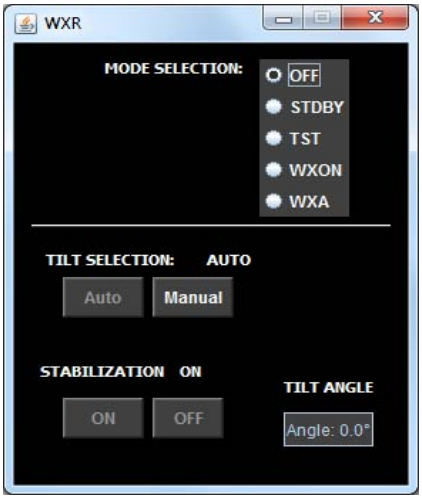


Figure 71. Capture d'écran de l'application WXR (Weather Radar)

La Figure 71 montre une capture d'écran de la partie graphique de cette application interactive. La partie supérieure (« MODE SELECTION ») permet aux membres de l'équipage de changer le mode du radar météorologique, alors que la partie inférieure (« TILT SELECTION », « STABILIZATION » et « TILT ANGLE ») permet d'ajuster l'angle d'orientation du radar.

2.1 Phase d'analyses des besoins et exigences et phase itérative de prototypage basse-fidélité

La première phase du processus de développement permet d'analyser et de compiler les besoins et exigences. Elle peut s'effectuer hors de notre environnement logiciel de mise en œuvre. Cet ensemble

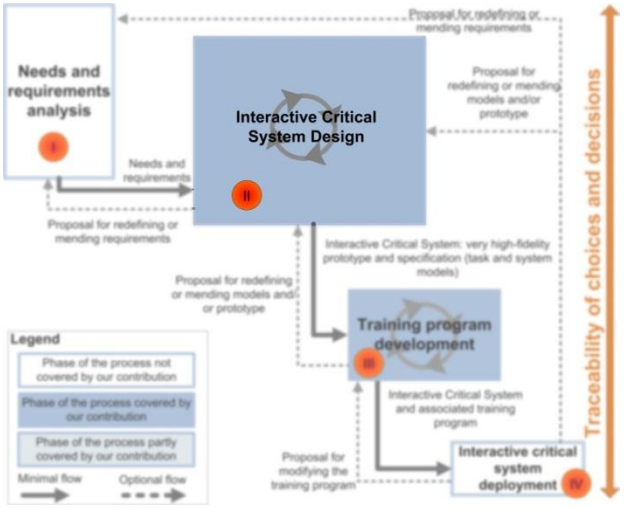
de besoins et d'exigences est le matériau d'entrée pour les phases suivantes. Notre environnement permet par la suite de tracer ces besoins et exigences par rapport aux différentes options de conception.

L'application WXR permet au pilote et au co-pilote de contrôler le radar météorologique de l'avion. Ce radar se situe sur le nez de l'avion et est fréquemment utilisé lorsque les conditions météorologiques sont mauvaises. Ce radar fournit des informations sur les perturbations météorologiques dans une zone conique dont le sommet se trouve sur le nez de l'avion. Les pilotes doivent donc fréquemment changer l'angle pour connaître les différentes zones de perturbation autour de l'avion et changer si nécessaire la trajectoire. L'application WXR doit leur permettre :

- De changer le mode du radar météorologique.
- De visualiser dans quel mode le radar météorologique se trouve.
- De changer l'angle du radar.
- De visualiser l'angle actuel du radar.

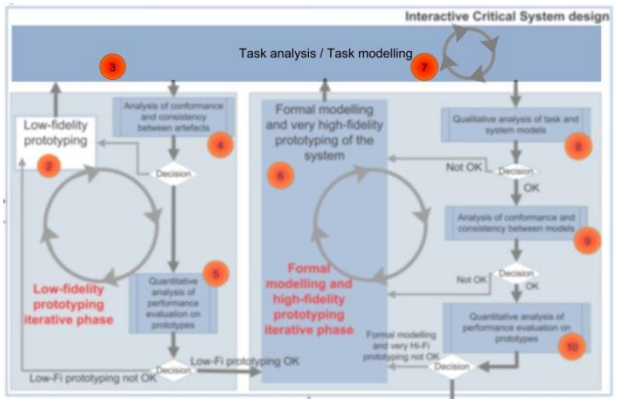
Dans les sous-sections suivantes, nous détaillons les phases du processus couvertes par notre contribution : la phase de conception du système interactif critique, la phase de développement du programme de formation associé ainsi que la traçabilité des besoins et exigences tout au long du processus de développement. Chaque sous-section est assortie d'un exemple démonstratif sélectionné pour mettre en valeur chaque phase ou sous-phase du processus.

2.2 Phase de conception de l'application logicielle interactive critique



Cette section illustre de manière détaillée la phase de conception du système interactif critique ayant été présentée de manière abstraite dans le chapitre 5.

2.2.1 Analyse et modélisation des tâches utilisateur



Durant la phase d'analyse et de modélisation des tâches utilisateur, plusieurs techniques peuvent être utilisées (observations, documentation, rédaction de scénarios d'utilisation, questionnaires,…) pour rassembler les informations sur les tâches utilisateurs. Ces informations peuvent ensuite être structurées et enregistrées sous la forme de modèles de tâches grâce à l'outil logiciel HAMSTERS.

Comme le montre la Figure 72, les tâches de

haut niveau pour la gestion du radar météo (« manage WXR ») sont décomposées en deux tâches, « setup WXR » et « decide WXR is ready ». La tâche « setup WXR » représente les deux activités de réglage de l'orientation du radar météorologique et du mode, tandis que la tâche « decide WXR is ready » peut l'interrompre à tout moment.

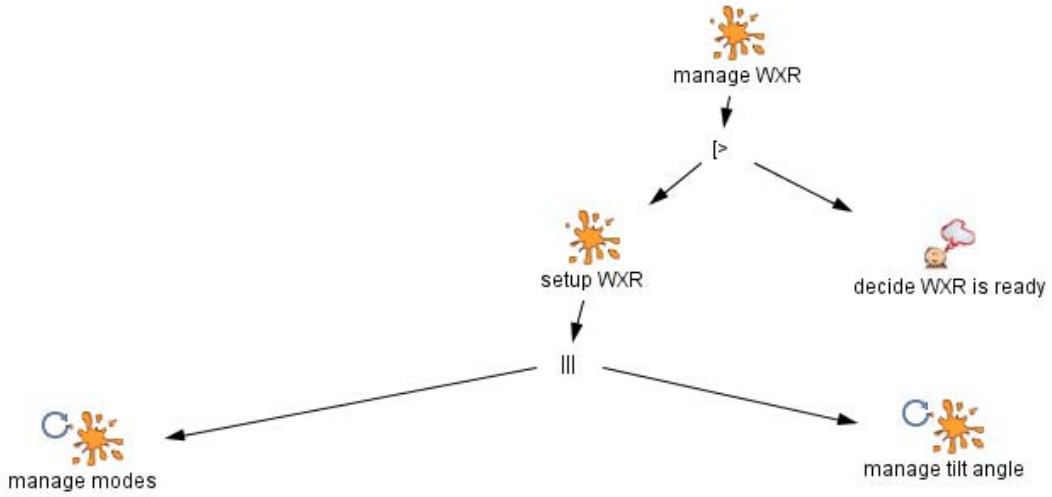


Figure 72 Ensemble haut-niveau des tâches pour la gestion du radar météo

Les deux tâches abstraites « «manage mode » et « manage tilt angle » sont détaillées dans les Figure 73 et Figure 74. Ces deux tâches sont activées par les deux tâches cognitives « decide change mode » et « decide change tilt angle » pour représenter l'activité de décision effectuée par les membres d'équipage avant d'interagir avec le système. Les membres de l'équipage peuvent basculer entre cinq modes du radar météo (WXON, OFF, STDBY, TST et WXA) comme le montre la Figure 73.

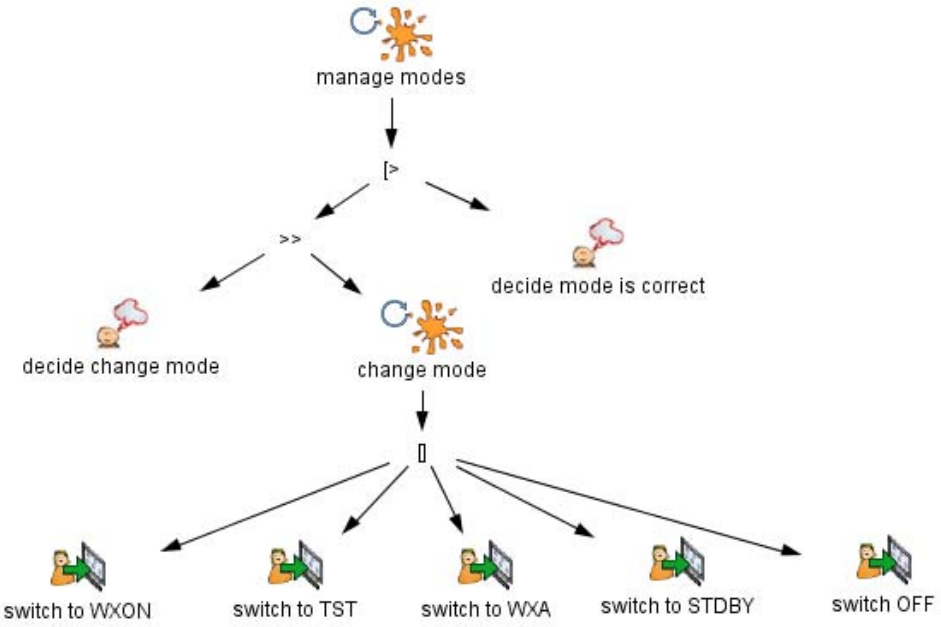


Figure 73 Ensemble détaillé des sous tâche de la tâche « change_mode »

La tâche « change mode » (Figure 73) englobe les sous-tâches suivantes : « switch to WXON », « switch to TST », « switch to WXA », « switch to STDBY », « switch OFF ». Ces tâches interactives

d'entrée représentent les actions possibles des membres d'équipage sur la partie de sélection du mode de l'application.

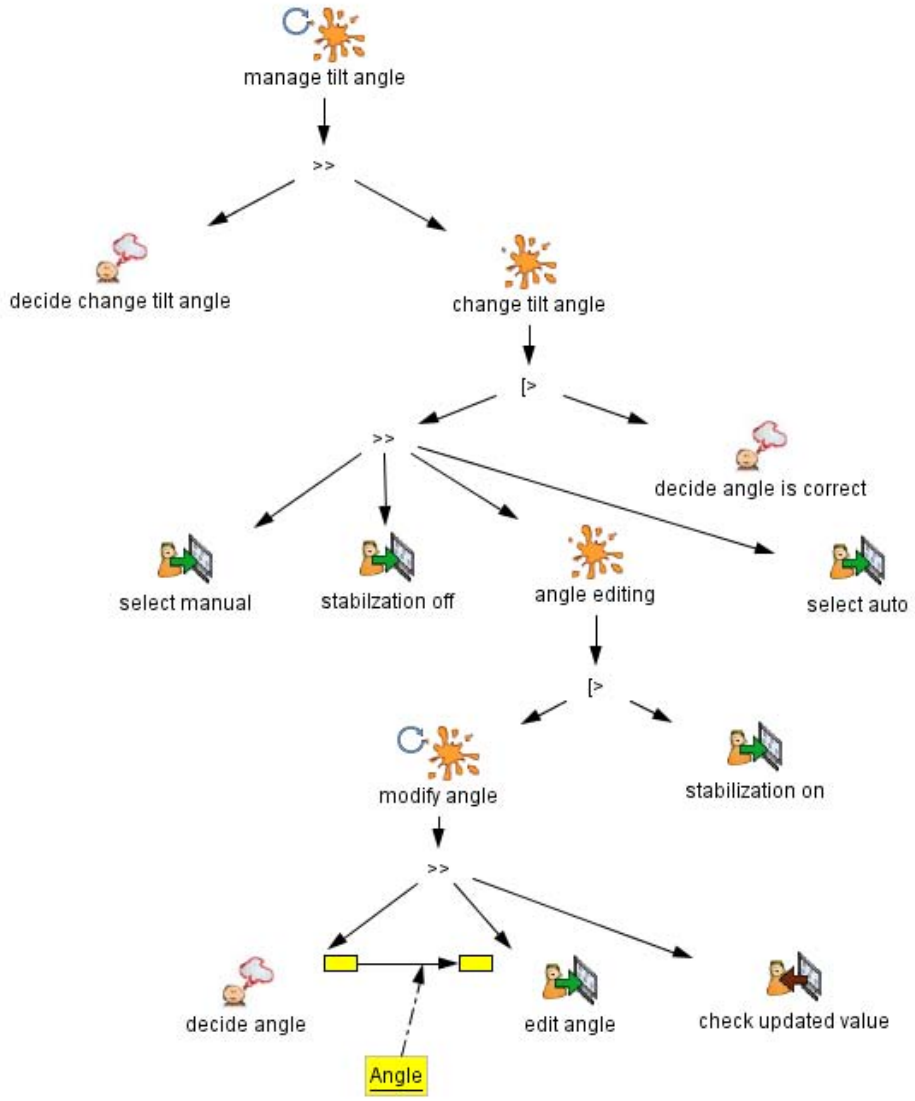


Figure 74 Sous tâches de la tâche abstraite « manage_tilt_mode »

Comme le montre la Figure 74, les membres de l'équipage peuvent ajuster l'orientation (« tilt angle ») du radar météo en cas de besoin (l'idée principale étant de pouvoir utiliser cette fonctionnalité uniquement si nécessaire, la plupart du temps, le comportement par défaut est le bon). Si les deux premières tâches sont de simples tâches interactives d'entrée, changer l'angle d'inclinaison manuellement implique quatre sous-tâches :

- La tâche cognitive « decide angle » est le choix d'une valeur par les membres d'équipage.
- La tâche interactive d'entrée « edit angle » est l'interaction pour l'édition de la valeur dans le système de l'avion.
- La tâche système « check updated value » est la tâche interactive de sortie de l'utilisateur pour vérifier que la valeur éditée a été prise en compte par le système.

La simulation de ces modèles avec le simulateur HAMSTERS permet de vérifier que les modèles sont conformes aux scénarios d'utilisation.

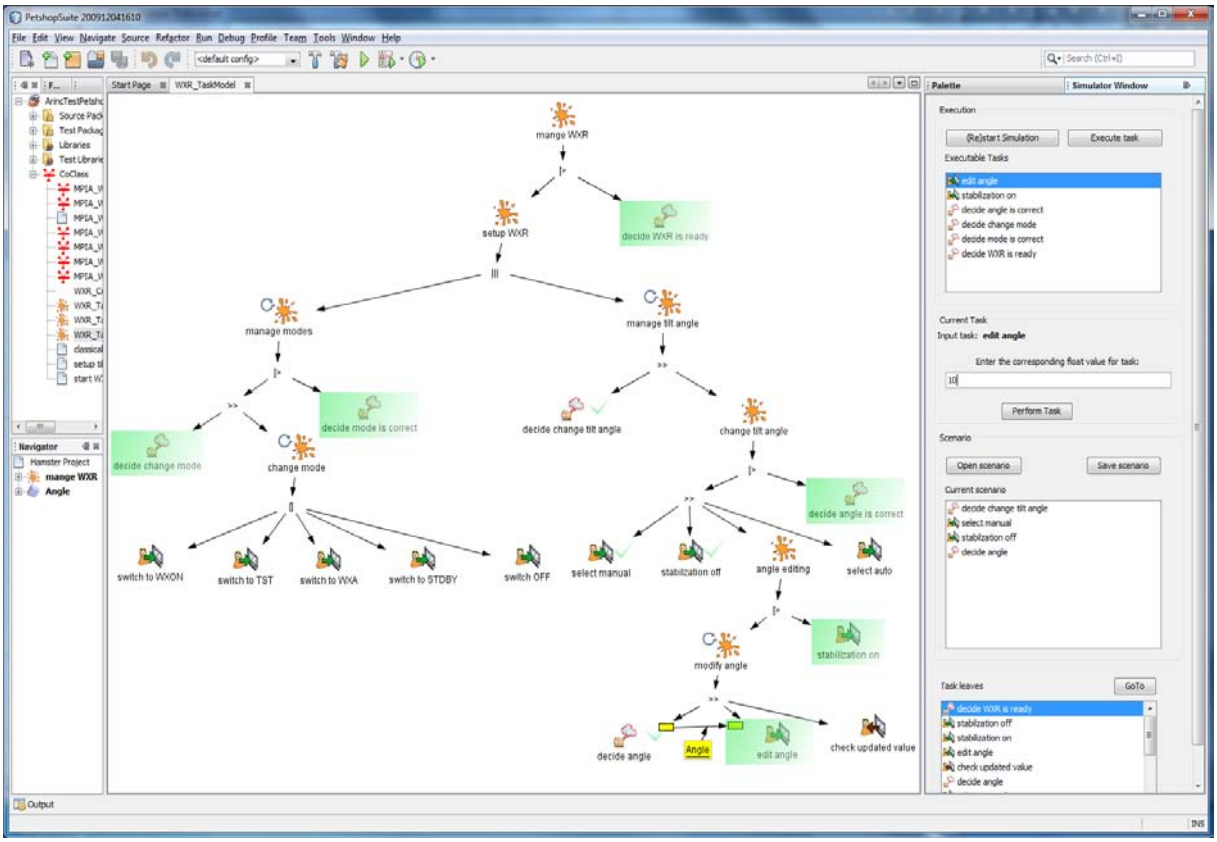


Figure 75. Simulation de la première itération du modèle de tâches utilisateur de gestion du radar météorologique

A l'issue des premiers tests de simulation du modèle de tâche, en examinant les scénarios produits avec HAMSTERS et en les comparant aux informations issues de l'analyse des tâches, il apparaît que l'activité périodique de test de l'application devant être effectuée par les membres de l'équipage n'apparaît pas dans cette première version du modèle. Une seconde version du modèle est donc produite pour spécifier cette activité (Figure 76).

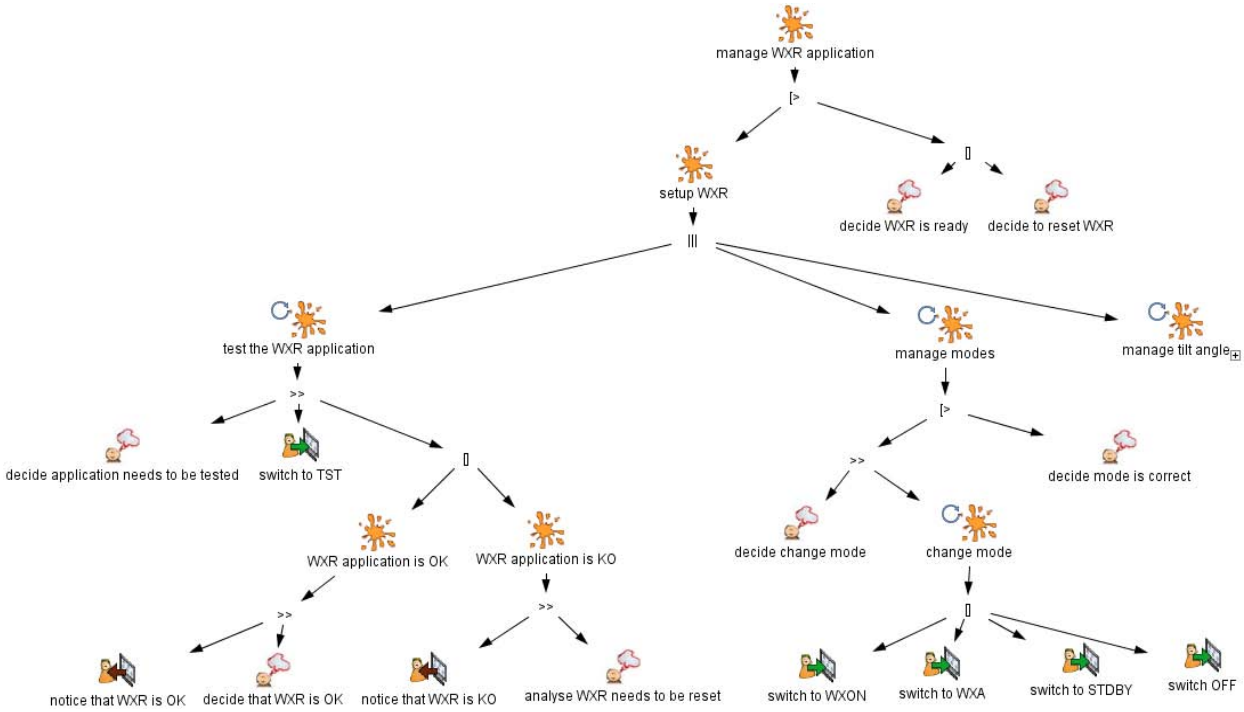
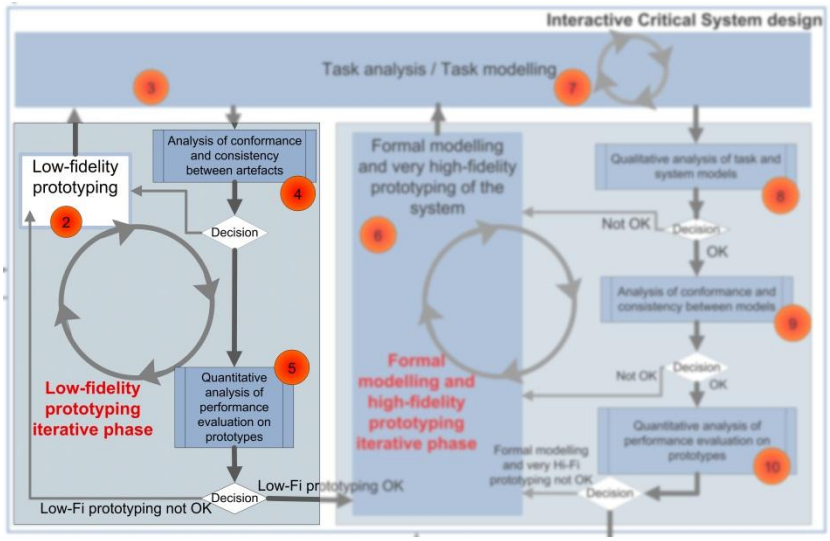


Figure 76. Seconde itération du modèle de tâche

La Figure 76 présente une nouvelle version du modèle de tâches basée sur la première version décrite précédemment. Le sous arbre droit du modèle chapeauté par la tâche abstraite « manage tilt angle » a été occulté (ou replié, symbole ⊞ situé en bas à droite de la description de la tâche) temporairement pour permettre une meilleure lisibilité du sous arbre « test the WXR application » dont ce paragraphe fait l'objet. Cette nouvelle version intègre la sous-tâche abstraite et répétitive de test de l'application « test the WXR application » et le sous arbre des tâches correspondantes (sur la partie gauche du modèle). Ce sous arbre de tâches permet d'indiquer que l'utilisateur décide de tester l'application (tâche cognitive « decide application needs to be tested ») puis sélectionne le mode « TST » de l'application (tâche interactive d'entrée « switch to TST »). Ensuite deux choix sont possibles : si l'application fonctionne correctement (tâche interactive de sortie « notice that WXR is OK » et tâche cognitive « decide that WXR is OK »), l'utilisateur peut poursuivre ses activités jusqu'au prochain test, sinon, si l'application ne fonctionne pas correctement (tâche interactive de sortie « notice that WXR is KO » et tâche cognitive « analyse WXR needs to be reset »), l'utilisateur décidera une remise à zéro de l'application (tâche cognitive « decide to reset WXR » en haut à gauche du modèle qui (avec la tâche cognitive « decide WXR is ready ») conditionne l'interruption de l'activité principale de réglage de l'application « setup WXR ».

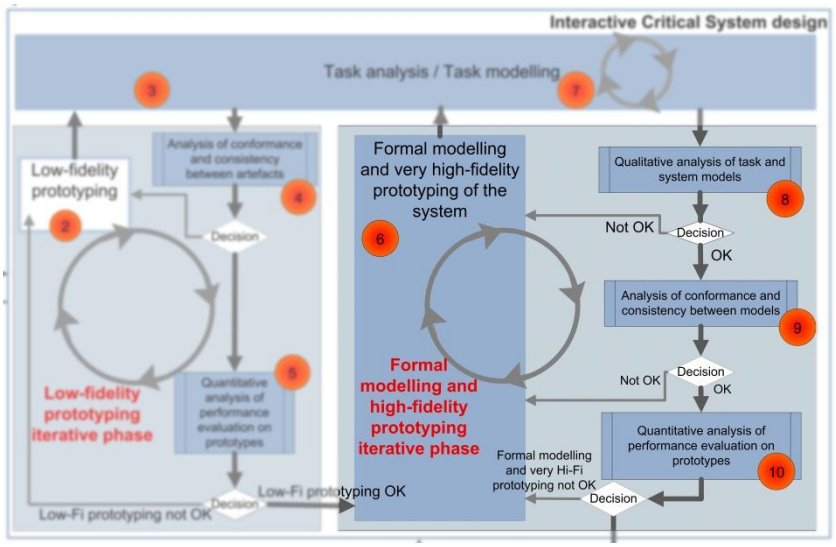
Lorsque les scénarios générés à partir de ce nouveau modèle sont conformes aux scénarios d'utilisation construits à partir des éléments analysés, la phase itérative de prototypage basse-fidélité peut être amorcée. Nous soulignons le fait que la description des tâches doit être exhaustive et détaillée afin de consigner finement les tâches d'interaction avec le système ainsi que les données échangées entre l'utilisateur et le système. Cette précision est nécessaire pour les phases suivantes et notamment la mise en correspondance entre les tâches interactives et les évènements d'entrée/sortie du système.

2.2.2 Phase itérative de prototypage basse-fidélité du système



Plusieurs techniques peuvent être utilisées pour le prototypage basse-fidélité et nous ne détaillerons pas cette étape car elle est réalisée hors de l'environnement de mise en œuvre du processus de développement. Cependant, lors de la phase itérative de prototypage, les prototypes basse-fidélité réalisés sont confrontés aux modèles de tâches HAMSTERS ainsi qu'aux scénarios d'utilisation afin d'analyser qualitativement la cohérence des prototypes proposés avec les modèles de tâches. Si la cohérence est assurée, une analyse quantitative avec des performances des utilisateurs sur les prototypes basse-fidélité est menée et permet de déterminer les éléments retenus des prototypes basse-fidélité pour la phase suivante. Dans le cas de notre exemple illustratif, portant sur une application interactive simple, nous considérons que le prototype basse-fidélité s'apparente à la capture d'écran de la Figure 71.

2.2.3 Phase itérative de modélisation formelle et de prototypage très haute-fidélité du système



Durant cette phase, l'environnement intégré de mise en œuvre est mis à profit pour :

- Spécifier les modèles ICO du système en fonction des exigences et besoins des utilisateurs et des éléments retenus à l'issue de la phase itérative de prototypage basse-fidélité.
- Editer les correspondances entre les modèles ICO et les modèles de tâches. Tant que la mise en correspondance révèle des non-conformités ou des incohérences, les modèles ICO du prototype très haute-fidélité doivent être modifiés dans le respect des objectifs requis sur les propriétés d'utilisabilité et de fiabilité du système en développement.

La Figure 77 résume le processus itératif de modélisation et de mise en correspondance des modèles.

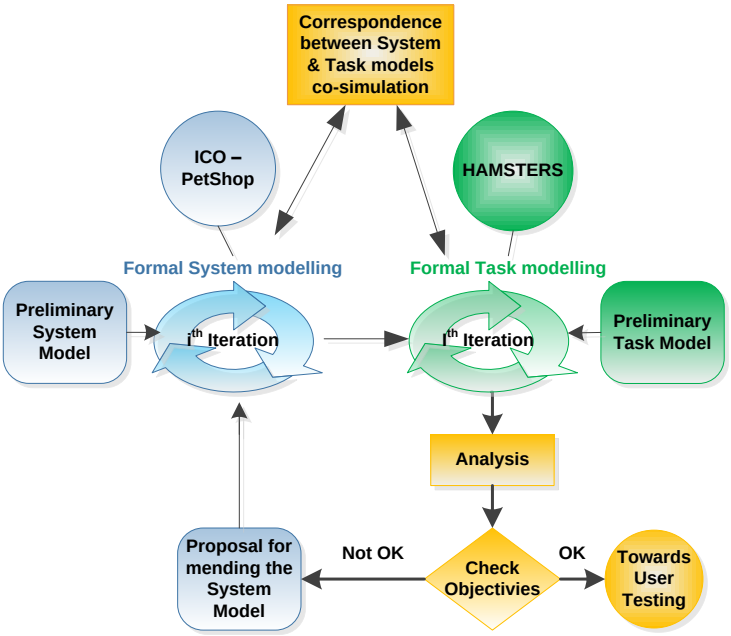


Figure 77. Conception itérative des modèles des tâches utilisateur et du prototype très haute-fidélité

L'environnement intégré de mise en œuvre permet, en suivant ce processus itératif, de valider et vérifier la cohérence entre les modèles et ainsi la synergie entre les tâches utilisateur et comportement du système car il fournit un support pour :

- Evaluer la compatibilité structurelle entre les modèles lors de l'édition des correspondances.
- Vérifier si les scenarios d'utilisation sont réalisables avec le système et vérifier la cohérence entre les tâches réalisables et l'état du système (par exemple, dans le cas d'un guichet bancaire automatique on pourra vérifier que les billets ne sont pas distribués par le système tant que l'utilisateur n'a pas repris sa carte bancaire et ainsi éviter à l'utilisateur une erreur de post-complétion).
- Construire et enregistrer des scénarios d'utilisation en utilisant le prototype très haute-fidélité.
- Identifier les tâches réalisables en fonction de l'état du système.
- Diriger l'exécution du prototype très haute-fidélité par les modèles de tâches.

Cette intégration permet une évolution en étroite collaboration entre les modèles des deux types car l'exécution de modèles d'un type impacte l'exécution des modèles de l'autre type.

2.2.3.1 Modélisation formelle de l'application interactive

Comme détaillé dans la sous-section 2.1.2 du chapitre 2, l'édition des modèles ICO consiste à :

- Modéliser formellement le comportement du système (partie Dialogue de la vision architecturelle Arch ou de la figure présentée dans la sous-section 1.1 du chapitre 1).
- Développer la partie présentation de l'interface utilisateur correspondante (partie présentation de la vision architecturelle Arch ou de la figure présentée dans la sous-section 1.1 du chapitre 1).
- Développer les fonctions d'activation.
- Développer les fonctions de rendu.

Comportement du système

Les Figure 78 et Figure 79 représentent les modèles de comportement des deux fonctions de l'application WXR.

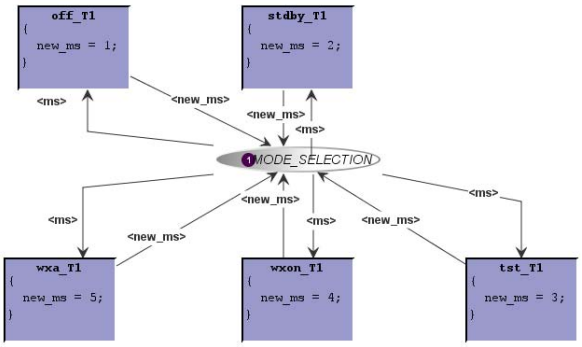


Figure 78. Modèle ICO de la fonctionnalité de changement de mode de l'application WXR

Le modèle de comportement décrit par le réseau de Petri de la Figure 78 gère les événements provenant des 5 boutons de sélection exclusive ou boutons radio (Figure 71). La sélection courante (une valeur entière de 1 à 5) est stockée dans le jeton de la place MODE_SELECTION et correspond à la sélection d'un des boutons radio (OFF, STDBY, TST, WXON, WXA). Le jeton est modifié par les transitions (new_ms = 3 par exemple) en utilisant des variables sur les arcs entrants et sortants comme paramètres formels.

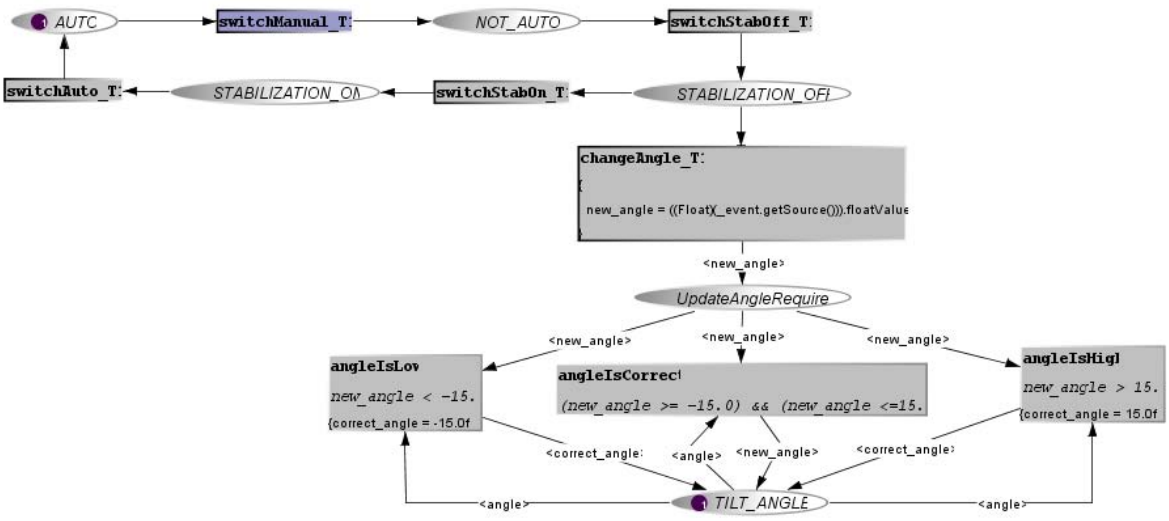


Figure 79. Modèle ICO de la fonctionnalité d'édition de l'angle d'orientation du radar météorologique

Le réseau de Petri de la Figure 79 gère les événements provenant des 2 boutons et de la zone d'édition. Dans l'état courant, cette partie de l'application est dans l'état automatique (un jeton est dans la place AUTO et un jeton est dans la TILT_ANGLE). Dans cette configuration, il n'est pas possible pour l'utilisateur de changer la valeur de l'angle du radar météorologique.

Présentation

La partie graphique de l'interface utilisateur retenue et développée est celle exposée dans la Figure 71.

La partie présentation correspond à la présentation logique, cachée par un ensemble de méthodes de rendu (afin de représenter les changements d'état et la disponibilité des gestionnaires d'événements) et un ensemble d'événements utilisateur, intégré dans une interface logicielle (Figure 80).

```
Public interface WXR_APP extends ICOWidget {

  //List of user events.

  public enum WXR_PAGE_events {asked_off,
asked_stdby, asked_wxa, asked_wxon, asked_tst,
asked_auto asked_stabilization, asked_changeAngle}

  //List of activation rendering methods.

  void
setWXRModeSelectEnabled(WXR_PAGE_events,
List<ISubstitution>);

  void setWXRTiltSelectionEnabled
(WXR_PAGE_events, List<ISubstitution>);

  //List of rendering methods.

  void showModeSelection (IMarkingEvent anEvent);

  void showTiltAngle (IMarkingEvent anEvent);

  void showAuto (IMarkingEvent anEvent);

  void showStab (IMarkingEvent anEvent);

}
```

Figure 80 Interface logicielle de la page WXR de l'application WXR

Fonction d'activation

Dans le cas de notre exemple illustratif, l'interaction de l'utilisateur vers le système (entrée) a lieu grâce à des composants d'interface graphique. Quand un événement utilisateur est déclenché, la fonction d'activation est notifiée et requiert que le modèle ICO tire le gestionnaire d'événements correspondant en intégrant la valeur reçue par l'événement utilisateur. Lorsque l'état d'un gestionnaire d'événements change (i.e. devient disponible ou non disponible), la fonction d'activation est notifiée (par l'intermédiaire de l'observateur et le mécanisme d'événement présenté ci-dessus) et appelle la méthode de rendu d'activation correspondante de la partie présentation en intégrant les valeurs du gestionnaire d'événements.

La Table 15 montre la fonction d'activation pour la page WXR.

Table 15. Fonction d'activation de l'application WXR

Objets graphique d'interaction	Evènements utilisateurs	Gestionnaires d'évènement	Méthodes de rendu d'activation
Bouton radio OFF	asked_off	off	setWXRModeSelectEnabled
Bouton radio STDBY	asked_stdby	stdby	setWXRModeSelectEnabled
Bouton de sélection d'option TST	asked_tst	tst	setWXRModeSelectEnabled
Bouton de sélection d'option WXON	asked_wxon	wxon	setWXRModeSelectEnabled
Bouton de sélection d'option WXA	asked_wxa	wxa	setWXRModeSelectEnabled
Boutons poussoir AUTO et MAUAL	asked_auto	switchAUTO	setWXRTiltSelectionEnabled
Boutons poussoir ON et OFF	asked_stabilization	switchSTABILIZATION	setWXRTiltSelectionEnabled
Zone d'édition	asked_changeAngle	changeAngle	setWXRTiltSelectionEnabled

Chaque ligne de cette table décrit les objets qui prennent part aux processus d'activation. La première ligne, par exemple, décrit la relation entre le bouton de sélection « OFF », l'événement utilisateur « asked_off » (produit en cliquant sur le bouton de sélection), le gestionnaire d'événement « off » (représentée dans le modèle par la transition off_t1) et la méthode de rendu d'activation « setWXRModeSelectEnabled » de la partie présentation.

Plus précisément dans cette Table 15 :

- Lorsque le gestionnaire d'événement « off » devient activé, la fonction d'activation appelle la méthode de rendu d'activation « setWXRModeSelectEnabled » lui fournissant des données sur l'activation du gestionnaire d'événements. Du côté de l'interaction physique, cet appel de méthode conduit à l'activation du composant graphique correspondant (par exemple la présentation de la case à cocher off comme étant disponible à l'utilisateur).
- Lorsque le bouton « OFF » de la partie présentation est pressé, la partie présentation déclenche l'événement appelé « asked_off ». Cet événement est reçu par la fonction d'activation qui exige de la partie comportement de déclencher le gestionnaire d'événement « off » (à savoir la transition off_T1 dans le réseau de Petri de la Figure 78).

Fonctions de rendu

L'interaction du système vers l'utilisateur (sortie) présente à l'utilisateur les changements d'état qui se produisent dans le système. La fonction de rendu maintient la cohérence entre l'état interne du système et de son apparence extérieure en reflétant les changements d'états du système sur l'interface utilisateur. En effet, lorsque l'état du modèle ICO change (par exemple un changement de marquage pour au moins une place), la fonction de rendu est notifiée (via l'observateur et le mécanisme

d'événements) et appelle la méthode de rendu correspondante de la partie de présentation avec la valeur des jetons ou du franchissement en tant que paramètres.

Table 16. Fonction de rendu de l'application WXR

Elément du modèle de comportement (places/transitions)	Evénement du modèle de comportement	Méthode de Rendu
MODE_SELECTION	token_enter	showModeSelection
TILT_ANGLE	token_enter	showTiltAngle
AUTO	marking_reset	showAuto
AUTO	token_enter	showAuto
AUTO	token_remove	showAuto
STABILIZATION_ON	marking_reset	showStab
STABILIZATION_ON	token_enter	showStab
STABILIZATION_ON	token_remove	showStab

La Table 16 présente les fonctions de rendu de l'application WXR dans un tableau où chaque ligne présente les objets qui participent au processus de rendu. Par exemple, la première ligne montre le lien entre la place « MODE_SELECTION », l'événement lié à cette place (un jeton entre dans la place) et la méthode de rendu « showModeSelection » du composant de la partie présentation. Il peut être lu comme suit : quand un jeton entre dans la place « MODE_SELECTION », la fonction de rendu est notifiée et la méthode de rendu « showModeSelection » est appelée avec des données relatives au nouveau marquage de cette place qui sont utilisés en tant que paramètres de la méthode de rendu.

Une fois la modélisation formelle et le développement de l'application effectuée, l'exécution du prototype très haute-fidélité permet de valider que l'application est conforme aux besoins et exigences émis dans les phases précédentes. La Figure 81 montre l'exécution de l'application et de son modèle de comportement sous-jacent. On peut aussi voir que l'ajout dynamique, en exécution, d'une place « p0 », sur le réseau de Petri de la partie supérieure, et sa connexion à la transition « off_T1 » change l'état du réseau. La transition « off_T1 » devient non franchissable et le bouton de sélection « OFF » est désactivé.

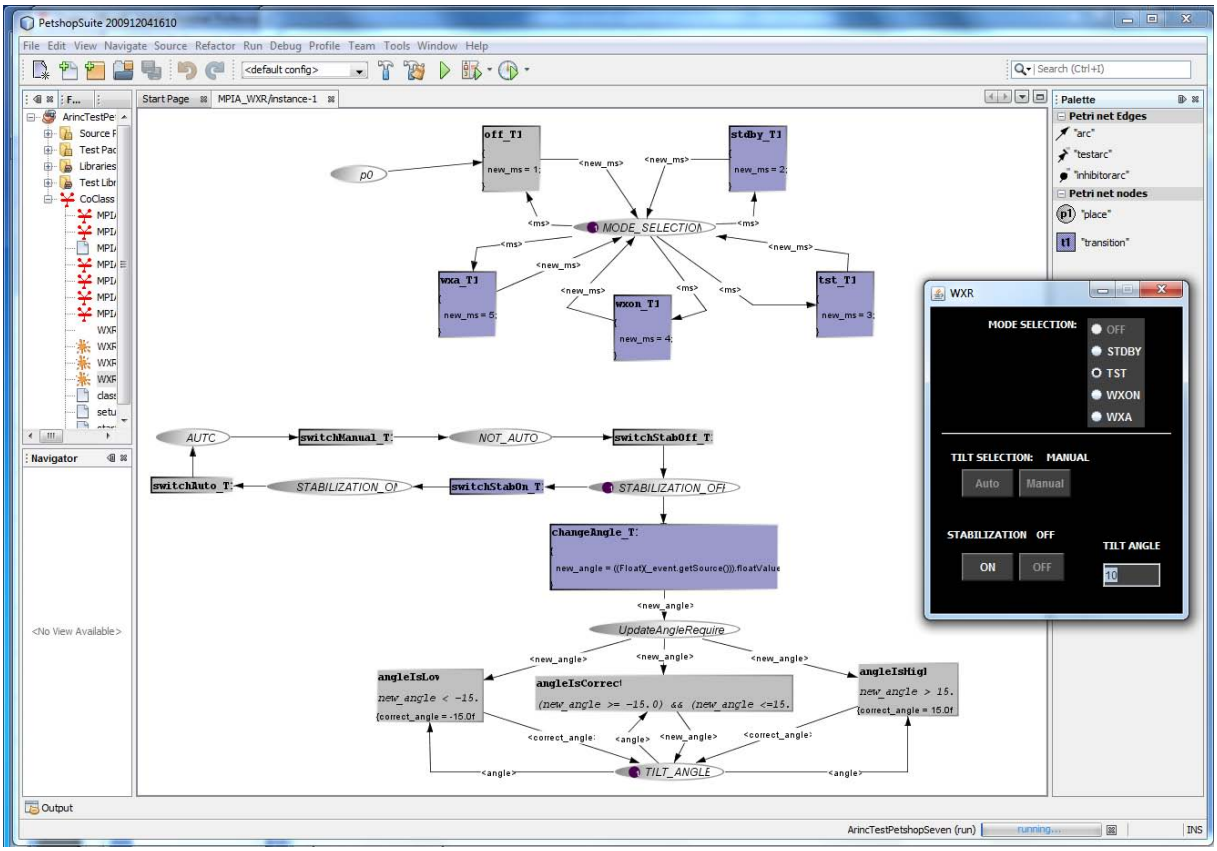


Figure 81. Copie d'écran de la fenêtre d'exécution du prototype très haute-fidélité et de son modèle formel correspondant

2.2.3.2 Mise en correspondance des modèles de tâches et du système

Dans cette section, nous présentons l'étape de mise en correspondance des modèles de tâches et des modèles du système afin d'analyser qualitativement si les tâches effectuées par l'utilisateur correspondent aux fonctions fournies par le système et aux interactions possibles avec le système.

Edition des correspondances

L'édition de correspondances entre les deux modèles est réalisée par un éditeur dédié (Figure 82) permettant de rassembler les tâches interactives d'entrée (à partir du modèle HAMSTERS des tâches) avec les entrées du système (à partir du modèle ICO du système) et les sorties du système (à partir du modèle ICO du système) avec les tâches interactives de sortie (à partir du modèle HAMSTERS des tâche).

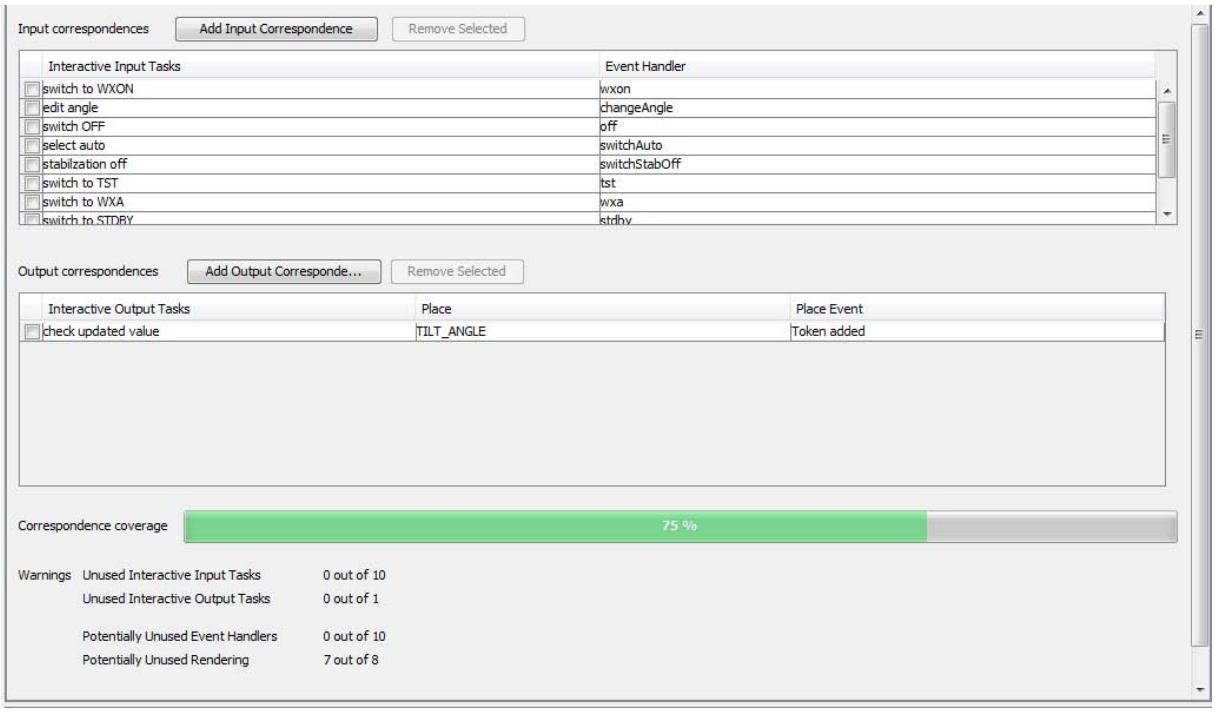


Figure 82. Copie d'écran de la fenêtre d'édition des correspondances entre modèles de types différents

La partie supérieure est composée de deux tableaux représentants l'état actuel de l'édition de la correspondance :

- Le tableau « Input correspondences » représente les correspondances des entrées (e.g. le lien entre une tâche d'entrée et un adaptateur d'activation (un événement de l'utilisateur)). Dans l'exemple, dix tâches sont déjà connectées à dix événements utilisateur (par exemple « switch OFF » est relié à l'événement utilisateur « off »).

- Le tableau « Output correspondences » représente les correspondances des sorties (e.g. le lien entre une tâche de sortie et un adaptateur de rendu). Dans l'exemple, une tâche est déjà connectée à un événement de rendu (un jeton ajouté dans la place « TILT_ANGLE » est connecté à l'événement de rendu « check updated value »).

La partie inférieure représente l'état actuel de l'édition de correspondances. Il est composé d'une barre de progression montrant la couverture actuelle des éléments des tâches et du système par les correspondances éditées, à savoir le taux d'éléments utilisés (l'édition actuelle de la Figure 82 montre 22 éléments utilisés sur 29 soit environ 75% de taux de couverture). En dessous de la barre de progression, un ensemble de mises en garde est affiché, indiquant le nombre de tâches et les éléments du système qui ne sont pas actuellement utilisées (par exemple 7, dans la Figure 82).

Co-exécution ou co-simulation

L'exécution simultanée du prototype très haute-fidélité et des modèles sous-jacents dans l'environnement de mise en œuvre peut être dirigée soit par les modèles de tâches soit par les modèles du système. Lorsque la co-exécution est lancée, une fenêtre de contrôle permet de surveiller et orienter cette co-exécution comme présenté en Figure 83:

- Le modèle ICO du comportement de l'application sur la gauche.

- Le modèle HAMSTERS des tâches utilisateur sur la droite.

- Le contrôleur de co-exécution ou contrôleur de simulation sur la partie inférieure. Elle contient un panneau vide sur le côté droit qui affichera, si nécessaire, des moyens de fournir des valeurs pour l'exécution de la tâche (i.e. les valeurs numériques tapées dans un champ de texte, ou des objets plus complexes sélectionnés à l'aide d'une liste).

- La fenêtre de l'application WXR dans la fenêtre noire en bas à droite, prête à réagir aux évènements de l'utilisateur et aux évènements issus du modèle de comportement.

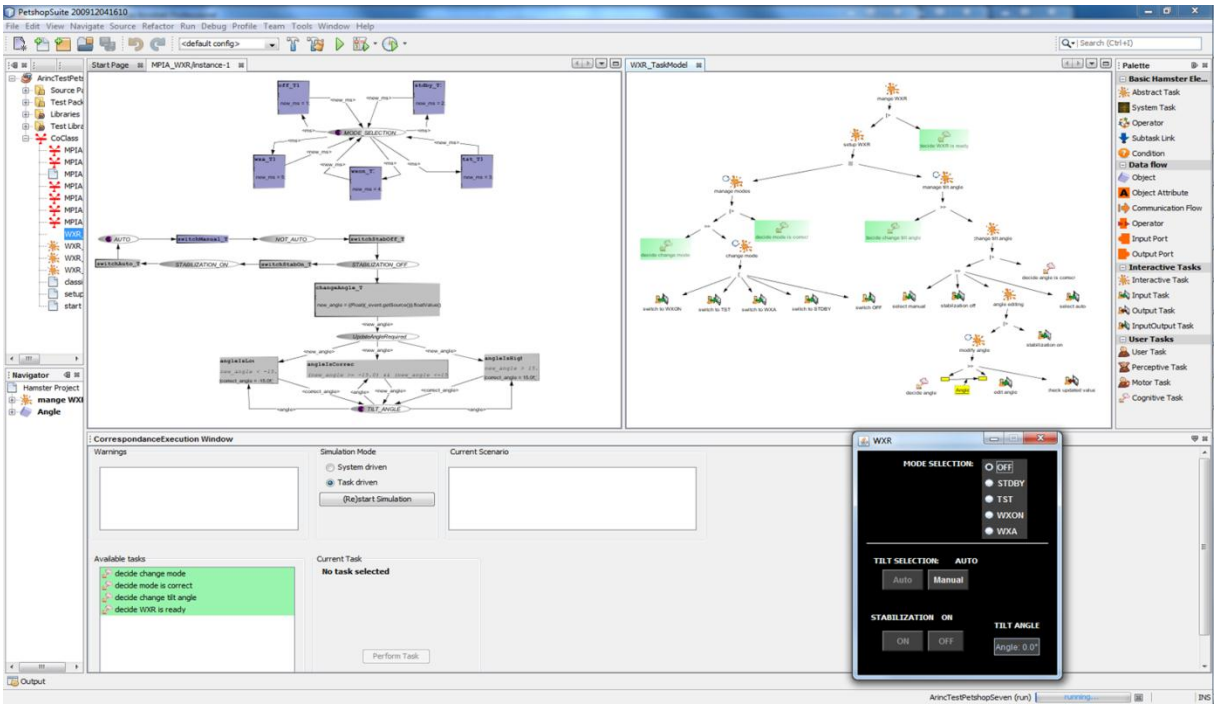


Figure 83. Copie d'écran de la fenêtre de contrôle de la co-exécution du prototype très haute-fidélité et de ses modèles sous-jacents (co-exécution dirigée par le modèle de tâche)

Co-exécution dirigée par les modèles de tâches

L'exécution d'un modèle de tâche produit une séquence de tâches, notamment des tâches interactives (entrée et sortie). Les tâches non interactives ne sont pas liées à l'exécution du système car elles impliquent l'utilisateur sans interaction avec le système ou le système sans feedback à l'utilisateur. Les correspondances identifiées au sein de l'éditeur, permettent de convertir la séquence des tâches interactives en une séquence d'événements utilisateurs se déclenchant dans le modèle ICO, contrôlant l'exécution du système, comme si le scénario était joué par un utilisateur.

Dans cet exemple, lors du démarrage de la fenêtre de contrôle de la co-exécution, la première série de tâches disponibles contient « decide change mode », « decide tilt angle », « decide mode is correct » et « decide WXR is ready » (Figure 84 a)). Réaliser une de ces deux tâches est rendu possible en double-cliquant dessus ou en utilisant le bouton dédié.

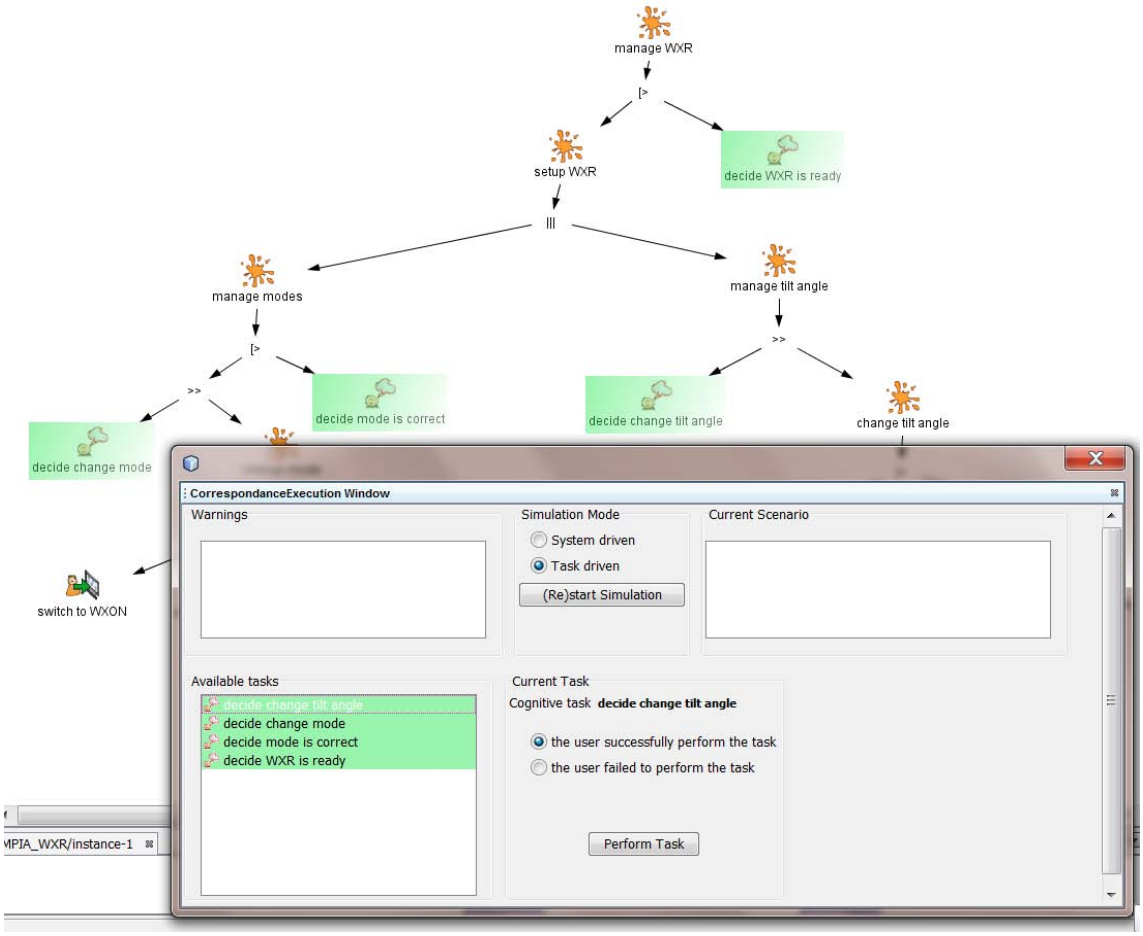


Figure 84. Copie d'écran de la fenêtre de contrôle de la co-exécution dirigée par les modèles de tâches

L'exécution du modèle du système piloté par le modèle de tâche est accomplie tâche après tâche dans le contrôleur de simulation HAMSTERS jusqu'à ce qu'il atteigne la fin du scénario. Si aucun élément du système ne correspond à l'une des tâches disponibles, alors le moniteur de co-exécution affichera un avertissement (comme illustré par la Figure 85) où la tâche « switch OFF » est disponible et l'événement utilisateur correspondant est désactivé. Ce cas pourrait être normal et correspondre à une séquence d'actions forcées par le système à des fins de sécurité par exemple.

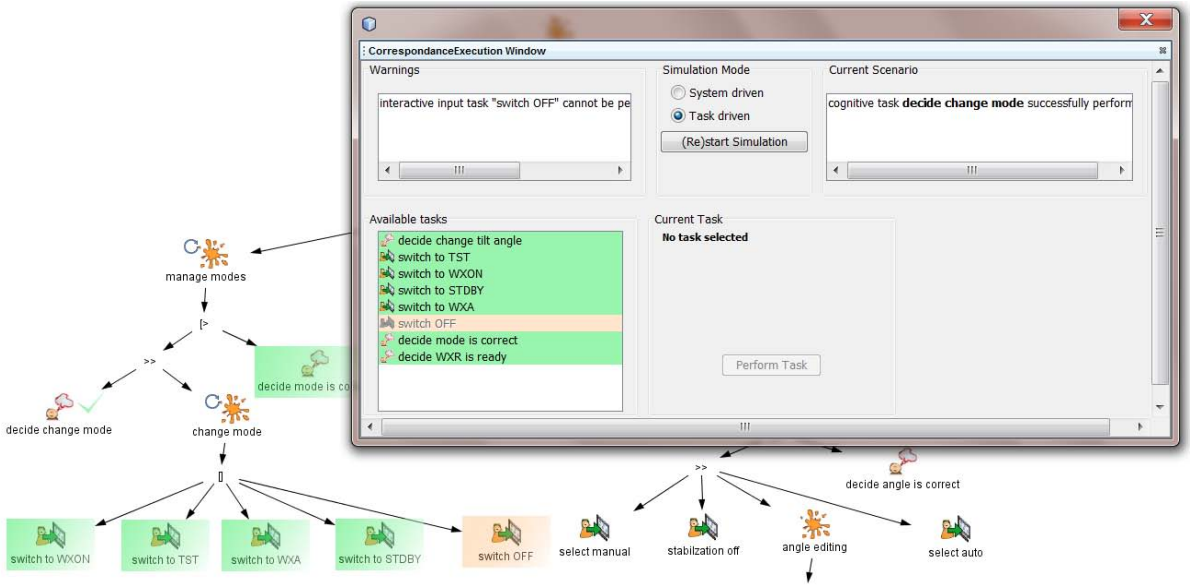


Figure 85. Copie d'écran de la fenêtre de contrôle de la co-exécution dirigée par les modèles de tâches avec un avertissement

La tâche « decide angle » a un port de sortie qui représente la valeur que l'utilisateur veut mettre comme angle d'inclinaison. L'exécution de cette tâche rend active la tâche interactive de sortie « edit angle » qui reçoit la valeur de l'angle d'inclinaison de son port d'entrée. Si l'événement utilisateur correspondant à la tâche « edit angle » est activé (par exemple l'édition de l'angle d'inclinaison en utilisant l'edit box), accomplir la tâche interactive requiert les valeurs courantes. Ces valeurs peuvent être les valeurs du système (les valeurs dans le modèle de système) ou des valeurs saisies manuellement (telles que des numéros). Lors de l'exécution de ces tâches, la fenêtre de contrôle de la co-exécution fournit les moyens d'entrer ou de sélectionner la valeur correspondante. L'identification du type de valeur est effectuée selon la description des artefacts attachés au port de sortie de la tâche interactive correspondante dans le modèle HAMSTERS et de l'adaptateur d'activation correspondant. Un exemple de cette exécution est fourni par la Figure 86.

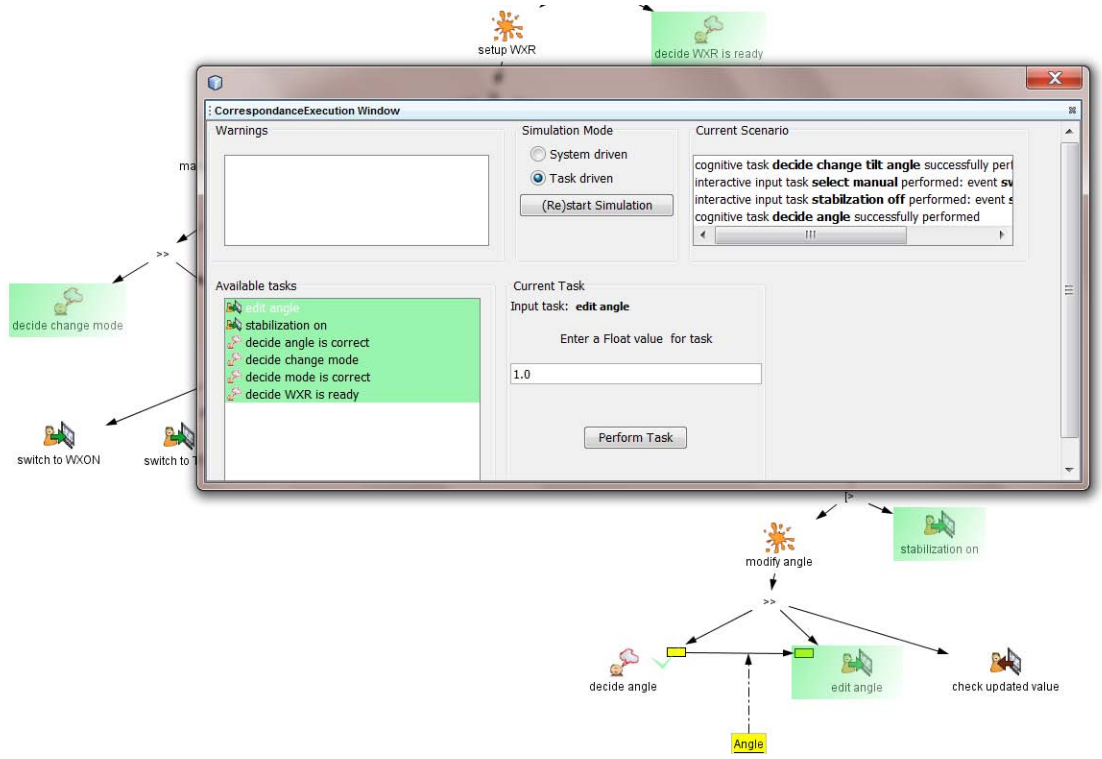


Figure 86 Copie d'écran de la fenêtre de contrôle de la co-exécution avec l'édition des objets

La simulation se termine lorsqu'il n'y a plus de tâche interactive disponible.

Co-exécution dirigée par les modèles du système

Une séquence d'actions sur l'application utilisateur (jouée en utilisant les modèle ICO) peut être visualisée sur un modèle de tâches, grâce à la correspondance entre :

- les gestionnaires d'évènements de l'application et les tâches interactives d'entrée.
- Les évènements déclenchés par l'arrivée d'un jeton dans certaines des places du modèle de comportement et les tâches interactives de sortie.

Les tâches interactives ayant été exécutées sont surlignées en bleu pâle avec un numéro d'exécution inscrit en haut à droite du surlignage correspondant à l'ordre dans lequel elles ont été exécutées. Cette visualisation est rendue possible grâce au fait que chaque action de l'utilisateur peut être liée à une tâche interactive d'entrée.

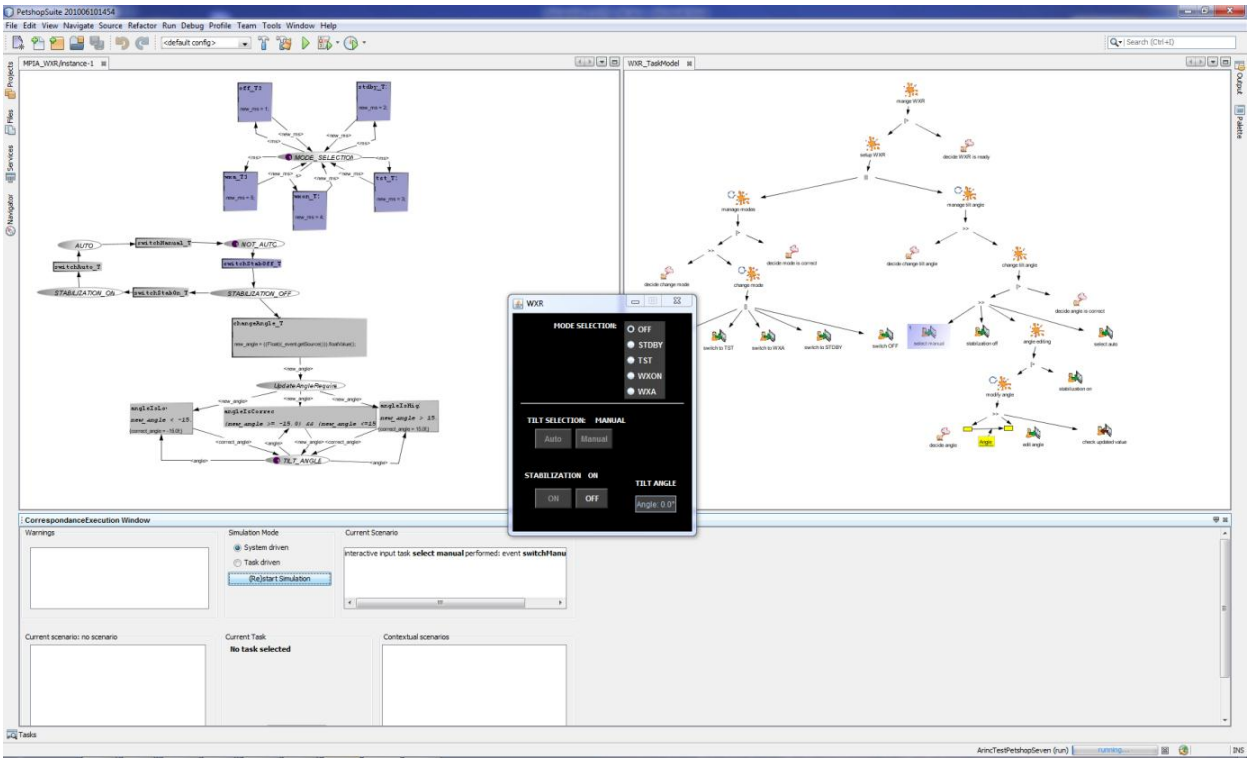
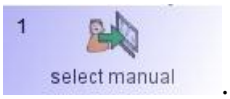


Figure 87. Copie d'écran de la fenêtre de co-exécution du prototype très haute-fidélité et de ses modèles sous-jacents (co-exécution dirigée par le modèle du système)

Il est ainsi possible de suivre l'exécution du système dans le modèle de tâche. Par exemple, la Figure 87 indique que la tâche interactive d'entrée « select manual » a été effectuée en premier



.

La co-exécution du prototype très haute-fidélité et de ses modèles de différents types sous-jacents permet de déterminer si le passage d'une tâche interactive à l'autre (en fonction de l'exécution du système), en utilisant l'analyse du chemin sur le modèle de tâche, et ainsi de mettre en valeurs les incohérences suivantes dès la phase de conception :

- Les séquences d'actions de l'utilisateur autorisées par le modèle du système et interdites par le modèle de tâche en phase de conception des modèles
- Les séquences d'actions de l'utilisateur autorisées par le système et les modèles qui ne satisfont pas les propriétés d'utilisabilité et/ou de fiabilité.

Dans notre exemple, la co-exécution du prototype très haute-fidélité met en lumière que l'utilisateur, pour s'assurer que l'application fonctionne correctement, doit sélectionner le mode « TST » régulièrement. Cette tâche fastidieuse génère une surcharge cognitive inutile pouvant nuire à l'utilisabilité et à la fiabilité des opérations. Elle pourrait être évitée en étant réalisée de manière automatique par le système.

Une nouvelle itération des opérations de modélisation des tâches et du système est donc effectuée, à l'issue de laquelle deux nouvelles versions des modèles sont produits (Figure 88 et Figure 89).

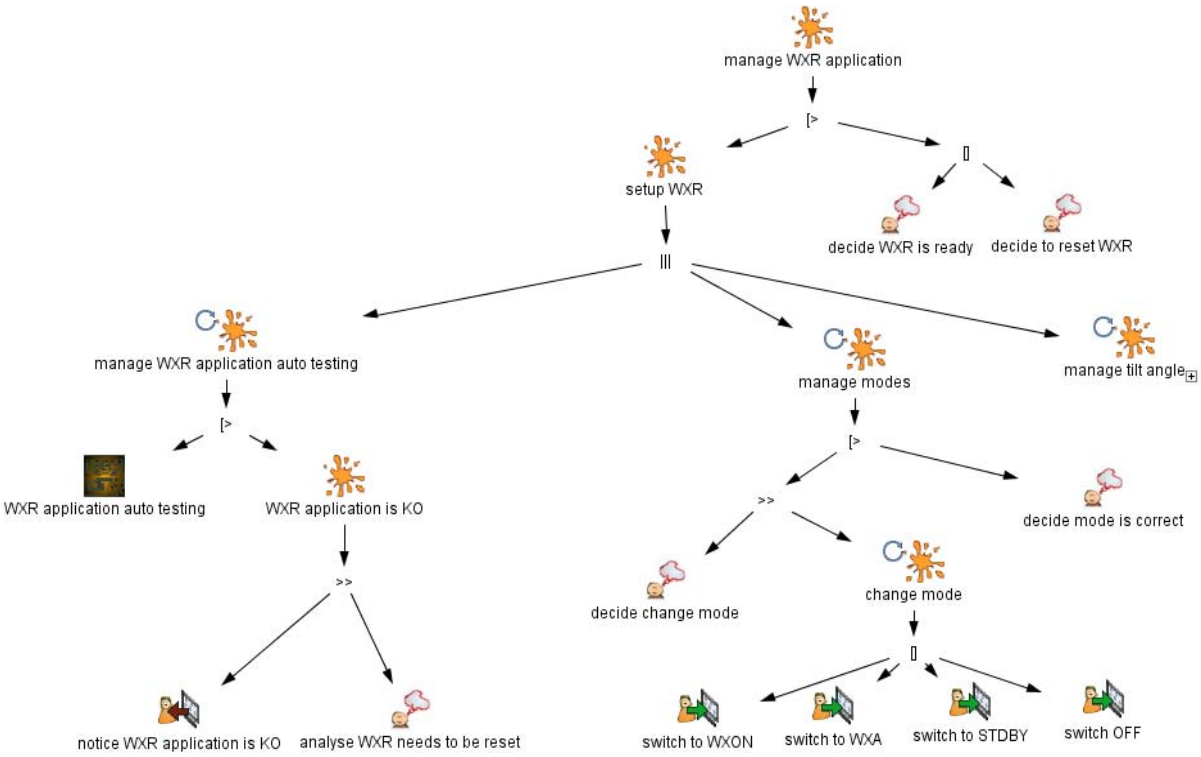


Figure 88. Troisième itération du modèle de tâches (activités de test de l'état de l'application automatisées)

La Figure 88 montre la troisième itération du modèle de tâches avec la fonction de test automatique réalisée par le système. Le sous arbre de test de l'application, dont la tâche abstraite « manage WXR application » est le sommet, a été modifié pour indiquer que le système effectue ce test automatiquement (tâche système « WXR application auto testing »). Cette tâche est effectuée de manière répétitive tant que l'état de l'application est correct. Si un problème est détecté, le système rendra visible à l'utilisateur (tâche interactive de sortie « notice WXR application is KO ») que l'état de l'application est incorrect. L'utilisateur comprendra alors que la remise à zéro du système est nécessaire (tâches cognitives « analyse WXR needs to be reset »).

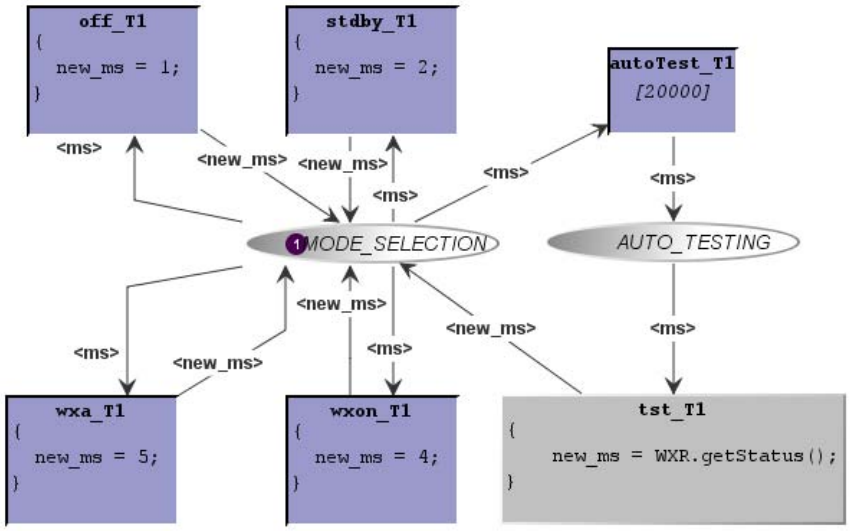


Figure 89. Seconde itération du modèle ICO de la fonctionnalité de changement de mode (avec une fonction de test automatique de l'état de l'application)

La Figure 89 présente la seconde itération du modèle ICO de la fonctionnalité de changement de mode. Par rapport à l'itération précédente de ce modèle, la transition temporisée « autoTest_T1 » a été ajoutée pour déclencher automatiquement la sélection de la fonctionnalité de test toutes les 20000 millisecondes.

Lorsque la correspondance entre les modèles ICO et les modèles de tâches HAMSTERS satisfait les propriétés d'utilisabilité et de fiabilité requises, la phase d'évaluation quantitative des performances de l'utilisateur peut démarrer.

2.2.4 Analyse quantitative des performances utilisateur

Le Chapitre 1 détaille l'ensemble des étapes d'analyse quantitative des performances des utilisateurs avec un prototype très haute-fidélité et les modèles formels du système. La Figure 90 reprend cet ensemble d'étapes en les complétant avec les notations et outils de notre environnement de mise en œuvre (PetShop, HAMSTERS). De plus, PetShop possède une fonctionnalité de récolte et d'enregistrement des évènements permettant de consigner temporellement tous les évènements se produisant sur chaque place et chaque transition de toutes les instances de tous les modèles de comportement couramment exécutés.

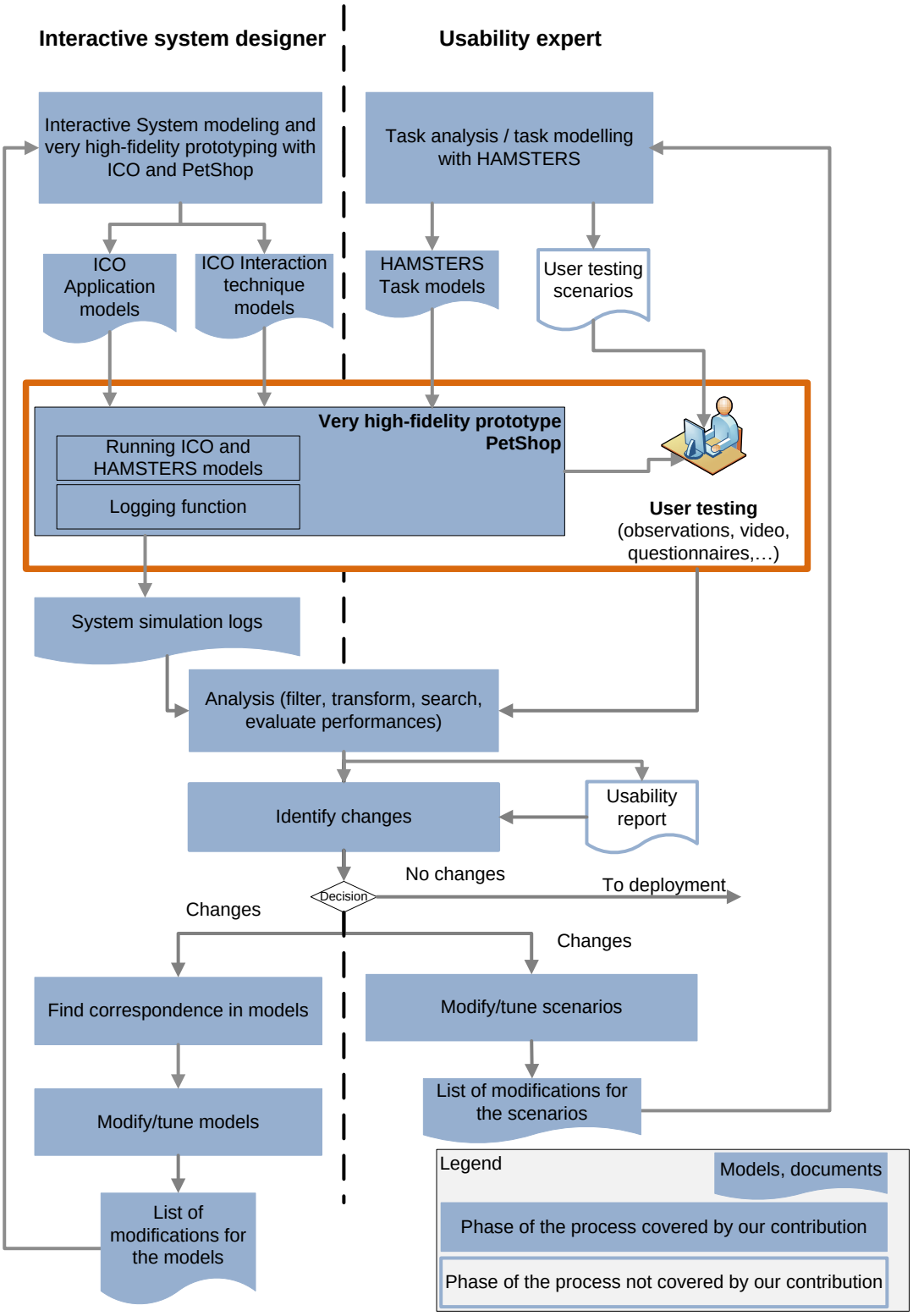


Figure 90. Diagramme de flux de l'évaluation de l'utilisabilité d'un système interactif avec l'environnement de mise en œuvre

Notre environnement de mise en œuvre permet, notamment grâce à la modélisation ICO complète et non ambigüe du comportement du prototype très haute-fidélité et grâce à la fonctionnalité de récolte des évènements:

- Le jeu répété des scenarios sur le prototype très haute-fidélité et les modèles sous-jacents.

- L'observation du comportement des utilisateurs lors de la réalisation de leurs tâches sur le prototype très haute-fidélité.
- La collecte de tous les types d'évènements dans le système (évènements au niveau des pilotes de périphériques, enchaînement d'évènements de plus haut niveau dans les couches logicielles du prototype et ayant un impact sur le comportement de l'application).
- La personnalisation fine de la technique d'interaction (des couches logicielles applicatives aux pilotes de périphériques).
- La mesure quantifiée précise et systématique des performances de l'utilisateur sur le prototype.

Pour illustrer cette sous-phase d'analyse quantitative, nous avons choisi de ne pas utiliser l'exemple illustratif de l'application WXR mais plutôt de prendre un exemple d'application basé sur une technique d'interaction plus complexe de click combiné pour la sélection d'un objet graphique. Ce type de technique d'interaction se retrouve dans les cockpits d'avions commerciaux lorsque le pilote et le copilote, chacun muni d'un périphérique d'entrée appelé KCCU (Keyboard and Cursor Control Unit, un exemple est présenté sur la Figure 91), doivent sélectionner un objet graphique pour que la sélection de cet objet graphique soit prise en compte.

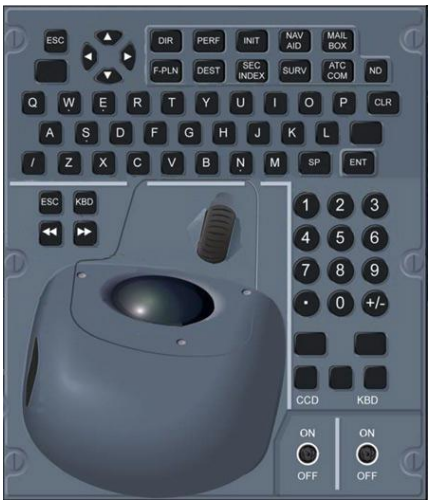


Figure 91. Exemple de Keyboard and Cursor Control Unit (KCCU)

L'application utilisée pour mettre en œuvre la phase d'analyse quantitative des performances utilisateur sur un click mixte est l'application logicielle de « nettoyage » des icônes d'un bureau, dont une copie d'écran de l'interface est présentée sur la (Figure 92).

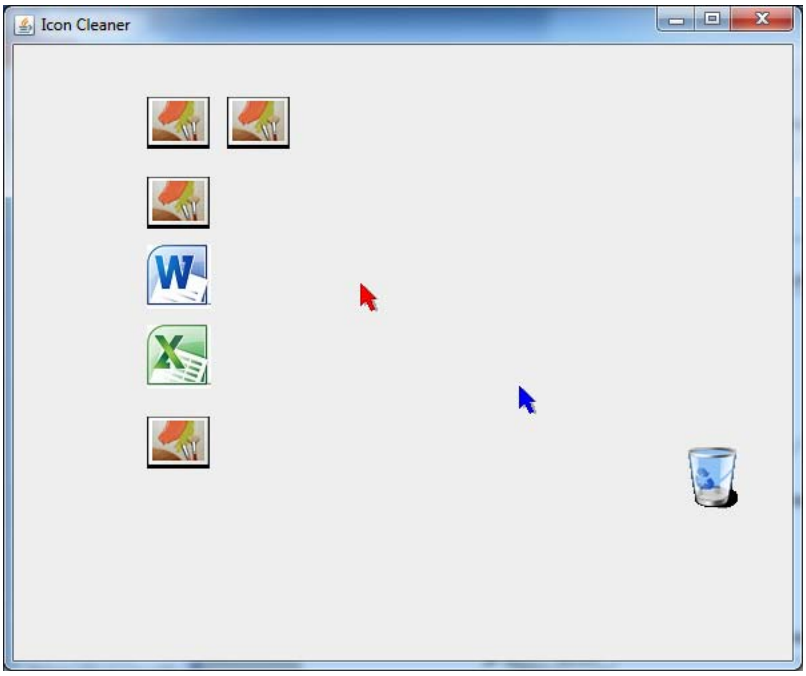


Figure 92. Copie d'écran de l'application de nettoyage des icônes du bureau

Lors de l'exécution de cette application, l'utilisateur a pour but de déplacer 4 icônes jusqu'à la corbeille excepté l'icône Word et l'icône Excel. Pour cela il doit utiliser deux souris (curseurs rouge et bleu sur la Figure 92) et cliquer une fois sur une icône puis une fois sur l'icône « poubelle ». Le fichier sélectionné pour être effacé du bureau le sera effectivement si l'utilisateur a réussi son click combiné en un temps prédéterminé.

Lors de l'analyse quantitative des performances, nous allons chercher à savoir combien d'opérations de click combiné ont échouées lors de l'utilisation de l'application. Dans le cadre de cet exemple, nous voulons montrer comment les actions utilisateurs effectuées lors des sessions d'analyse quantitative des performances et tests d'utilisabilité peuvent être collectée par notre environnement et tracées dans plusieurs parties du système pour faciliter la résolution de problèmes et permettre des corrections fines.

Afin de modéliser l'application et développer le prototype très haute-fidélité, trois modèles de comportement sont nécessaires :

- Un modèle gérant la récupération des coordonnées absolues d'une souris (Figure 93). Deux instances de ce modèle s'exécuteront en même temps que l'application.
- Un modèle de l'application de nettoyage du bureau (Figure 94). Une instance de ce modèle s'exécutera en même temps que l'application.
- Un modèle de la gestion de la technique d'interaction (Figure 95). Une instance de ce modèle s'exécutera en même temps que l'application.

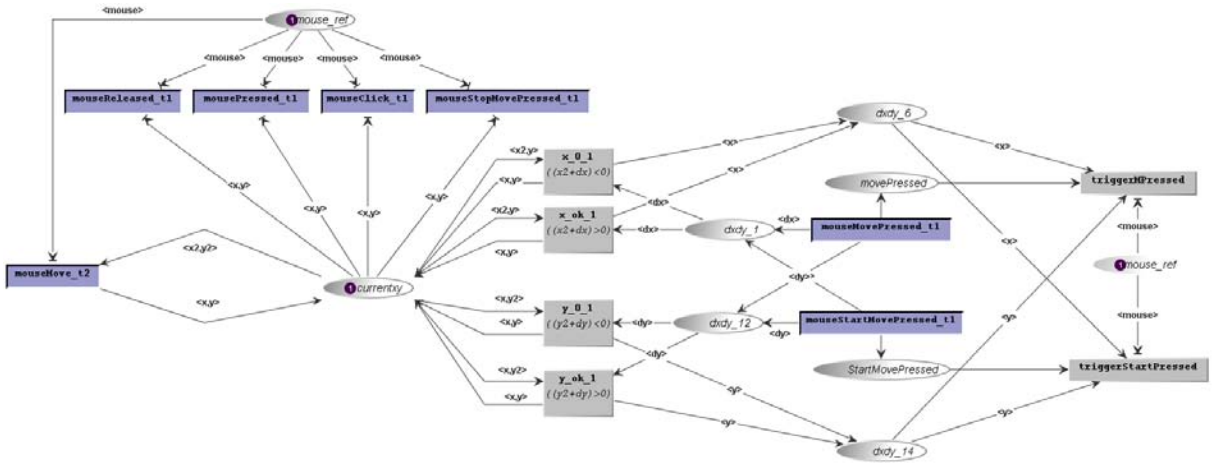


Figure 93. Modèle de comportement utilisé pour récupérer les coordonnées absolues d'une souris

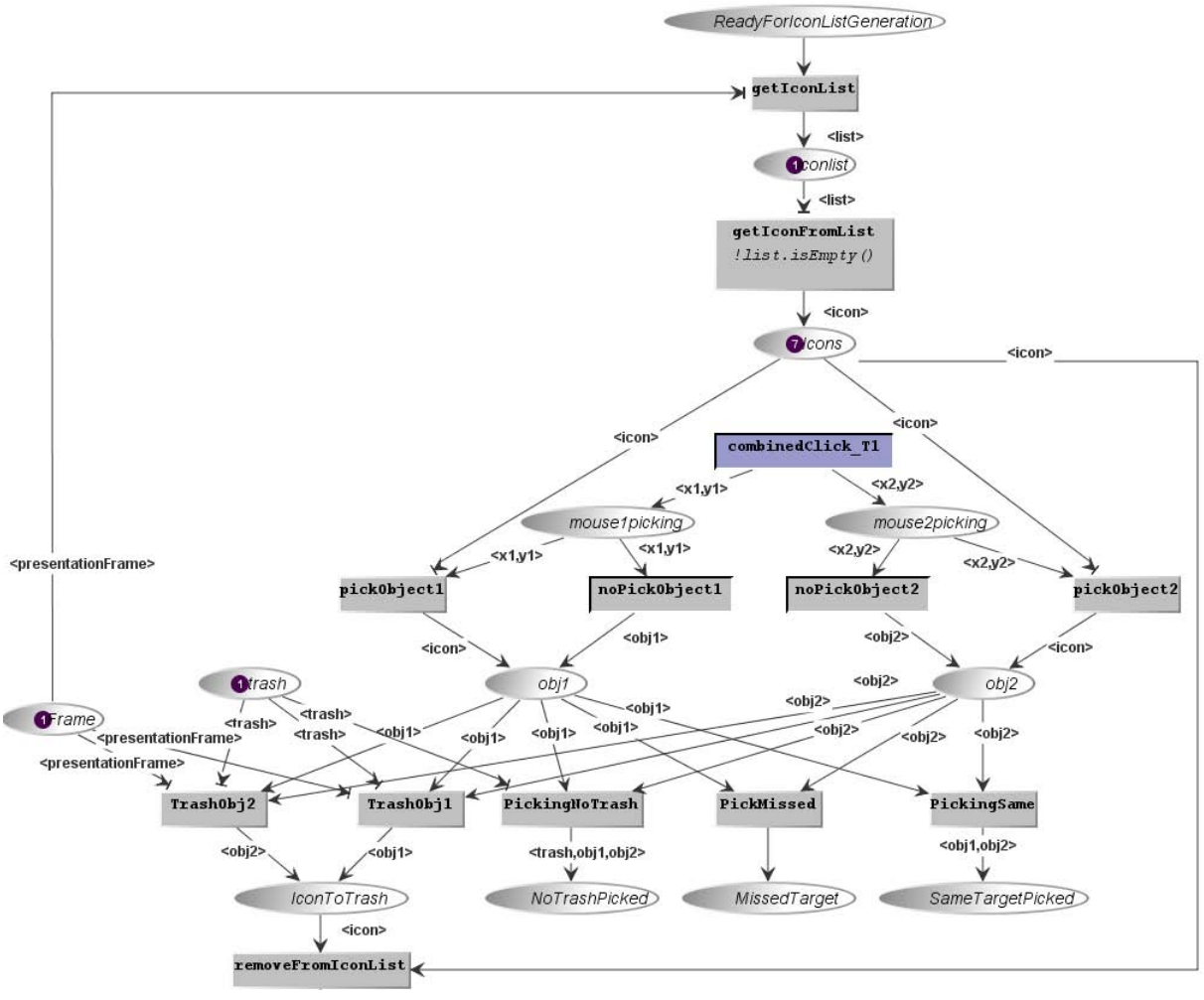


Figure 94. Modèle de comportement de l'application de nettoyage du bureau

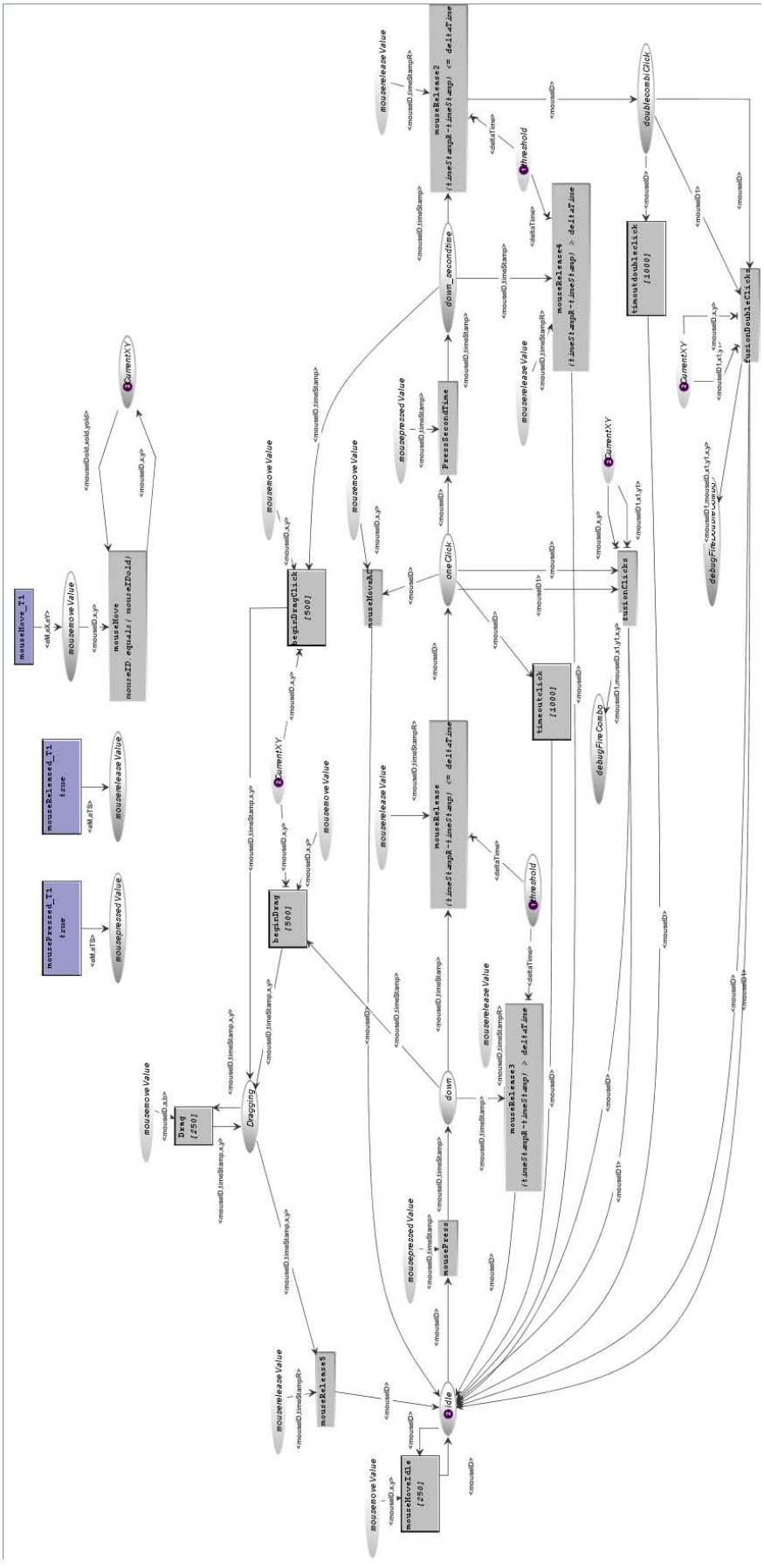


Figure 95. Modèle de comportement de la technique d'interaction avec deux souris

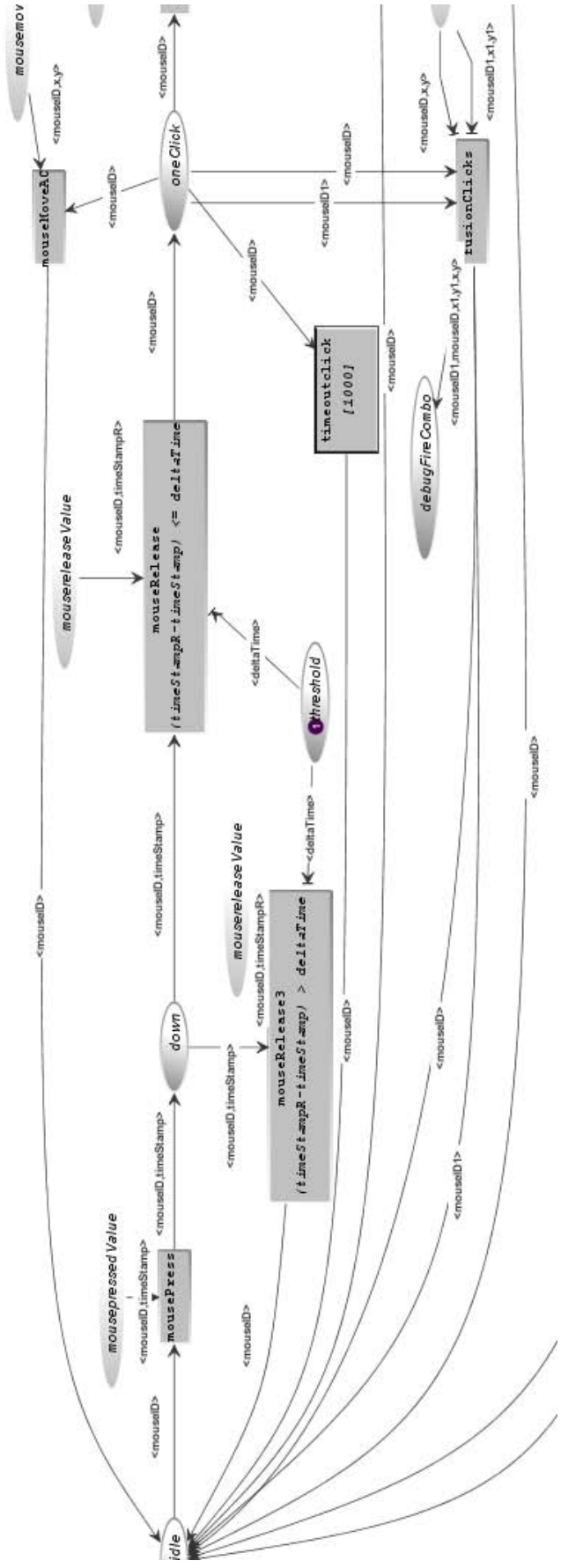


Figure 96. Agrandissement de la zone du modèle de technique d'interaction décrivant le click combiné

La Figure 96 est un agrandissement d'une sous-partie du modèle de la technique d'interaction précédemment présentée sur la Figure 95. Il nous permet d'expliquer le lien entre les modèles de comportement et les traces relevées suite aux tests d'utilisabilité. Dans l'état initial du modèle de la technique d'interaction, les deux jetons se trouvent dans la place « Idle ». Cet agrandissement permet de montrer que si l'utilisateur effectue une pression sur une souris (franchissement de la transition « mousePress ») puis relache la pression (franchissement de la transition « mouseRelease ») dans un temps donné « deltaTime », il aura effectué un click, (et potentiellement une sélection d'objet, cet aspect étant géré par le modèle de la Figure 94). S'il réussit à effectuer ce click avec chaque souris, deux jetons se trouveront dans la place « oneClick », la transition « fusionClick » sera franchie : le click combiné sera réussi. Par contre, si un jeton arrivé dans la place « oneClick » attend plus de 1000 ms (valeur paramétrable) seul dans cette place, la transition temporisée « timeoutclick » sera franchie et le jeton repartira dans la place « Idle ».

Une session de test de l'application est effectuée lors de laquelle un utilisateur se sert de l'application de nettoyage du bureau pour effacer les 4 icônes. Le fichier de traces généré durant cette session de test est disponible au format XML[6]. Il est ensuite importé dans un tableur. La Figure 97 montre un extrait du ce tableur. Il est fragmenté en deux parties horizontales afin de permettre la visualisation du début des traces et de la fin des traces.

	A	B	C	D	E	F	G
1	time delta	type	name	action	time	instance	class
2	0	place	currentxy	token_removed	1,308E+12	instance2	AbsoluteCoord
3	0	place	currentxy	token_added	1,308E+12	instance2	AbsoluteCoord
4	0	transition	mouseMove_t2	firetransition	1,308E+12	instance2	AbsoluteCoord
5	0	transition	y_0_1	substitution_changed	1,308E+12	instance2	AbsoluteCoord
6	0	transition	mouseMove_t2	substitution_changed	1,308E+12	instance2	AbsoluteCoord
7	0	transition	x_0_1	substitution_changed	1,308E+12	instance2	AbsoluteCoord
8	0	transition	mousePressed_t1	substitution_changed	1,308E+12	instance2	AbsoluteCoord
9	0	transition	mouseStopMovePressed_t1	substitution_changed	1,308E+12	instance2	AbsoluteCoord
10	0	transition	mouseReleased_t1	substitution_changed	1,308E+12	instance2	AbsoluteCoord
5071	31044	place	IconToTrash	token_removed	1,308E+12	instance1	Picking
5072	31044	place	Icons	token_removed	1,308E+12	instance1	Picking
5073	31044	place	removedIcon	token_added	1,308E+12	instance1	Picking
5074	31044	transition	removeFromIconList	firetransition	1,308E+12	instance1	Picking
5075	31044	transition	removeFromIconList	disabled	1,308E+12	instance1	Picking
5076	31044	transition	pickObject1	substitution_changed	1,308E+12	instance1	Picking
5077	31044	transition	pickObject2	substitution_changed	1,308E+12	instance1	Picking
5078	31668	place	currentxy	token_removed	1,308E+12	instance2	AbsoluteCoord
5079	31668	place	currentxy	token_added	1,308E+12	instance2	AbsoluteCoord
5080	31668	transition	mouseMove_t2	firetransition	1,308E+12	instance2	AbsoluteCoord

Figure 97. Extraits de l'ensemble des traces des événements dans les modèles de comportement (tableur fractionné en deux parties pour les premières et dernières traces)

Toutes les colonnes du tableur de la Figure 97 sont issues du fichier de traces, excepté la première qui a été rajoutée pour calculer automatiquement le temps écoulé entre un évènement et l'instant de démarrage de la fonctionnalité d'enregistrement des traces. Les colonnes issues du fichier de traces sont les suivantes :

- La colonne « type » indique le type d'élément du modèle auquel fait référence une trace : transition ou place.
- La colonne « name » indique le nom de l'élément auquel fait référence une trace.

[6] http://www.w3.org/XML

- La colonne « action » indique l'action qui s'est produite sur l'élément. Pour une place, il s'agit d'un jeton enlevé (« token_removed ») ou d'un jeton ajouté (« token_added »). Pour une transition, il s'agit d'une activation (« enabled ») ou désactivation (« disabled »), d'un déclenchement (« firetransition »), ou d'une substitution de paramètres (« substitution_changed »).
- La colonne « time » indique à quel instant s'est produit l'évènement depuis l'activation de la fonctionnalité de traçage.
- La colonne « instance » indique sur quelle instance du modèle l'évènement s'est produit.
- La colonne « classe » indique le nom du modèle.

Pour notre exemple, nous allons être particulièrement attentifs aux déclenchements « firetransition » des transitions nommées « fusionClicks » et « timeoutClick ». Le déclenchement de la transition « fusionClicks » indique que le click combiné a réussi et le déclenchement de la transition « timeoutClick » indique qu'il a échoué.

	A	B	C	D	E	F	G
1	time delta	type	name	action	time	instance	class
2263	9922	transition	fusionClicks	firetransition	1,308E+12	instance1	MiceEventManager
2762	12668	transition	fusionClicks	firetransition	1,308E+12	instance1	MiceEventManager
4034	22340	transition	fusionClicks	firetransition	1,308E+12	instance1	MiceEventManager
5026	30998	transition	fusionClicks	firetransition	1,308E+12	instance1	MiceEventManager

Figure 98. Tableur des traces filtré indiquant le nombre de clicks combinés réussis

La Figure 98 montre le résultat d'une opération de filtrage sur les évènements produits sur un élément de type « transition », de nom « fusionClicks » et déclenchés par une action « firetransition » (déclenchement de la transition). Ce filtre correspond à une recherche de clicks combinés réussis. L'utilisateur a réussi quatre clicks combinés.

	A	B	C	D	E	F	G
1	time delta	type	name	action	time	instance	class
1733	6006	transition	timeoutclick	firetransition	1,308E+12	instance1	MiceEventManager
4934	29079	transition	timeoutclick	firetransition	1,308E+12	instance1	MiceEventManager

Figure 99. Tableur des traces filtré indiquant le nombre de clicks combiné ayant échoués

La Figure 99 montre le résultat d'une opération de filtrage sur les évènements produits sur un élément de « type » transition, de nom « timeoutClick » et déclenchés par une action « firetransition » (déclenchement de la transition). Ce filtre correspond à une recherche de clicks combinés ayant échoués à cause du temps pris par l'utilisateur entre le click effectué sur chacune des deux souris. L'utilisateur a échoué deux clicks combinés.

Cette analyse quantitative des performances nous permet ainsi de recommander des solutions pour pallier aux échecs constatés :

- Augmenter le temps autorisé entre les deux clicks (qui correspond à changer la temporisation de la transition « timeoutClick »).
- Entraîner les utilisateurs aux opérations de clicks combinés.

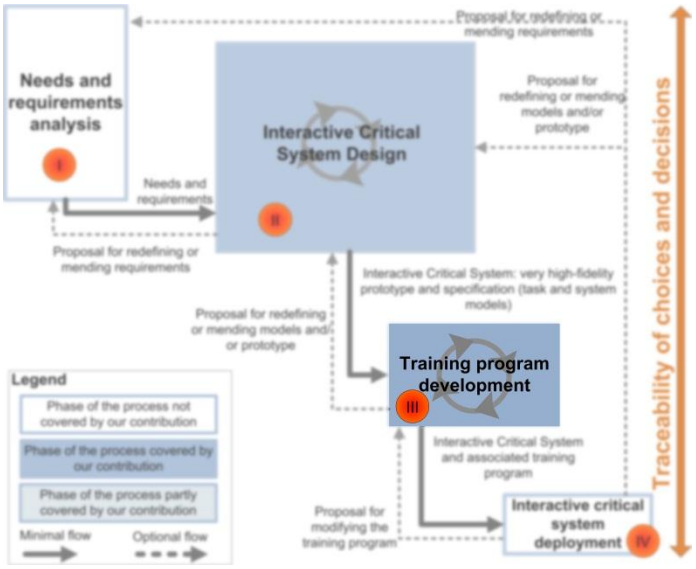
La solution sera choisie en fonction des propriétés d'utilisabilité et de fiabilité requises.

Cette série d'étapes et l'environnement de mise en œuvre permettent aussi d'effectuer une évaluation de l'utilisabilité plus approfondie (selon le standard ISO d'utilisabilité 9241-11 (International Standard Organization, 1996)) et plus complète avec des mesures d'*efficacité* (taux de succès d'accomplissement de la tâche) et des mesures d'*efficience* (temps nécessaire à cet accomplissement).

Grâce à cette fonctionnalité de traçage, les mesures suivantes pourraient par exemple être effectuées de manière précise : nombre de clicks en dehors des icônes, nombre de clicks combinés en dehors de la fenêtre temporelle paramétrée.

Pour conclure cette section, l'évaluation quantitative des performances de l'utilisateur permet de détecter des problèmes de conception et de signifier lors de la phase itérative de modélisation formelle et prototypage haute-fidélité que certains modèles doivent être modifiés. Lorsque les résultats de l'évaluation sont satisfaisants par rapport aux objectifs requis sur les critères d'utilisabilité et de fiabilité, la phase de développement du programme de formation peut être amorcée.

2.3 Phase de développement du programme de formation associé



Cette section détaille la mise en œuvre de la phase de développement du programme de formation, décrite auparavant de manière abstraite dans le Chapitre 1 et développée suite à de premiers travaux (Martinie, Palanque, Navarre, & Winckler, 2010) . Elle est ici instanciée et détaillée avec les notations et outils de notre environnement de mise en œuvre. En plus des propositions d'utilisation des modèles de tâches et du système faites dans le Chapitre 1 pour complémenter les étapes de développement d'un programme de formation, l'environnement de mise en œuvre du processus de développement permet de complémenter ces étapes avec les fonctionnalités d'exploitation synergique du prototype et des modèles sous-jacents et leurs modules logiciels correspondants (« Correspondence manager », « Simulation controller ») et avec le module « Training » de notre environnement intégré Figure 63). Ces modules logiciels peuvent ainsi être utilisés pour mettre en place des sessions de formation assistées par ordinateur.

La Figure 100 détaille quatre étapes, parmi celles requises pour développer un programme de formation (blocs rectangulaires sur la partie gauche), pour lesquelles notre environnement de mise en œuvre est utilisé. La partie droite de la Figure 100 montre les sous-phases correspondantes dans le processus ADDIE de développement d'un programme de formation. Chaque sous-phase est complémentée par les activités de : modélisation des tâches T, modélisation du système S, co-exécution du prototype très haute-fidélité et des modèles sous-jacents C.

La mise en œuvre de ces quatre étapes est exposée ci-après :

- L'étape d'analyse et de conception du programme de formation (premier bloc « Training program analysis and design » en haut sur la partie gauche de la Figure 100), les fonctionnalités de correspondance et de co-exécution permettent de déterminer, de manière plus fine en fonction des contraintes liées au système, les performances requises pour opérer le système. Ces fonctionnalités permettent aussi d'aider à mettre en place la structure et la séquence des sessions de formation grâce aux scénarios d'utilisation générés de manière synergique.

- L'étape de préparation des sessions de formation (second bloc « Training sessions development ») et de leur contenu est détaillée dans la section 2.3.1.
- L'étape de déroulement des sessions de formation (troisième bloc « Training sessions execution ») est détaillée dans la section 2.3.2.
- L'étape d'analyse des performances des opérateurs suite au déroulement des sessions (quatrième bloc « Training sessions evaluation ») est détaillée dans la section 2.3.3.

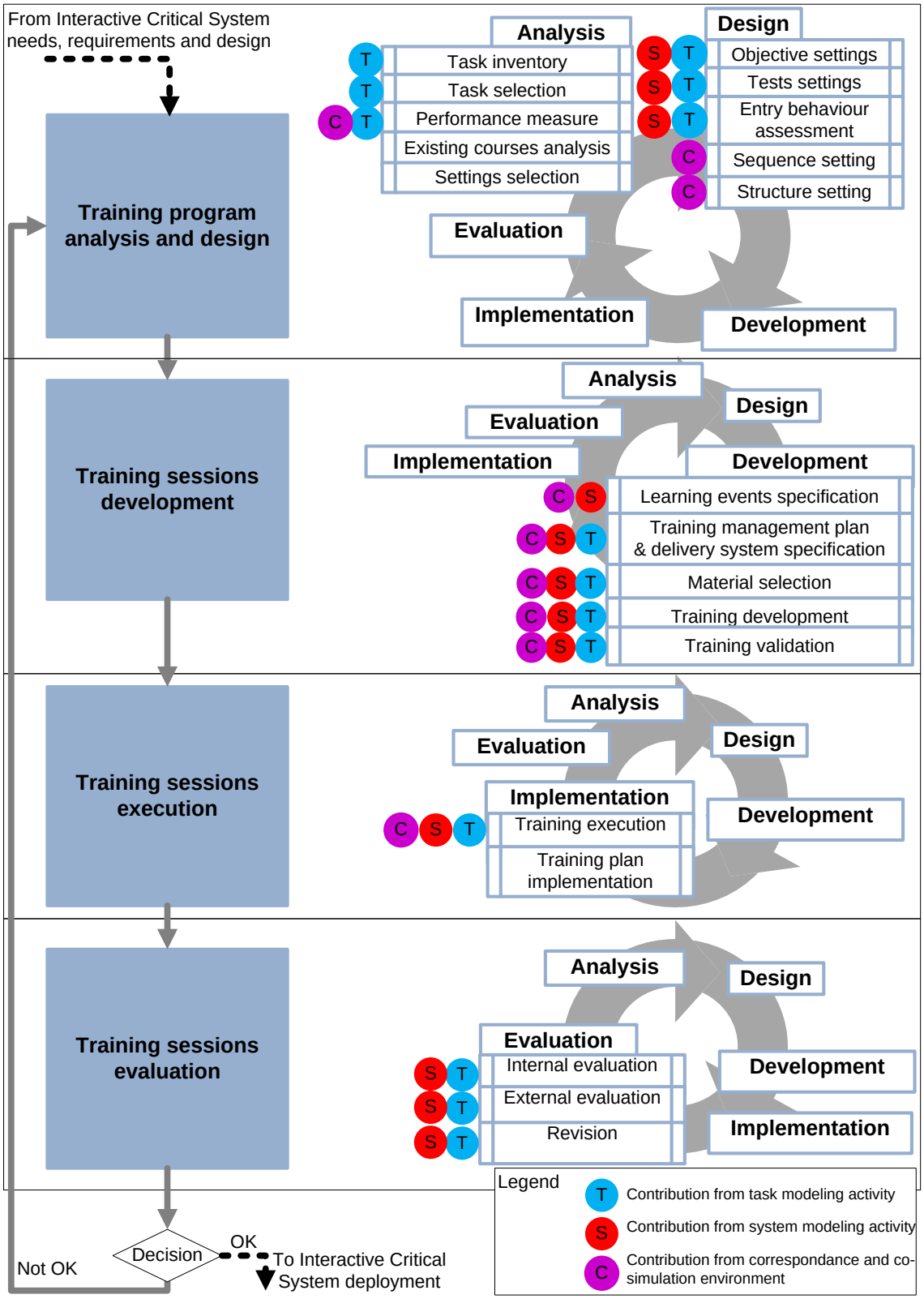


Figure 100. Mise en œuvre du processus de développement d'un programme de formation avec notre environnement intégré

2.3.1 Préparation des sessions de formation

Durant la phase de développement des sessions de formation, le module logiciel « Training module » de l'environnement de mise en œuvre permet d'éditer et d'analyser les plans de session en utilisant les informations des modèles de tâches et du comportement du système. La Figure 101 décrit les différentes parties de l'environnement de mise en œuvre utilisées lors de cette phase ainsi que les artefacts utilisés et générés grâce à l'environnement intégré de mise en œuvre et la manière dont le formateur utilise cet environnement.

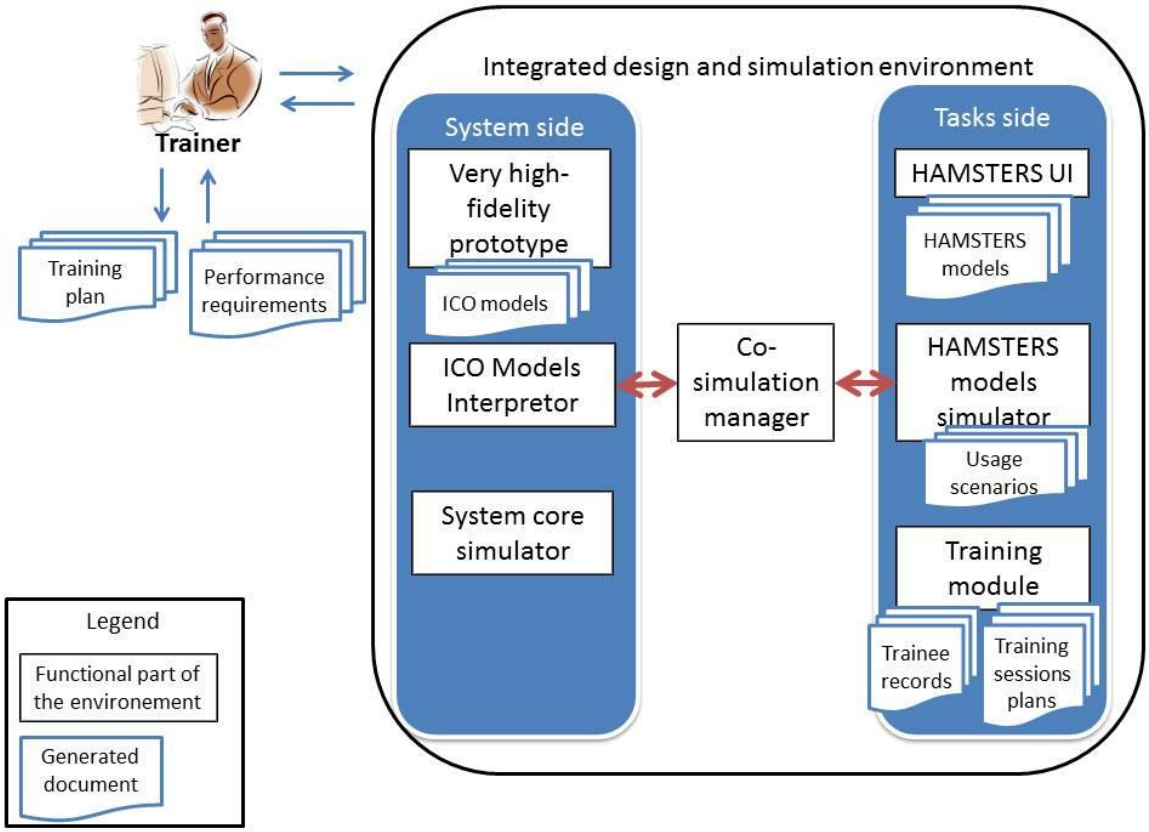


Figure 101. Edition et analyse d'une session de formation utilisant l'environnement intégré de conception et simulation des modèles et du prototype très haute-fidélité

On retrouve dans ce schéma (Figure 101) les éléments de la co-exécution (pont logiciel entre la partie système et la partie tâche), les éléments liés au programme de formation (plan de formation et exigences en termes de performances pour la qualification du personnel) ainsi que les éléments liés aux sessions de formations (scénarios d'utilisation, enregistrement des informations de sessions effectuées par le personnel formé, plans des sessions de formation).

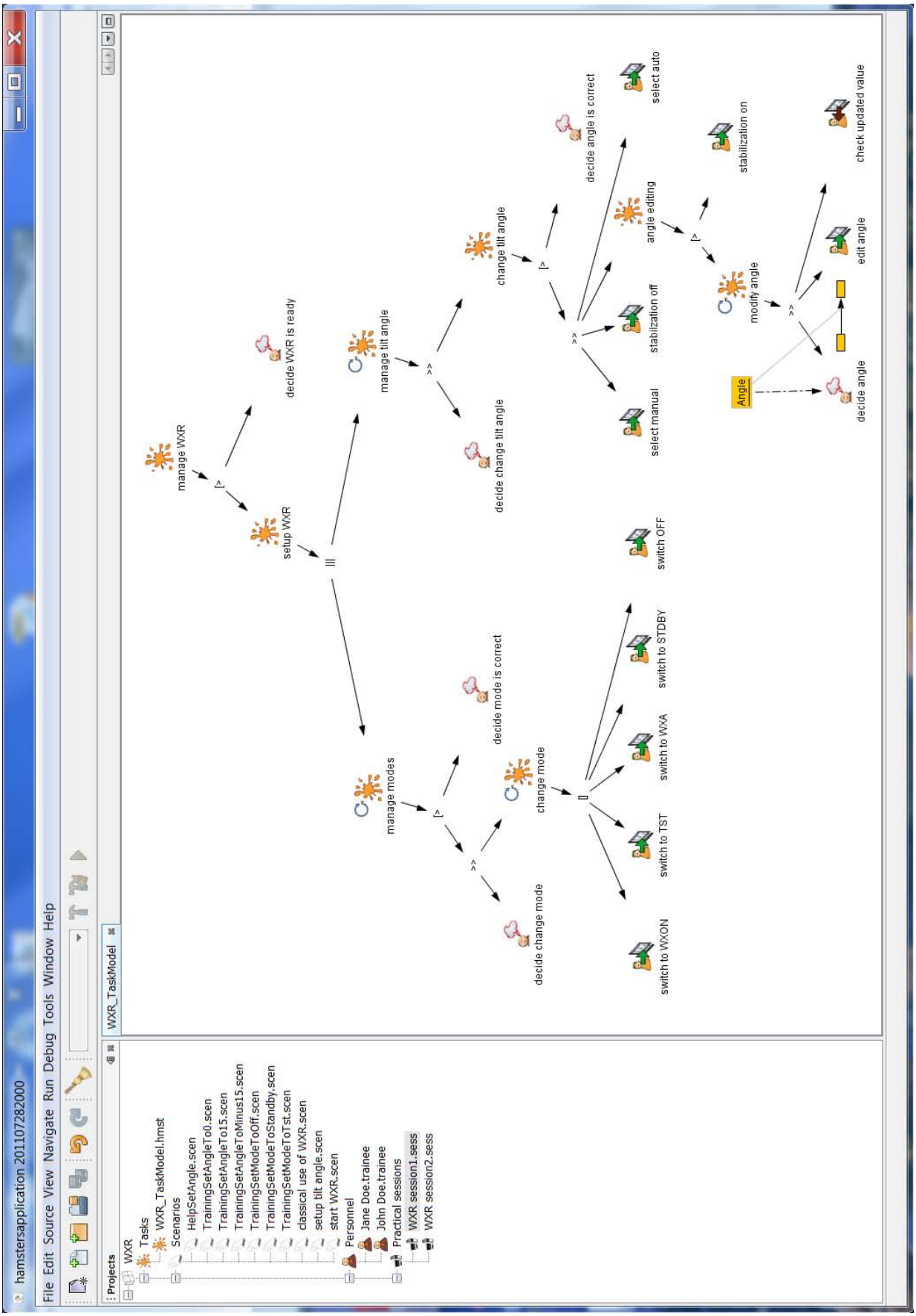


Figure 102. Copie d'écran du navigateur de projets de sessions de formation

La Figure 102 présente le navigateur de projets ou programmes des sessions de formation. Chaque programme est indiqué par l'icône et se décompose en ensembles de :

- Modèles de tâches représentés par le symbole.
- Scénarios d'utilisation.
- Fichiers d'informations sur les sessions effectuées par le personnel.
- Sessions de formation.

Les modèles du système n'apparaissent pas dans cette copie d'écran pour des raisons de lisibilité mais ils peuvent aussi être sélectionnés pour être affichés de la même manière que les fonctionnalités de mise en correspondance et de co-exécution (ces perspectives graphiques de visualisation de modèles du système et de co-exécution sont présentées dans le chapitre 7 pour l'étude de cas).

La Figure 103 montre une copie d'écran de l'éditeur des sessions de formation. Cette partie de l'environnement est utilisée lors de la phase de développement des sessions de formation.

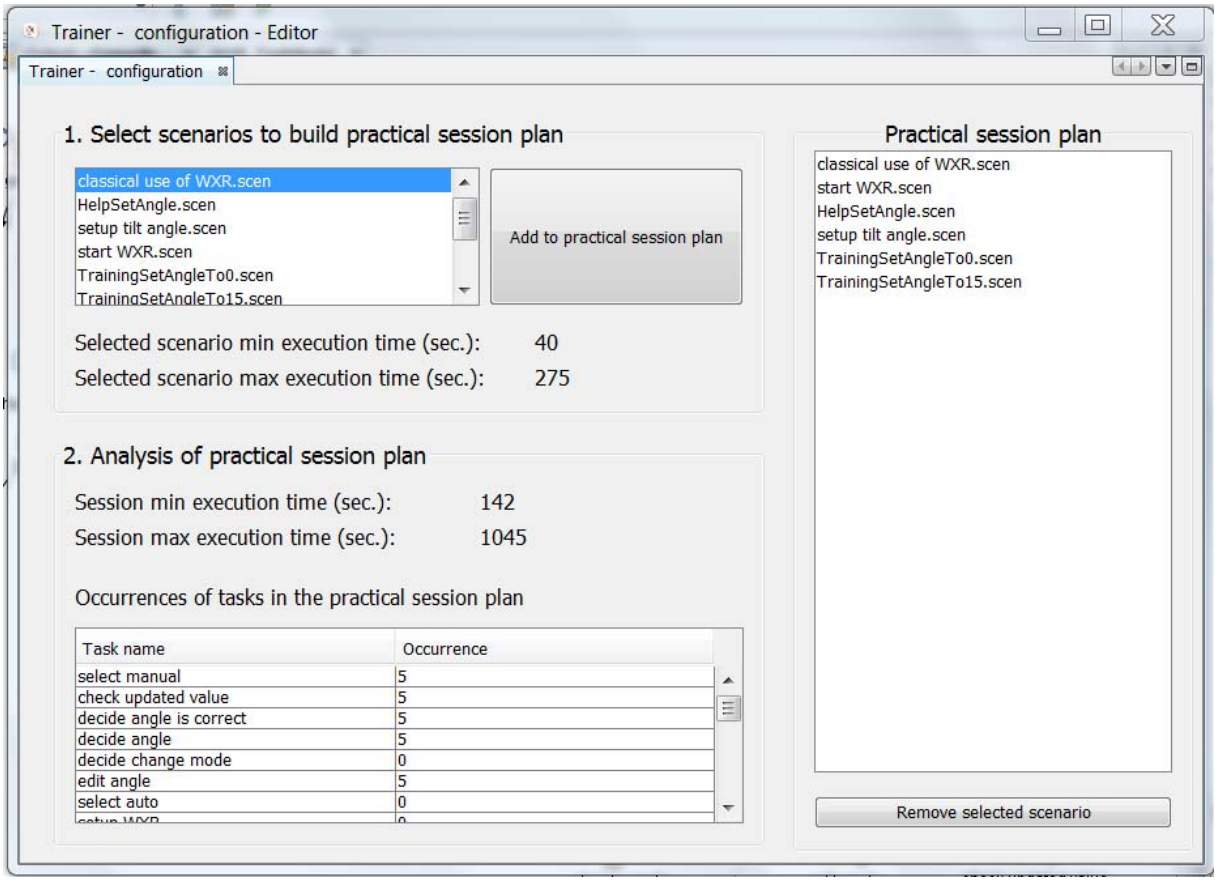


Figure 103. Copie d'écran de l'interface d'édition d'une session de formation par un formateur

Cette interface d'édition permet au formateur de :

- Sélectionner les scénarios d'utilisation sur lesquels le personnel formé devra s'entraîner (partie « 1. Select scenarios to build practical session plan » de la Figure 103). De plus, lorsque le formateur sélectionne un scenario, les temps minimum et maximum requis pour l'exécuter sont automatiquement calculés en fonctions des informations provenant des modèles de tâches correspondants.
- Visualiser le contenu du plan de session actuel en termes de scénarios à effectuer (partie « Practical session plan » de la Figure 103).

- Analyser le plan de session (partie « 2. Analysis of practical session plan » de la Figure 103). Cette partie de l'éditeur permet au formateur de connaître rapidement quelles seront les durée minimales et maximales pour exécuter le plan ainsi que la liste des tâches contenues dans le plan de session actuel et leur nombre d'occurrence. Il peut ainsi vérifier que les tâches sélectionnées lors de la phase d'analyse du programme de formation sont bien présentes.

2.3.2 Déroulement des sessions de formation

Durant la phase d'exécution des sessions de formation, l'environnement intégré de mise en œuvre permet au personnel formé de s'entraîner en suivant une séquence de scénarios d'utilisation liés aux compétences que le personnel doit acquérir. Cette séquence est prédéfinie par le formateur avec le prototype très haute-fidélité. La Figure 104 décrit les différentes parties de l'environnement de mise en œuvre utilisées lors de cette phase ainsi que les artefacts utilisés et générés. Deux types de travaux pratiques sont possibles dans l'environnement intégré :

- Le formateur peut contrôler les évènements internes liés au système et à la mission opérationnelle (que nous appellerons scénarios opérationnels) du personnel afin que le personnel formé réalise les opérations liées aux scénarios d'utilisation du prototype.
- Les scénarios opérationnels peuvent être préenregistrés et sélectionné par le personnel formé pour s'entraîner sur les scénarios d'utilisation du prototype.

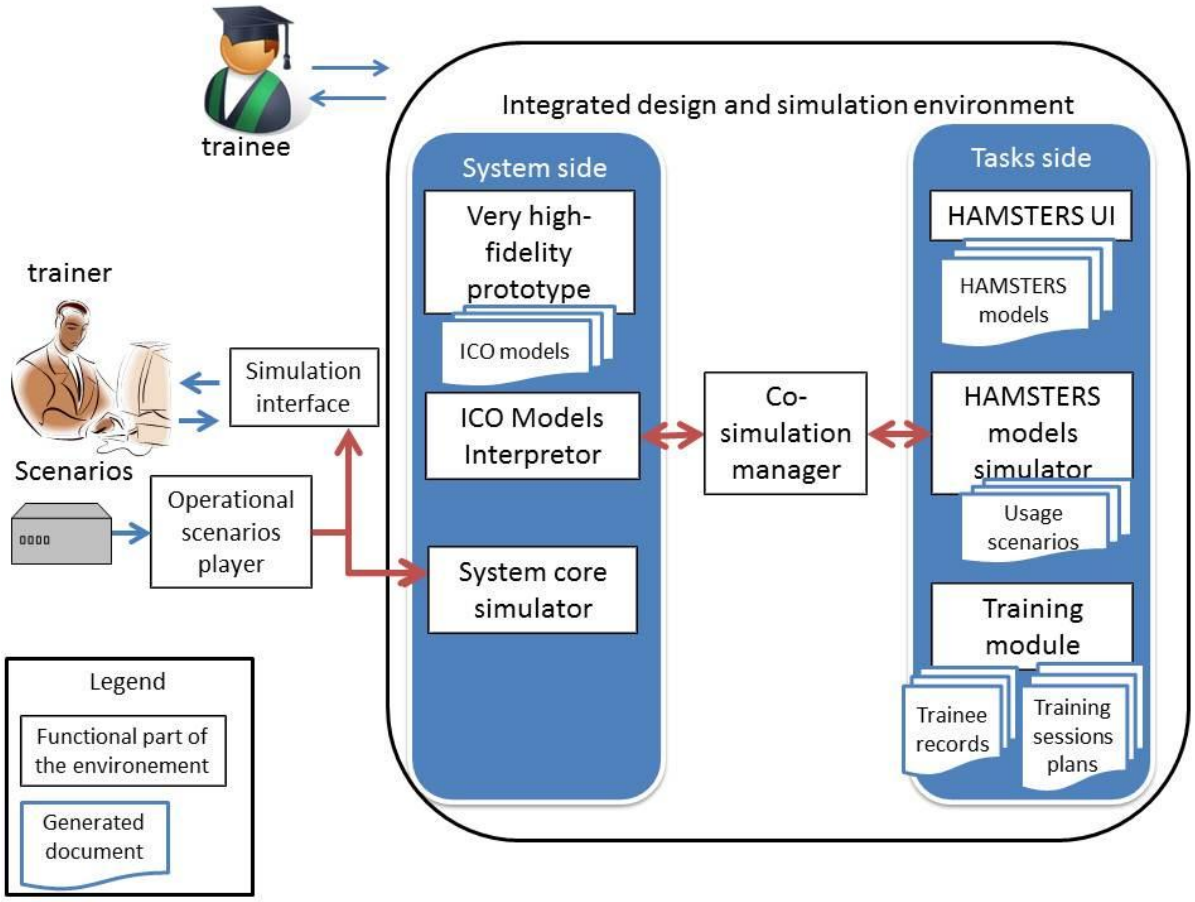


Figure 104. Exécution d'une session de formation utilisant l'environnement intégré de conception et simulation des modèles et du prototype très haute-fidélité

Durant l'exécution de la session de formation, le personnel formé expérimente l'utilisation du prototype. Au sein du même environnement intégré, le personnel formé contrôle le déroulement des scénarios d'utilisation de la session de formation grâce à l'interface de la Figure 105. Cette interface

est basée sur les fonctionnalités de co-exécution de l'environnement synergique intégré et permet de contrôler la co-exécution par un scénario préenregistré. La partie en haut à droite intitulée « Planned exercices » indique au personnel formé la liste des scénarios d'utilisation qu'il doit accomplir. Sous cette partie, plus en bas à gauche, la partie intitulée « Exercise execution » lui permet de contrôler effectivement le déroulement d'un scénario. Sur la partie droite, le personnel formé peut suivre sur le modèle de tâche correspondant, les tâches qu'il doit accomplir.

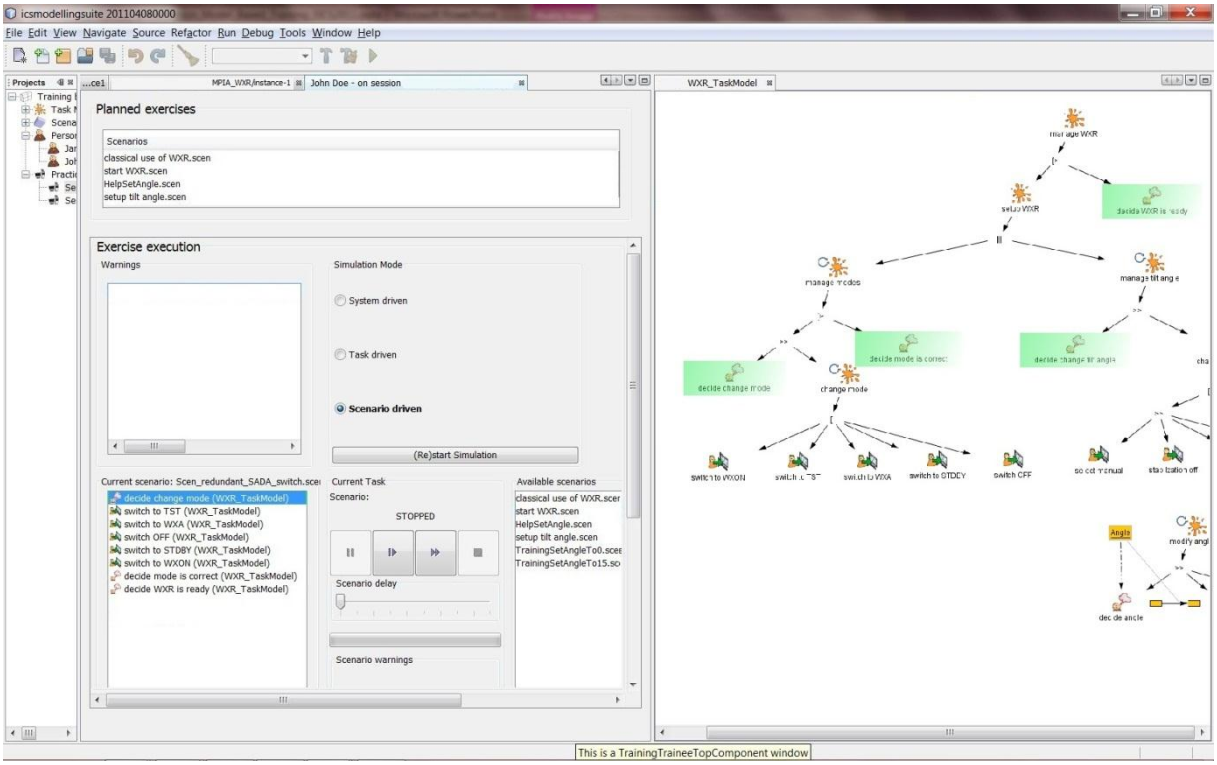


Figure 105. Copie d'écran de l'interface d'exécution d'un plan d'une session de formation

2.3.3 Analyse des performances suite au déroulement des sessions de formation

Durant la phase d'évaluation des sessions effectuées par le personnel, le formateur peut analyser les statistiques liées aux opérations effectuées durant les sessions pour chaque personne formée et ainsi vérifier que les séquences de scénarios ont bien été effectuées et que les objectifs de performance sont atteints.

Lorsqu'une ou plusieurs sessions de formation sont terminées, le formateur peut analyser les données enregistrées par l'environnement synergique intégré lors de l'exécution des sessions. La Figure 106 montre une copie d'écran de l'interface d'évaluation des sessions de formation effectuées par le membre du personnel « John Doe ». La première partie en haut à gauche de l'interface intitulée « Trainee information » indique des informations générales à propos du personnel formé (nom, date de la première et date de la dernière session) mais aussi la liste des sessions effectuées. Lorsque le formateur sélectionne une de ces sessions, la partie droite intitulée « Content of practical session » est rempli avec la liste des scénarios de la session élaborée par le formateur. Chaque scénario est surligné en vert ou en rouge selon que le membre du personnel formé l'a effectué correctement ou non. En bas à gauche, la partie intitulée « Statistics accross practical sessions » donne des informations sur la couverture du plan de formation par le membre du personnel formé en termes de scénarios d'utilisation sur l'ensemble des sessions qu'il/elle a effectué (« Training plan coverage ») et du temps qu'il/elle a mis pour les effectuer (« Execution times on scenarios »).

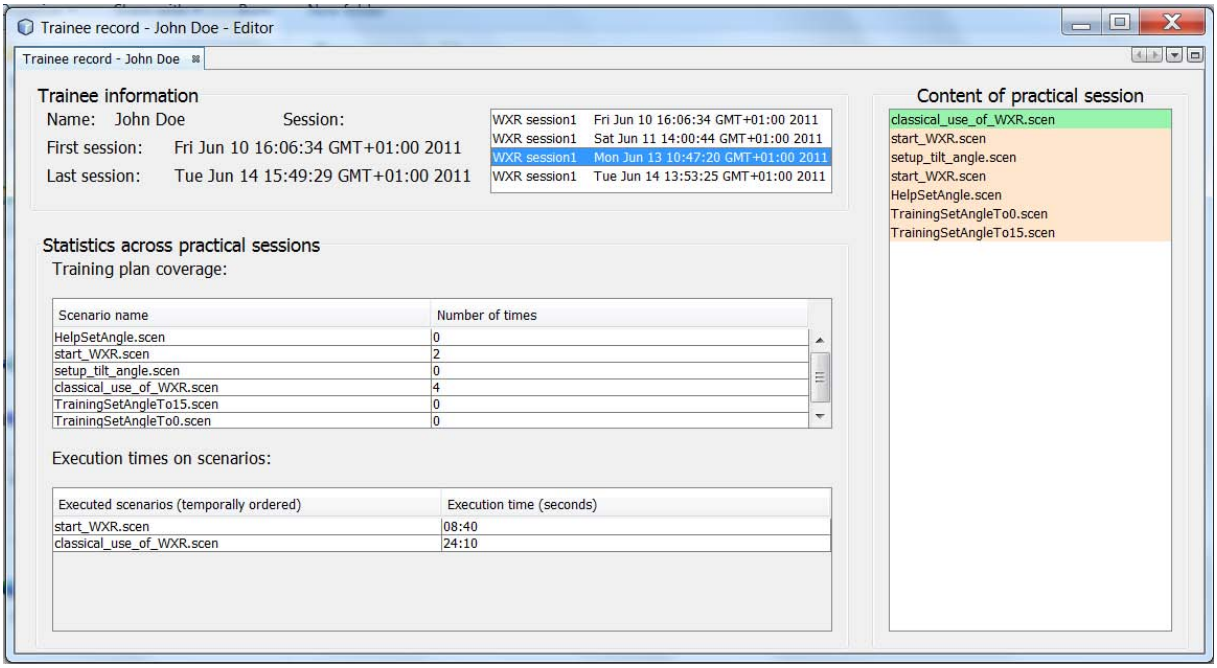
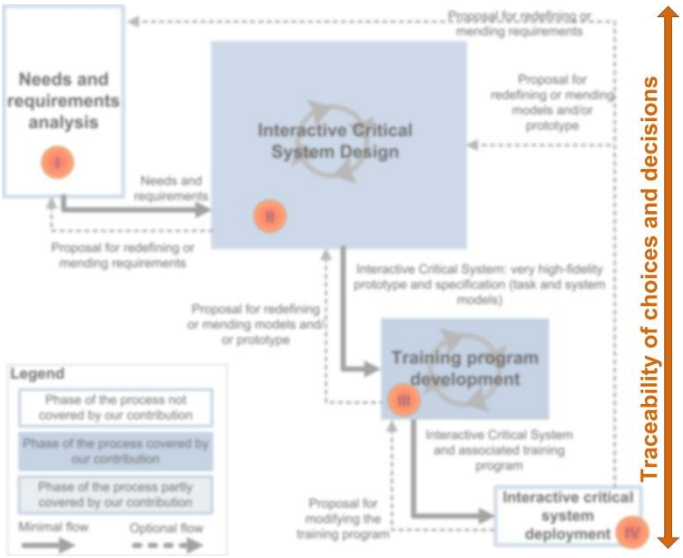


Figure 106. Copie d'écran de l'interface d'évaluation des résultats de l'exécution des sessions de formation pour un membre du personnel formé

Cet environnement de mise en œuvre permet donc de quantifier les performances des futurs opérateurs d'un système interactif critique à l'issue du programme de formation et ainsi d'aider au processus de qualification du personnel avant la mise en opération du système. Les éléments de préparation des sessions et d'analyse statistique sont non exhaustifs et permettent d'illustrer notre approche. D'autres informations utiles pourraient être ajoutées à ces interfaces pour affiner la préparation et l'analyse, comme par exemple la visualisation des temps moyens d'exécution pour chaque scénarios ainsi que des courbes pour mieux visualiser les temps minimum et maximum effectués par les opérateur sur un scénario donné.

2.4 Traçabilité des besoins et exigences tout au long du processus de développement



Cette section détaille la mise en œuvre de la phase transversale de traçage des besoins et exigences tout au long du processus de développement d'un système interactif critique décrite de manière abstraite dans le Chapitre 1. Comme indiqué dans la première section de ce chapitre, nous utilisons l'environnement de conception rationnalisé DREAM basé sur la notation TEAM. L'exemple utilisé pour montrer la traçabilité des exigences tout au long de la conception du composant RadioBox2 est issu d'une étude de cas consultable dans (Martinie, Palanque, Winckler, & Conversy, 2010).

Pour illustrer cette phase transversale du processus, nous prenons comme exemple la traçabilité des exigences de conceptions définies pour un composant graphique par rapport aux choix effectués lors de la conception de ce composant. Le composant graphique RadioBox2 est utilisé par les systèmes d'affichage de cockpits d'avion et défini par le standard (Airlines Electronic Engineering Committe, 2002). Ce composant graphique permet de sélectionner de manière exclusive une option parmi un ensemble d'options. Il est utilisé pour différents types d'applications dans les cockpits d'avion comme celle de gestion des paramètres du radar météo. La Figure 107 montre une apparence visuelle possible pour un composants graphiques RadioBox2.

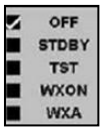


Figure 107. Exemples d'aspect visuel d'un composant graphique RadioBox2

La Figure 108 montre une sous-partie du diagramme des choix effectués lors de la conception de ce composant graphique. Une des questions émises lors de la conception (rectangle rose aux bords arrondis) est : « Une option devrait-elle être sélectionnée par défaut ? ». Deux options sont possibles (disques orangés) :

- Une et une seule option sera toujours sélectionnée par défaut.
- Au plus une option pourra être sélectionnée et le composant graphique pourra afficher qu'aucune option n'est sélectionnée.

A chaque option est associé le symbole d'un trombone, qui signifie que des modèles du système sont liés à cette option (l'éditeur permet d'associer une option avec un ou plusieurs fichiers, dont les modèles du système concernés). Ces options sont évaluées par rapport à trois critères triangles vert : l'effort d'apprentissage, la réussite des opérations de manipulation, la fréquence des échecs lors de l'utilisation. Ces trois critères sont chacun liés à 3 facteurs (triangles rectangles bleus) : facilité d'apprentissage, opérabilité et fiabilité. Dans le cas de cet exemple, les critères sont issus du standard ISO sur la qualité logicielle (International Standard Organisation, ISO IEC 9126: Software

engineering -- Product quality, 2001) mais peuvent être choisis parmi d'autres standards. Les liens entre les options et les critères ont été pondérés (trait plus ou moins gras) en fonction des évaluations quantitatives effectuées à partir de l'utilisation du prototype très haute-fidélité et des modèles sous-jacents. Cette figure montre aussi que la seconde option a été retenue (disque orangé plus foncé et entouré d'un cercle noir).

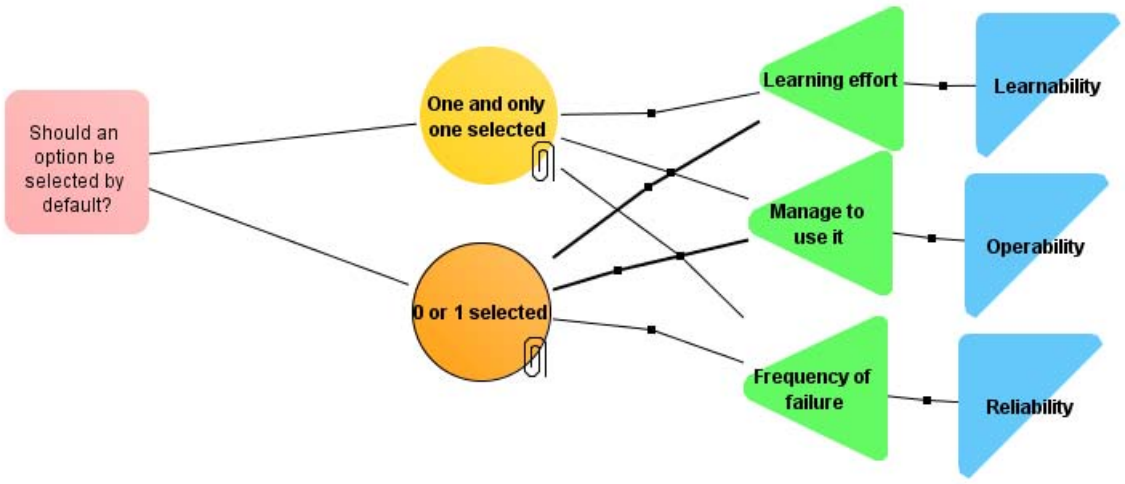


Figure 108. Exemple de diagramme de conception rationnalisée du composant graphique RadioBox2 avec la notation TEAM

Grâce aux modifications apportées à la notation TEAM et à l'environnement DREAM (décrites dans la section 1.2), nous pouvons désormais tracer les besoins et exigences tout au long du cycle de conception du système interactif. La Figure 109 montre un exemple de traçabilité d'une exigence (« REQ008 ») sur les choix de conception du composant graphique RadioBox2.

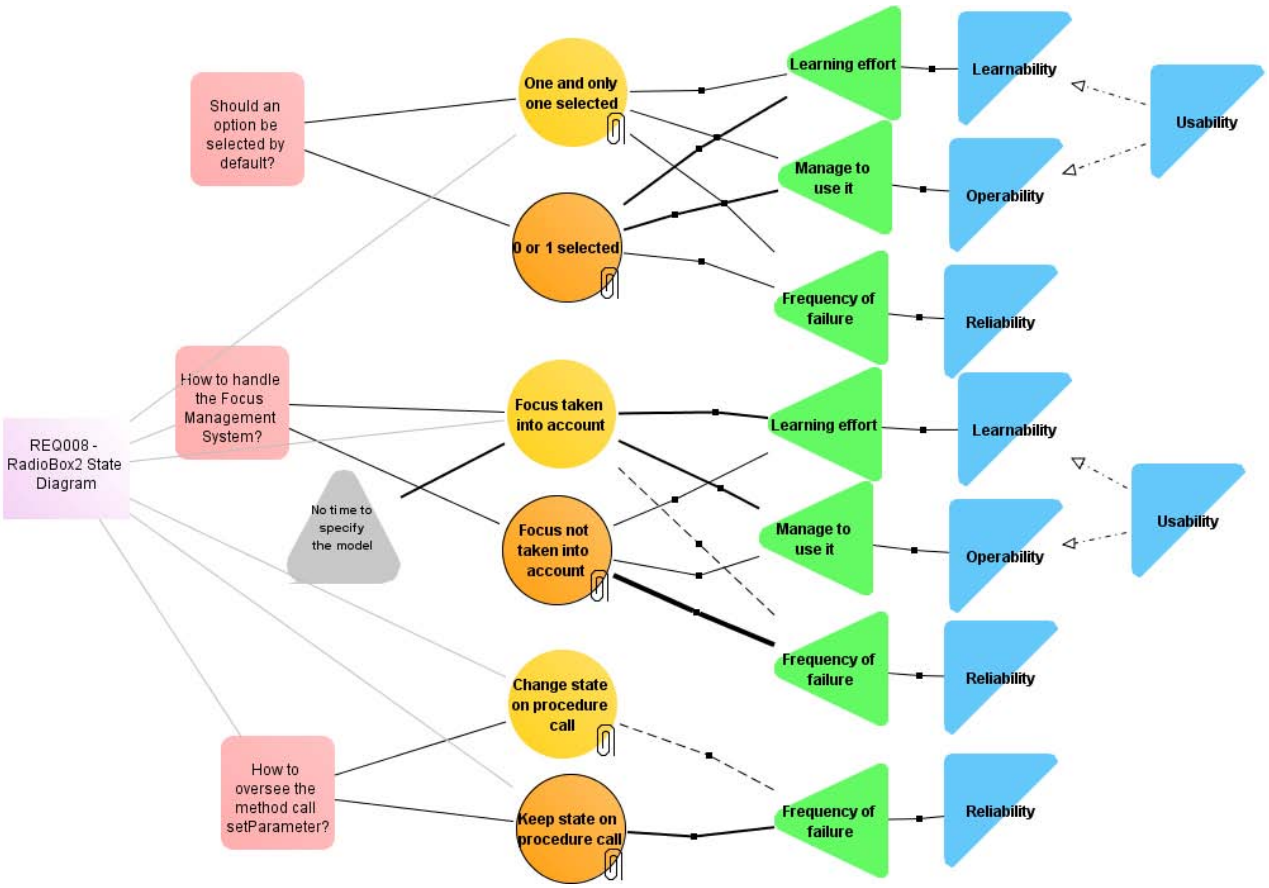


Figure 109. Extrait du diagramme de conception rationnalisée du composant graphique RadioBox2 avec la notation TEAM étendue

L'exigence « REQ008 » est une exigence fonctionnelle portant sur l'aspect graphique du composant graphique RadioBox2. Elle indique que le composant graphique RadioBox2 doit pouvoir changer d'aspect sur réception d'un message interne venant du système de gestion de l'affichage du cockpit. La Figure 109 indique que cette exigence a soulevé deux questions lors de la conception (connexion par des traits gris entre le rectangle « REQ008 » et les carrés « How to handle the Focus Management System ? » et « How to oversee the method call setParameter ? ».

De plus, la Figure 109 montre aussi que les facteurs d'apprentissage (triangle rectangle bleu « Learnability ») et d'opérabilité (triangle rectangle bleu « Operability ») ont été regroupés sous le facteur d'utilisabilité (triangle rectangle bleu « Usability »). Ce regroupement est décrit par les flèches en pointillé partant d'un facteur vers ses sous-facteurs.

La Figure 110 montre le diagramme de traçabilité de l'ensemble des exigences fonctionnelles concernant le composant graphique RadioBox2.

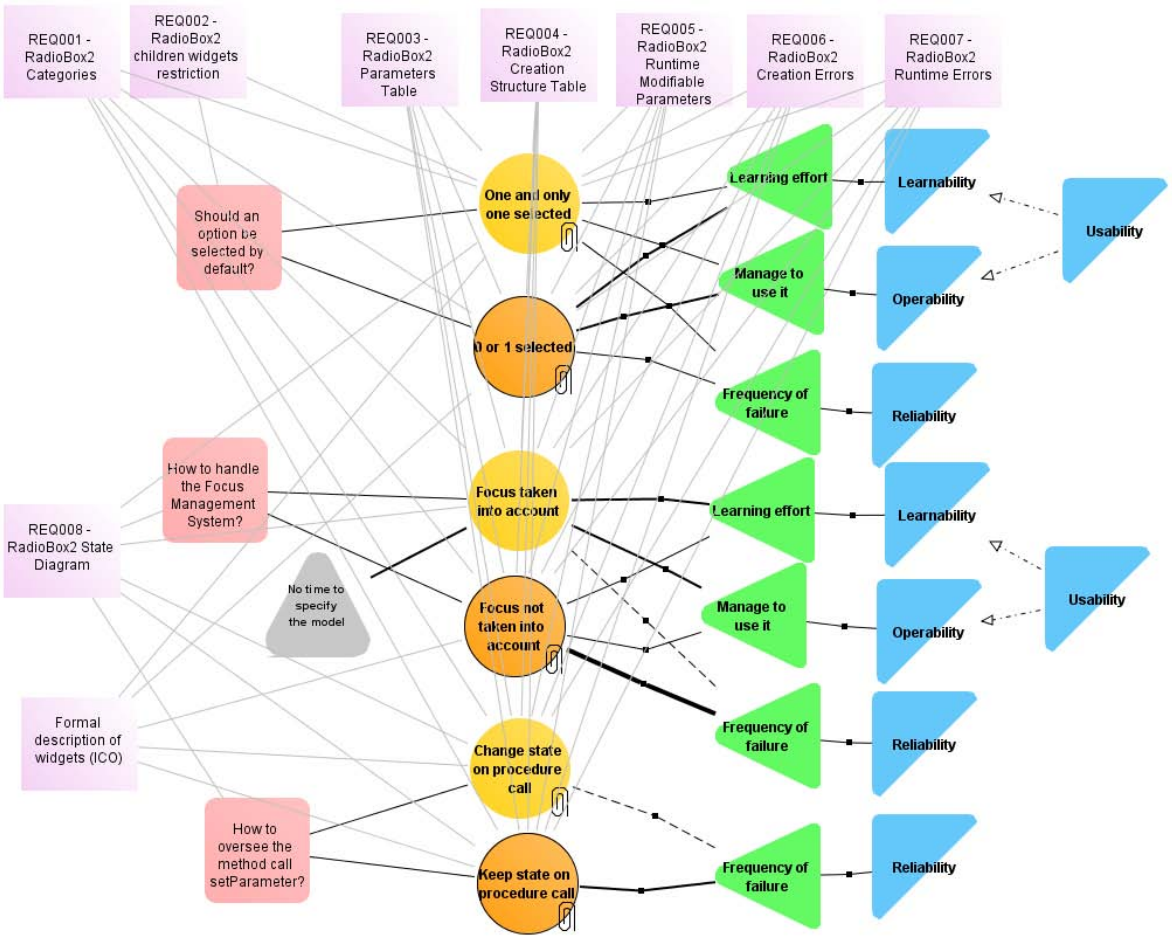


Figure 110. Diagramme complet de conception rationnalisée du composant graphique RadioBox2 avec la notation TEAM étendue

Le diagramme présenté par la Figure 110, même s'il s'agit d'un exemple simple, est visuellement chargé d'un grand nombre d'éléments. La Figure 111 donne un aperçu de ce diagramme dans la nouvelle version de l'environnement de conception rationnalisée DREAM.

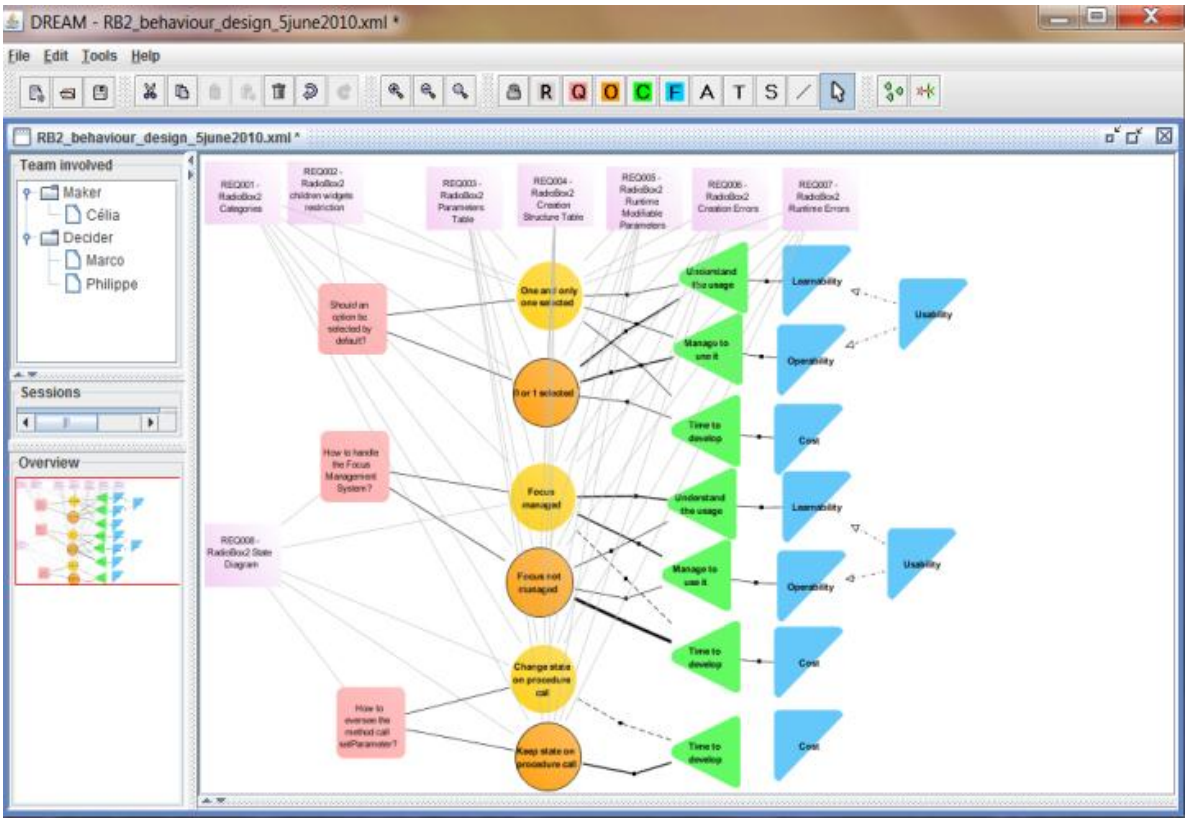


Figure 111. Environnement DREAM (nouvelle version avec la traçabilité des exigences)

Lors du passage à l'échelle vers une étude de cas industrielle, sachant que les diagrammes risquent d'être encore plus chargés, nous proposons de mettre en œuvre de nouveaux moyens de visualisation pour aider les concepteurs et ingénieurs dans leur analyse de la traçabilité des exigences. Ces moyens de visualisation sont inspirés des travaux précédents de (Bertin, 1967) et (Henry & Fekete, 2006). Ils permettent aux utilisateurs de l'environnement DREAM d'examiner les données de conception rationnalisée dans des matrices colorées. Deux types de matrices sont proposés :

- Une matrice de visualisation des options de conceptions et des exigences qu'elles satisfont ou non. Les options de conceptions forment les intitulés des lignes de la matrice et les exigences forment les intitulés des colonnes. Lorsqu'une exigence satisfait une option de conception, la case à l'intersection de cette exigence et de cette colonne s'affiche en vert. Dans le cas contraire, elle s'affiche en rouge. Une matrice de type est présentée dans la Figure 112, à partir des données de notre exemple. Cette figure nous permet d'identifier plus rapidement quelles exigences ne sont pas satisfaites et par quelles options de conception.

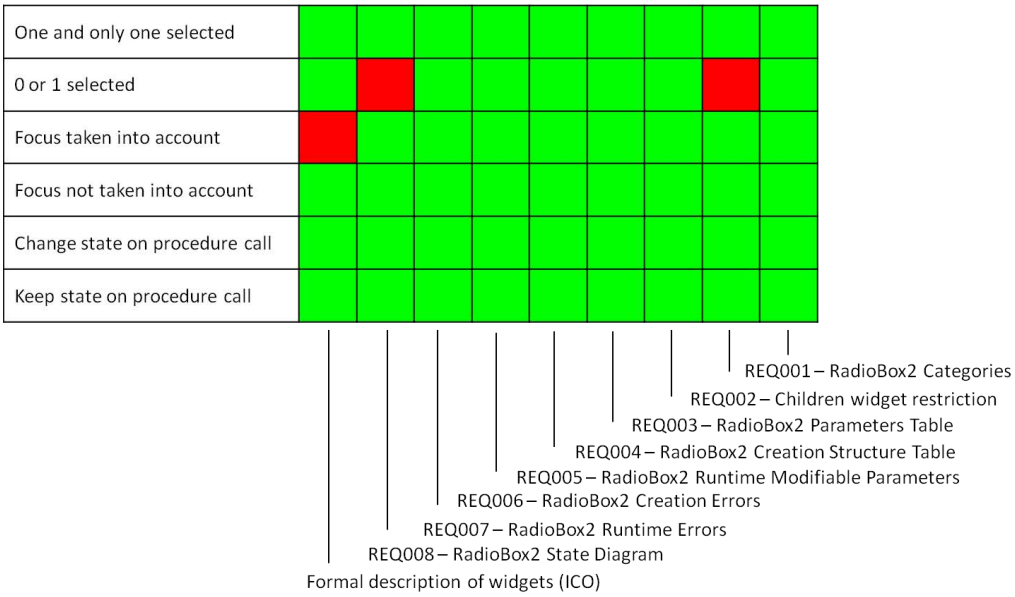


Figure 112. Matrice colorée de visualisation des options de conception (lignes) couvertes et non couvertes par les exigences (colonnes)

- Une matrice de visualisation des résultats d'évaluation de la satisfaction de chaque critère par chaque option liée à ce critère. Les options de conception forment les intitulés des lignes de la matrice et les critères forment les intitulés des colonnes de la matrice. Un lien entre une option et un critère pouvant être pondéré sur une échelle de 1 à 5, les cases de la matrice peuvent s'afficher en 5 couleurs différentes (rouge, rouge foncé, marron, vert foncé, vert). Une matrice de ce type est présentée sur la Figure 113, à partir des données de notre exemple.

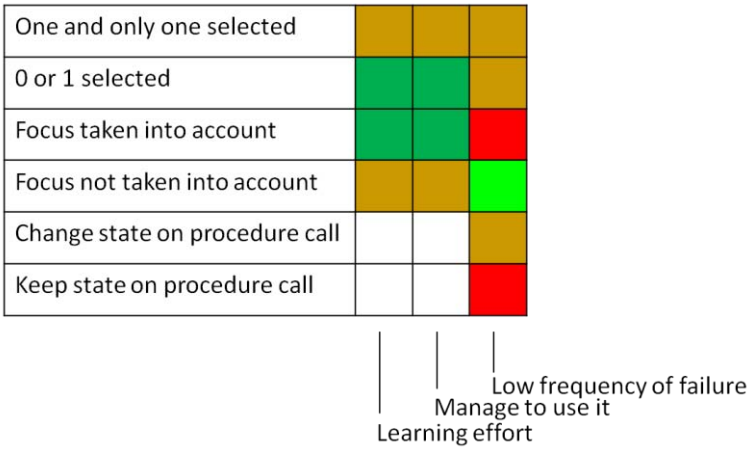


Figure 113. Matrice colorée de visualisation de la pondération des liens entre critères (colonnes) et options (lignes)

Cette section clôt la mise en œuvre du processus de développement d'un système interactif. La traçabilité des exigences et besoins tout au long du processus de développement permet de consigner et capitaliser les informations sur les éléments clés utilisés dans ce processus et donne la possibilité de réutiliser ces informations pour le développement des systèmes interactif critiques.

3 Remarques sur le maintien des compétences et l'aide contextuelle en opérations

Cet environnement de mise en œuvre fournit aussi un support à un entraînement personnel des opérateurs suite à leur prise de fonction et durant leur mission. Ainsi, il peut être utilisé pour s'entraîner périodiquement sur des scénarios basés sur des évènements arrivant peu fréquemment mais très importants, comme une panne d'une partie vitale du système. Cet environnement offre donc une possibilité de maintenir les connaissances et compétences des opérateurs.

D'autre part, cet environnement logiciel pourrait aussi être utilisé pour fournir une aide contextuelle au personnel. En effet, l'opérateur pourrait reproduire le scénario opérationnel de son choix dans l'environnement intégré et utiliser les modèles pour comprendre l'état du système et des activités qu'il devrait mener. Ce type d'utilisation a été introduit par (Palanque, Bastide, & Dourte, 1993). Un exemple d'une telle aide a été mise en œuvre dans l'environnement synergique intégré et est détaillé dans (Palanque & Martinie, 2011).

4 Conclusion

Ce chapitre nous a permis de montrer les techniques et outils permettant de mettre en œuvre notre approche de conception et développement pour les systèmes interactifs critiques et leur programme de formation associé.

Les techniques et outils de la partie génie logiciel de notre contribution reposent sur :

- L'exploitation synergique des modèles de comportement du système et des modèles de tâches utilisateur.
- L'exécution simultanée du prototype très haute-fidélité (qui correspond à une version très proche ou à la version exacte de la partie interactive du système utilisé) et des modèles sous-jacents du comportement du système et des tâches utilisateurs.
- La collecte, la structuration et le filtrage des évènements gérés dans et générés par les modèles en exécution.
- La mise en rapport des besoins et exigences pour le système avec les choix de conception et les modèles associés.

Le chapitre suivant utilise ces techniques et outils pour montrer la faisabilité de notre approche de développement pour les systèmes interactifs critiques.

Chapitre 7 - Etude de cas : Développement d'une application segment sol de commande et contrôle de la mission satellitaire PLEIADES

Ce chapitre présente la mise en œuvre de l'approche proposée par cette thèse sur l'étude de cas d'une application de commande et contrôle d'une mission satellitaire. Une nouvelle version de l'application de commande et contrôle de la mission satellitaires PLEIADES a été conçue et modélisée dans le cadre du projet de recherche Tortuga[7]. Avec les données issues de ce projet, nous avons mis en œuvre :

- L'approche proposée par cette thèse et décrite dans le Chapitre 1.
- Les techniques et leur environnement logiciel décrit dans le Chapitre 1.

L'acronyme Tortuga signifie "Tasks, Operations, Reliability and Training for Users of Ground Applications". Ce projet de recherche, auquel nous avons participé, a pour but d'étudier l'utilisabilité et la fiabilité des systèmes interactifs de commande et contrôle de différentes missions satellitaires du CNES (Centre National d'Etudes Spatiales) et de fournir un support à l'amélioration de ces propriétés pour le moyen terme et le long terme. Ce projet a aussi pour but d'améliorer la fiabilité des opérateurs utilisant ces systèmes, notamment par le biais de la formation.

Dans la première section de ce chapitre, nous présentons le contexte et les objectifs de la mission satellitaire PLEIADES et démontrons que les applications de commande et contrôle de ce type de mission satellitaire possède les caractéristiques d'un système interactif complexe et critique (telles que décrites dans le chapitre 1). La seconde section est consacrée à la démonstration de la faisabilité de l'approche proposée par cette thèse pour concevoir et développer une application de commande et contrôle et de son programme de formation associé. La faisabilité de cette approche est démontrée grâce à la mise en œuvre des techniques et outils logiciels de l'environnement décrit dans le chapitre 6.

1 Description du segment sol de la mission PLEIADES

PLEIADES est un système satellitaire d'observation de la terre très haute résolution. Un premier lancement est prévu pour la fin de l'année 2011 et l'exploitation de la mission doit permettre de recueillir des données de cartographie fine pouvant compléter la photographie aérienne. Ce satellite doit fournir, entre autres, un support aux missions de défense et de sécurité civile.

Les opérateurs de PLEIADES, comme pour toute autre mission satellitaire, doivent acquérir des connaissances et compétences de détection de pannes et des connaissances sur les activités à mener dans ces cas de panne avant d'être autorisés à surveiller une mission satellitaire (ce prérequis a été décrit dans le chapitre 3).

[7] http://www.irit.fr/recherches/ICS/projects/tortuga/

1.1 Contexte d'une mission satellitaire

La Figure 114 décrit les différentes entités composant un système satellitaire selon le standard ECSS-E-70C (European Cooperation for Space Standardization, 2008) :

- Le **segment spatial** concerne le satellite. Il est composé d'une partie plateforme et d'une partie charge utile. La partie plateforme a pour fonctions de maintenir le satellite à poste en prenant en charge les éléments vitaux du satellite comme la batterie et la gestion des panneaux solaires. La partie charge utile a pour fonction d'effectuer les opérations définies par la mission. Ces deux parties sont composées d'éléments matériels, électroniques et logiciels.
- Le **segment sol** comprend une ou plusieurs stations de télécommunications ainsi qu'un ou plusieurs centres de commande et contrôle. Il comprend aussi un centre d'exploitation des données de la mission.

Pour cette étude de cas, nous nous intéressons au système sol de contrôle des opérations. Cette entité est chargée de maintenir le satellite en opérationnel afin que la mission de récupération de données de cartographie dans notre cas puisse être accomplie.

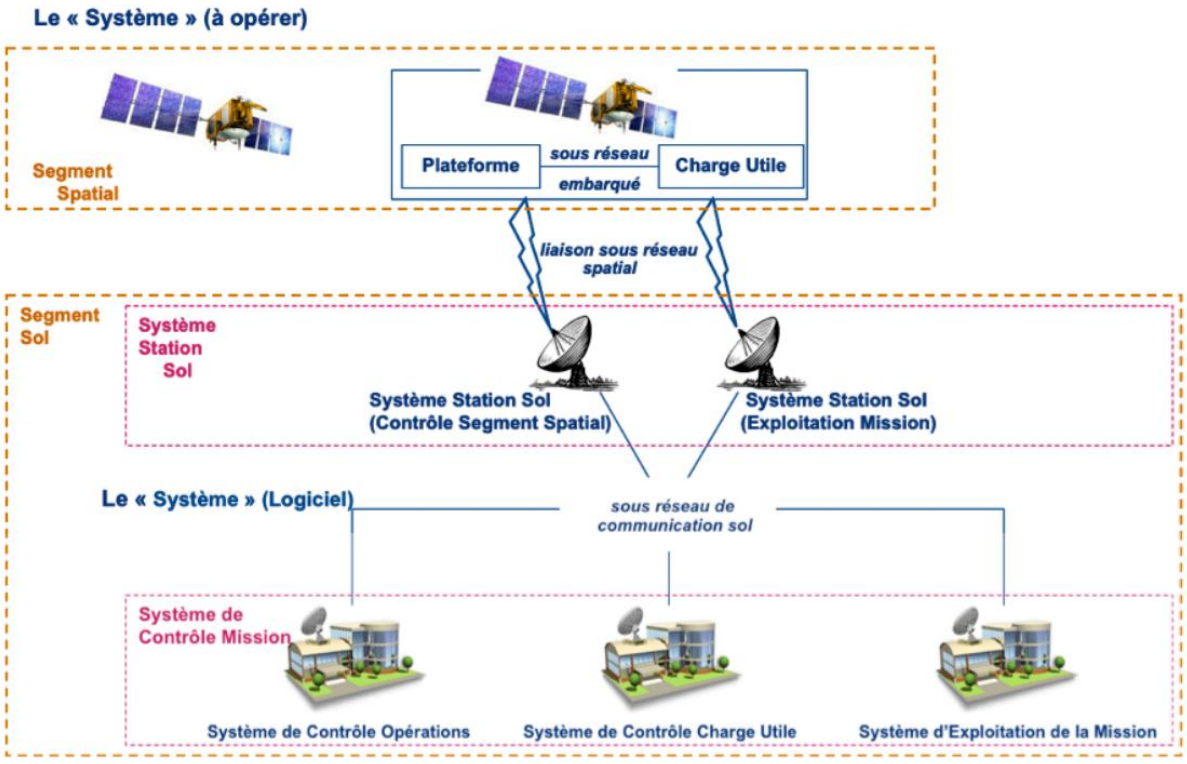


Figure 114. Les différentes entités d'un système satellitaire

1.2 Opérations dans un centre de commande et contrôle

Un centre de commande et contrôle des opérations (présenté comme « Système de Contrôle Opérations » sur la Figure 114) contient des systèmes informatiques et divers équipements électroniques dédiés à :

- La réception des informations en provenance du satellite et transitant par la station sol de télécommunication, appelée **télémesure ou TM**.
- L'archivage de ces informations.
- L'envoi de commandes au satellite, appelée **télécommande ou TC**.

Un satellite est dit en période de visibilité lorsque sa position par rapport à la station sol de communication permet de recevoir des télémesures ou de transmettre des télécommandes. L'intervalle temporel de réception des télémesures est plus grand que l'intervalle temporel d'envoi des télécommandes.

Dans ces centres de contrôle, les opérateurs sont chargés de surveiller l'état de la plateforme du segment spatial, en examinant régulièrement plusieurs paramètres tels que la tension des générateurs de courant, le mode du satellite et l'état du lien de communication. L'objectif principal des opérateurs est de maintenir le satellite dans un mode nominal, de manière à ce que la mission puisse être accomplie sans retard.

Lorsqu'un incident se produit, le satellite peut automatiquement changer de mode de fonctionnement et passer dans un état appelé « survival mode ». Cette transition est aussi appelée « passage en survie » et indique que seules les fonctions vitales de gestion de l'énergie et de communication restent actives mais que la mission est interrompue. Un des objectifs des opérateurs est d'éviter ce changement de mode, excepté si ce changement permet d'éviter de perdre le satellite. Lorsqu'un opérateur détecte un incident (généralement un échec du système), il/elle doit tenter de comprendre et résoudre cet incident. Si l'incident est plus compliqué à comprendre, l'opérateur doit informer immédiatement l'équipe afin d'initier une investigation collaborative.

Afin d'aider les opérateurs dans leurs tâches quotidiennes de routine, maintenance et résolution de problème, des **procédures opérationnelles** sont définies pour chaque mission et fournissent des lignes directives pour interagir avec le satellite via l'application segment sol de commande et contrôle. Ces procédures opérationnelles contiennent la spécification textuelle des tâches à effectuer par l'opérateur et des commandes à entrer dans le système selon les activités de routine, de maintenance ou de résolution de pannes. **Certaines parties** de ces procédures sont **automatisées** et stockées sous forme de fichiers de télécommandes exécutables par certaines applications segment sol.

1.3 Caractéristiques et propriétés des applications segment sol de commande et contrôle

Les applications segment sol de commande et contrôle **fournissent des informations** aux opérateurs :

- L'état des paramètres de surveillance de la partie plateforme du satellite.
- L'état des paramètres de surveillance de la partie charge utile du satellite.
- L'état des paramètres de surveillance des stations de communication.
- L'état des paramètres de surveillance de l'infrastructure réseau et applicative des centres de commande et contrôle.

Le nombre total de l'ensemble de ces paramètres dépasse le millier.

Les applications segment sol de commande et contrôle permettent aux opérateurs de **planifier et d'effectuer des procédures** de routine, de maintenance et de résolution de problèmes. Le nombre total de l'ensemble de ces procédures se chiffre en centaine et ces procédures peuvent être combinées pour des besoins particuliers de maintenance et de résolution de pannes.

Une application segment sol possède donc les caractéristiques suivantes :

- Elle est interactive et complexe car elle permet de surveiller et contrôler divers éléments matériels, électroniques et logiciels distribués entre le segment sol et le segment spatial.
- Elle doit permettre aux utilisateurs de gérer des quantités de données extrêmement importantes.

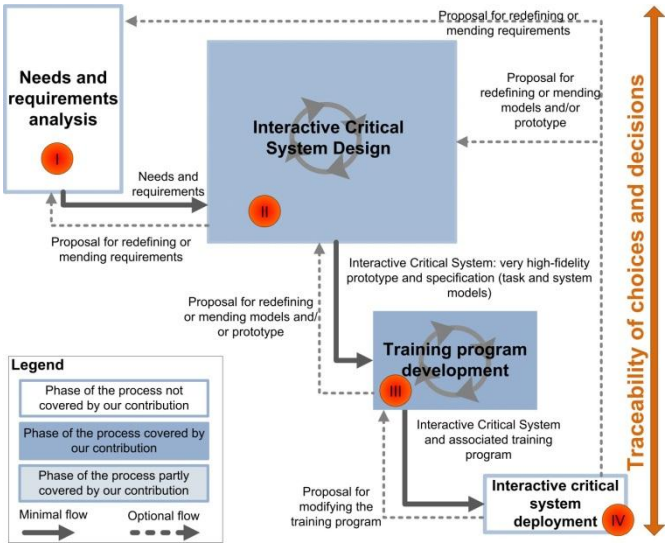
- Elle est critique car le coût de la perte d'un satellite en termes de matériels de mission et de sécurité, ne doit pas dépasser le coût de son développement. Les opérateurs sont formés et qualifiés avant d'être assignés sur une mission.

De par ses caractéristiques, une application segment sol doit remplir les propriétés suivantes :

- Elle doit être fiable pour prévenir la perte de la mission temporaire ou définitive.
- Elle doit être utilisable et permettre aux opérateurs d'opérer le système avec efficacité et efficience pour mener à bien leur mission.
- Son opérabilité doit pouvoir être garantie avant utilisation.

Une application segment sol de commande et contrôle d'une mission satellitaire est donc une candidate idéale pour démontrer la faisabilité de l'approche proposée dans cette thèse.

2 Application du processus de développement au segment sol PLEIADES



Cette section décrit la mise en œuvre du processus de développement et de l'environnement synergique de modélisation pour concevoir et développer une application de commande et contrôle de la mission satellitaire PLEIADES. Elle reprend les différentes étapes de l'approche proposée par cette thèse, de l'analyse des besoins et exigences au développement du programme de formation associé. Nous utilisons les techniques de mise en œuvre décrites dans le chapitre précédent mais ne les détaillons pas de manière approfondie puisqu'elles font l'objet du chapitre 6. Cependant, nous nous attachons à montrer la faisabilité de cette approche en précisant les artefacts et modèles produits par chaque étape du processus grâce aux techniques et à l'environnement de mise en œuvre.

Les résultats présentés dans les sous-sections 2.1, 2.3 et 2.4 utilisent en partie les travaux effectués dans le cadre d'un projet tutoré (Africha, Haffane, Lassalle, & Riegert, 2011), que nous avons supervisé dans le cadre du projet de recherche Tortuga.

2.1 Analyse des besoins et exigences

Une application segment sol de commande et contrôle existe déjà pour la mission PLEIADES et les opérateurs de cette mission ont émis des souhaits d'amélioration pour cette application. L'interface de l'application existante est présentée sur la Figure 115.

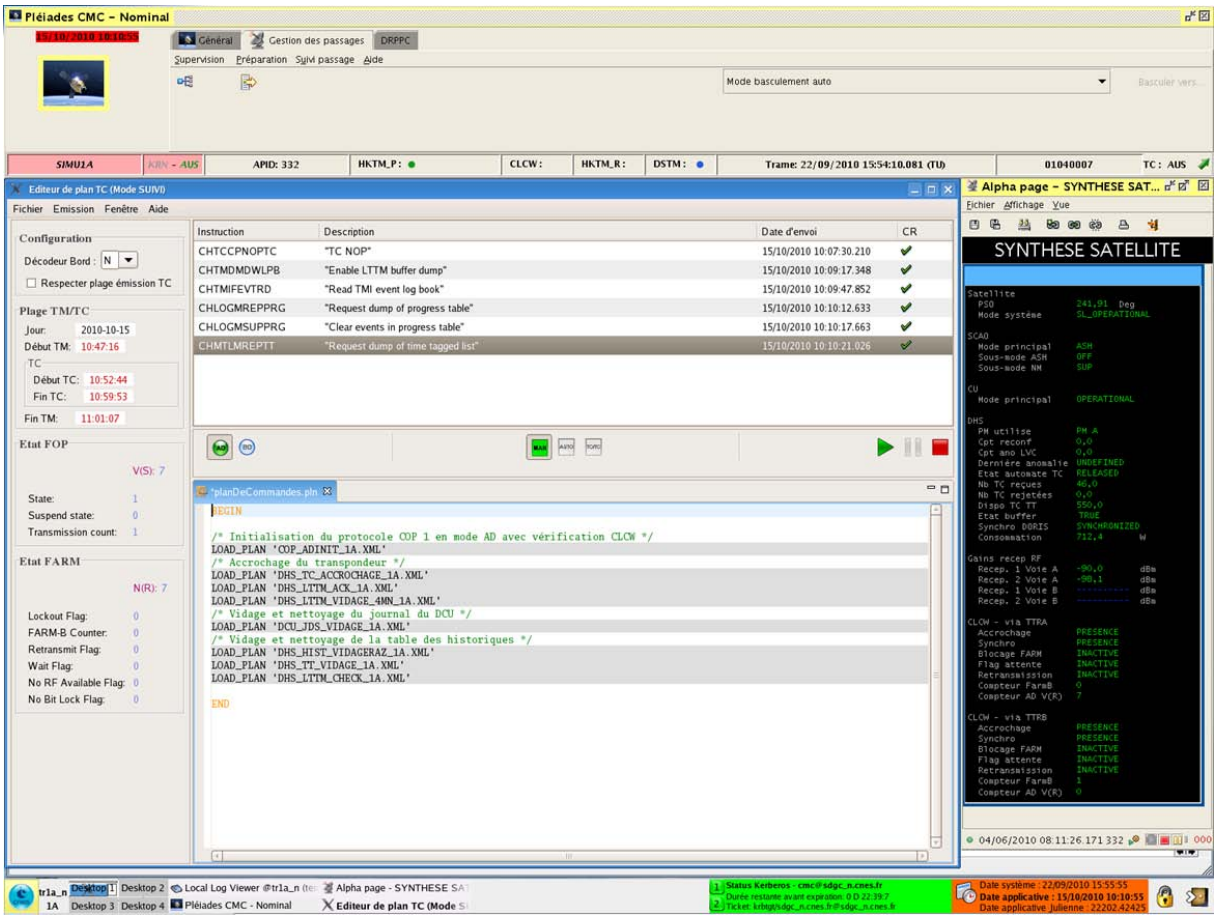


Figure 115. Copie d'écran de l'interface de l'application segment sol de commande et contrôle de la mission PLEIADES

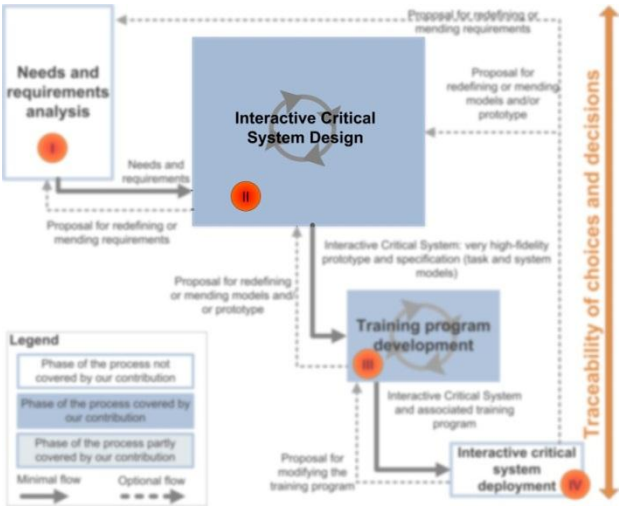
Plusieurs techniques ont été utilisées pour identifier plus précisément les besoins d'amélioration souhaitées par les opérateurs : entretiens avec des experts, analyse documentaire, questionnaires et observations lors des essais de qualification. Ces techniques ne sont pas détaillées ici car elles ne sont pas l'objet d'une contribution de cette thèse.

Parmi la liste de besoins spécifiés en sortie de cette phase d'analyse, nous retenons un sous-ensemble illustratif pour la suite de notre étude de cas :

a) L'interface devrait guider l'opérateur par rapport aux tâches qu'il doit effectuer avant et pendant l'intervalle de temps ou le satellite est en visibilité.
b) L'interface devrait rendre plus explicite les sous-périodes de visibilité ou la réception de télémesure est possible et où l'envoi de télécommandes est possible.
c) L'interface devrait fournir un moyen graphique symbolique pour la visualisation du temps de visibilité restant.
d) L'interface devrait permettre de rechercher une commande, une fonction de l'application ou un paramètre par mot-clé.
e) L'interface devrait permettre de regrouper les paramètres vitaux des différentes parties du système par zones.

De plus, des scénarios d'utilisation ont été spécifiés pour structurer les informations rassemblées grâce aux entretiens, questionnaires, observation et analyse documentaire.

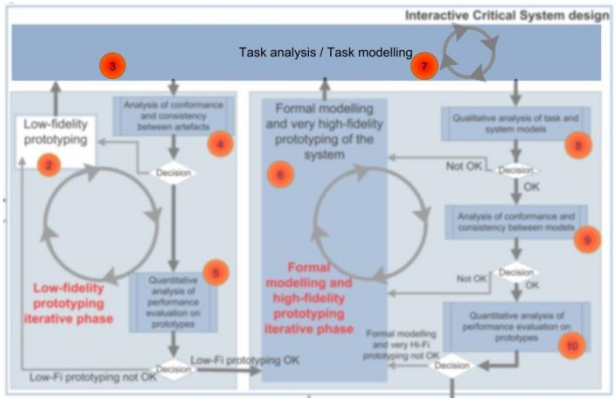
2.2 Conception de l'application segment sol



Cette section illustre de manière détaillée la phase de conception du système interactif critique ayant été présentée de manière abstraite dans le chapitre 5, et dont les techniques, notations et outils de mise en œuvre ont été décrits dans le chapitre 6.

2.2.1 Analyse des tâches d'un opérateur segment sol

Dans le cadre de cette étude de cas, la phase d'analyse de tâches a démarré en même temps que la phase d'analyse des besoins et a servi à raisonner sur les tâches des opérateurs et à mieux comprendre leurs missions. La notation outillée HAMSTERS a servi de support à cette activité d'analyses de tâches. Les modèles ont été édités et simulés avec l'outil logiciel HAMSTERS afin de comparer les scénarios d'utilisation de l'application segment sol avec les scénarios générés avec la fonctionnalité de simulation d'HAMSTERS. Les activités de commande et contrôle étant nombreuses, les mécanismes de structuration *copy task* et *subroutine* décrits dans la section 2.2 du chapitre 4 sont utilisés pour cette activité de modélisation.



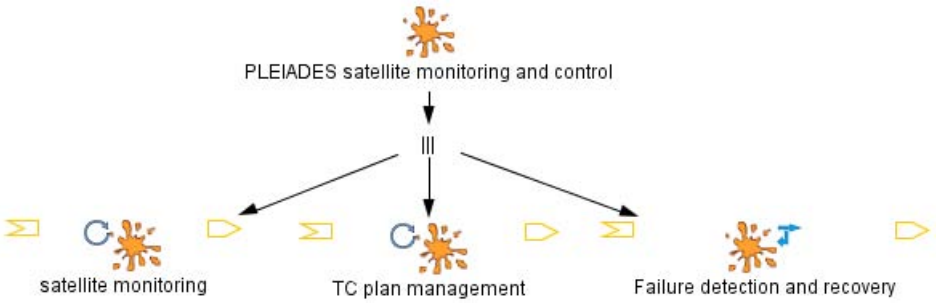


Figure 116. Modèle de tâches des activités principales d'un opérateur de la mission PLEIADES

La Figure 116 décrit les trois tâches principales d'un opérateur de la mission PLEIADES :

- La surveillance de l'état du satellite « satellite monitoring » (représentée par une tâche de type *subroutine*). Elle est détaillée dans la section 2.2.2.
- La gestion des plans de télécommandes « TC plan management » (représentée par une tâche de type *subroutine*). Elle est détaillée dans la section 2.2.1.

- La détection et résolution de pannes satellite « Failure detection and recovery » (représentée par une tâche de type *subroutine*). Elle est détaillée dans la section 2.2.2.

2.2.1.1 Gestion des plans de télécommandes

La gestion des plans de télécommandes est représentée par la subroutine « TC plan management » sur la Figure 117. Les tâches principales pour gérer un plan de télécommandes sont : la préparation du plan de télécommandes, la gestion du déroulement de ce plan lorsque le satellite passe en période de visibilité puis l'analyse post-passage du déroulement du plan de télécommandes.

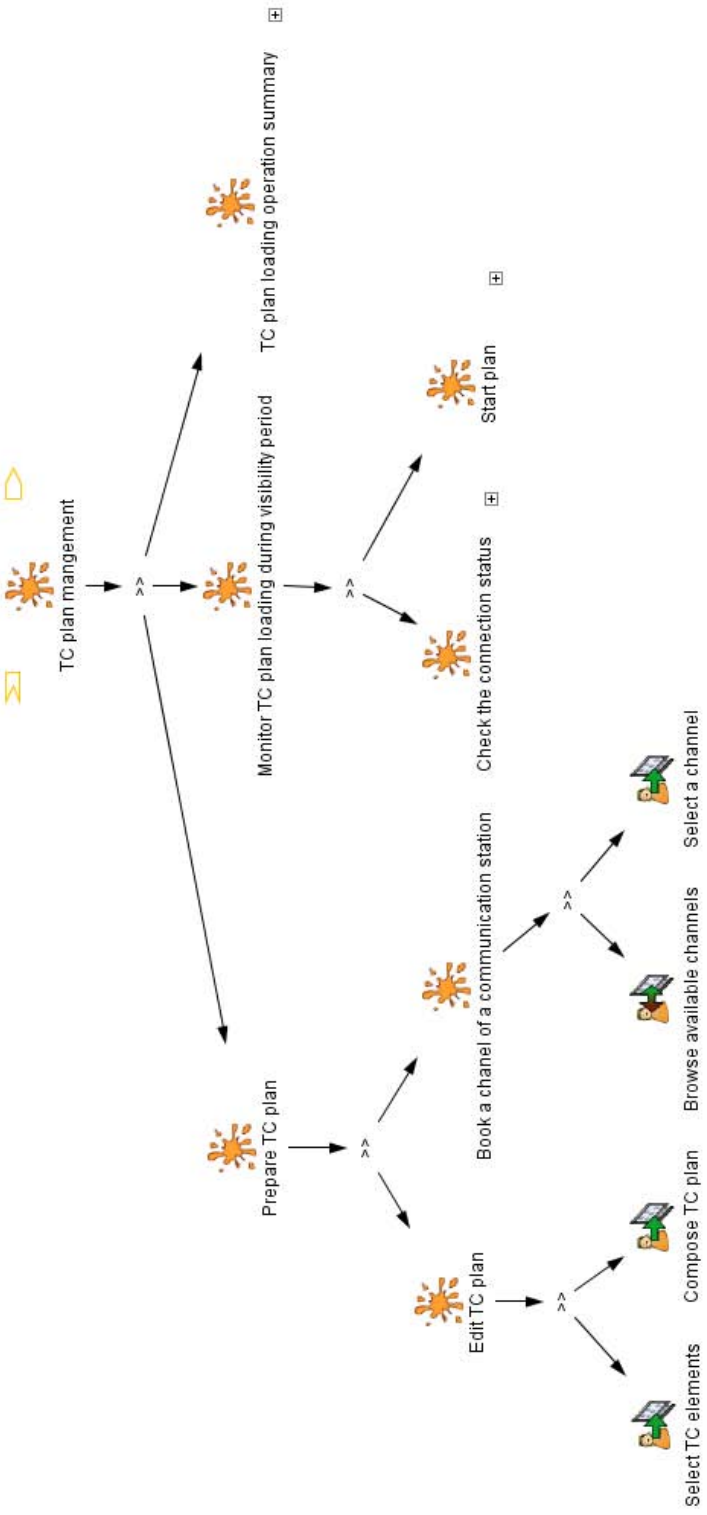


Figure 117. Modèle de tâches de la *subroutine* de gestion des plans de télécommandes

2.2.1.2 Opérations de surveillance et de résolution d'une panne

Les activités de surveillance des paramètres vitaux du satellite sont exposées dans la Figure 118.

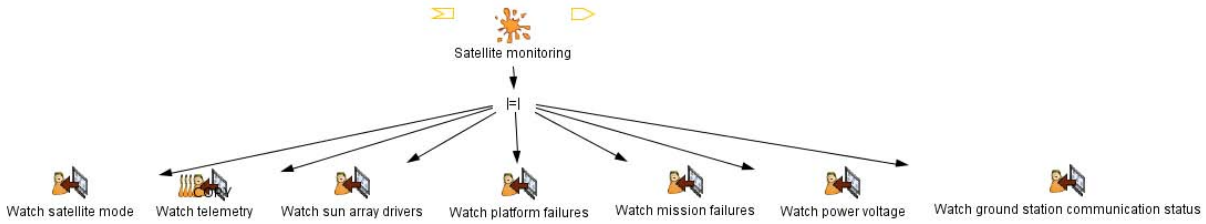


Figure 118. Modèle de tâche de la *subroutine* de surveillance des paramètres vitaux du satellite

Les paramètres vitaux surveillés le plus fréquemment sont (chacun est représenté par une tâche interactive de sortie) :

- Le mode du satellite afin de vérifier qu'un changement d'état vers le mode dit de « survie » ne s'est pas produit : « Watch satellite mode ».
- La présence de télémesure lorsque le satellite est en visibilité : « Watch telemetry ».
- La position des moteurs d'entraînement des panneaux solaires : « Watch sun array drivers ».
- Le statut des paramètres de pannes de la plateforme : « Watch platform failures ».
- Le statut des paramètres de pannes de la mission : « Watch mission failures ».
- La tension batterie : « Watch power voltage ».
- Le statut des liens de communication entre la station de commande et contrôle et la station de communication : « Watch ground station communication status ».

La Figure 119 présente le détail de la *subroutine* de détection et résolution d'une panne du satellite. Cette *subroutine* est elle-même composée de différents types de tâches mais aussi d'autres *subroutine*s permettant d'utiliser plusieurs fois un même ensemble de tâches sans avoir à le décrire plusieurs fois, comme c'est le cas de la tâche « Record failure », tâche effectuée par un opérateur après chaque incident survenu pendant la mission. Les tâches composant la subroutine « Record failure » sont décrites dans la Figure 121.

La Figure 120 présente le détail de la *subroutine* décrivant l'utilisation de la procédure de basculement d'un moteur d'entraînement (aussi appelé SADA pour Solar Array Drive Assembly) défaillant vers le moteur d'entraînement redondant.

La Figure 122 décrit les tâches composant la *subroutine* de réinitialisation du logiciel de vol et la Figure 123 décrit les tâches composant la *subroutine* de gestion du lien de communication entre la station sol et le satellite. Les flux de données, les conditions et assignations portant sur les valeurs des données utilisées permettent de :

- Factoriser les éléments de description des activités des opérateurs.
- Faire transiter les flux de données entre les différentes *subroutines*.

La Figure 124 montre la production de scénarios d'utilisation à partir des modèles de tâches présentés dans les figures précédentes. Elle permet de valider les modèles de tâches en fonction des informations rassemblées dans les phases amont. Ces modèles sont aussi un support pour valider la compréhension des tâches par les concepteurs de l'application lors des discussions avec les experts métiers des applications segment sol.

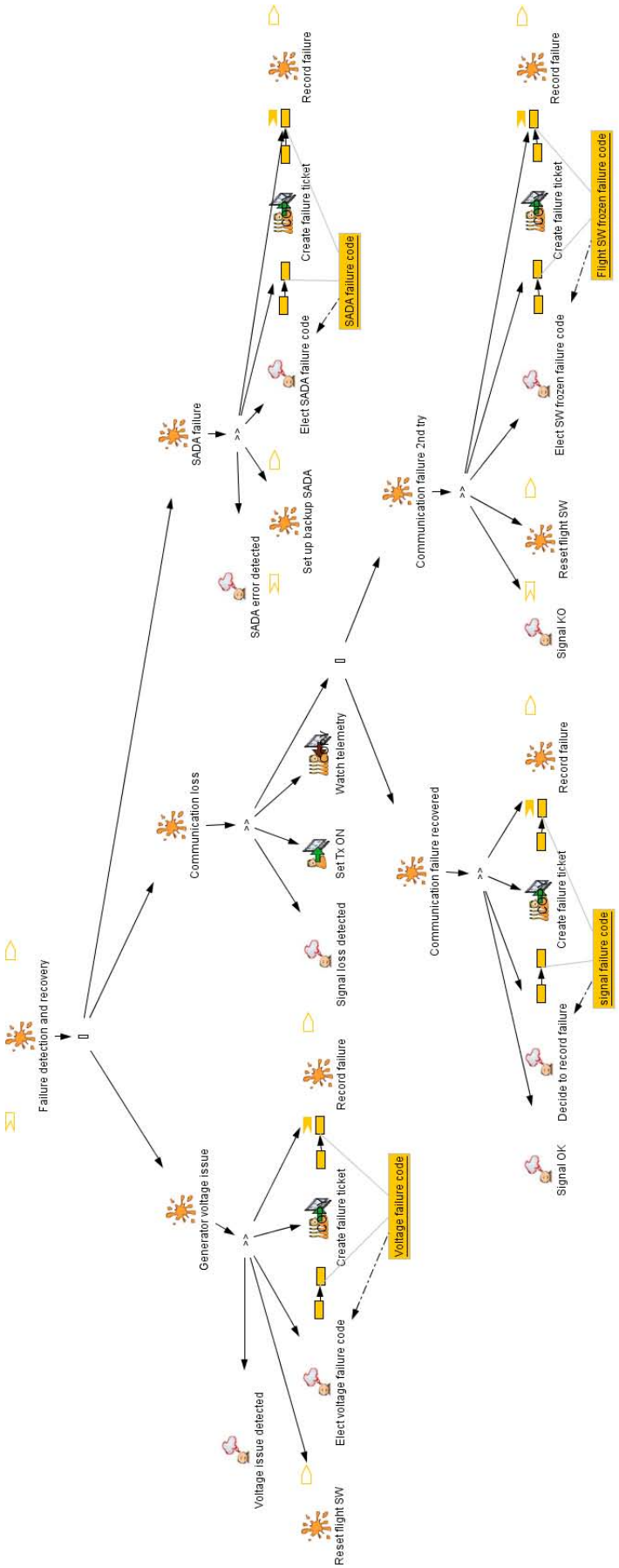


Figure 119. Modèle de tâches de la *subroutine* de détection et résolution de pannes

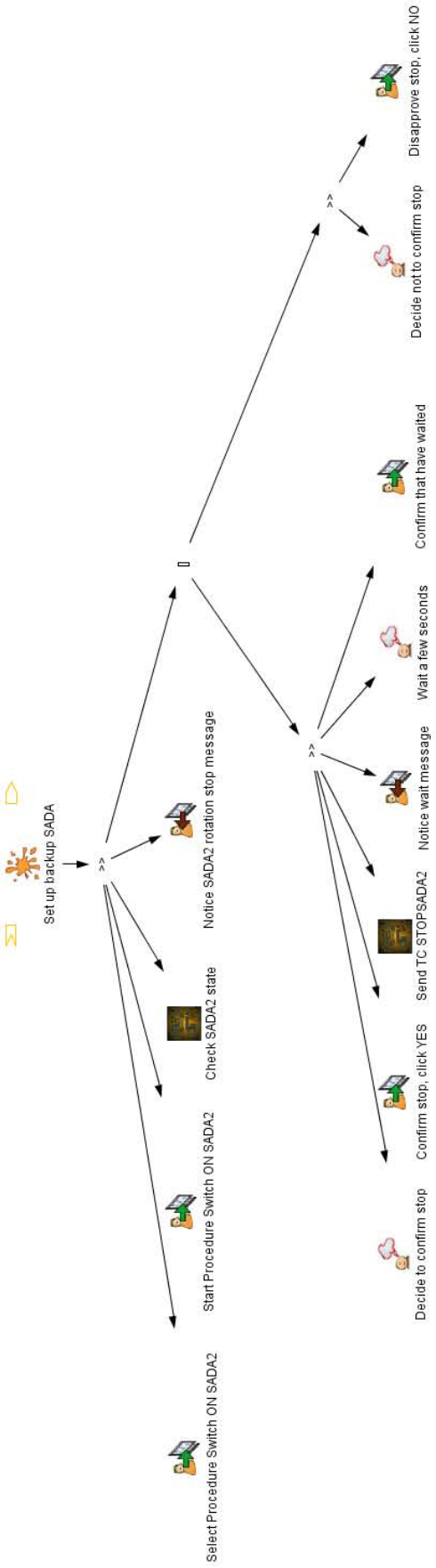


Figure 120. Modèle de tâches de la subroutine d'application de la procédure de changement de moteur d'entraînement des panneaux solaires.

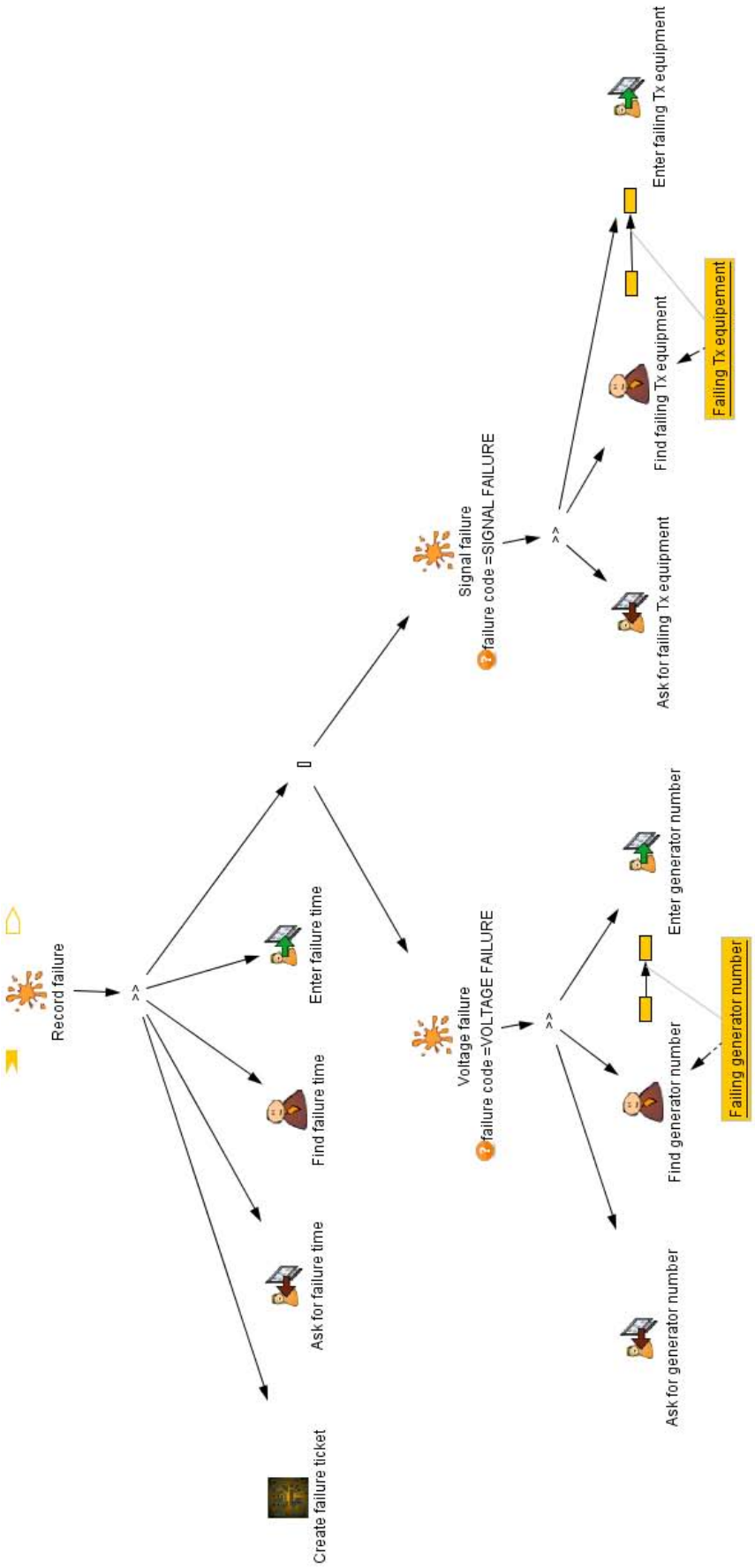


Figure 121. Modèle de tâches de la *subroutine* d'archivage d'une panne

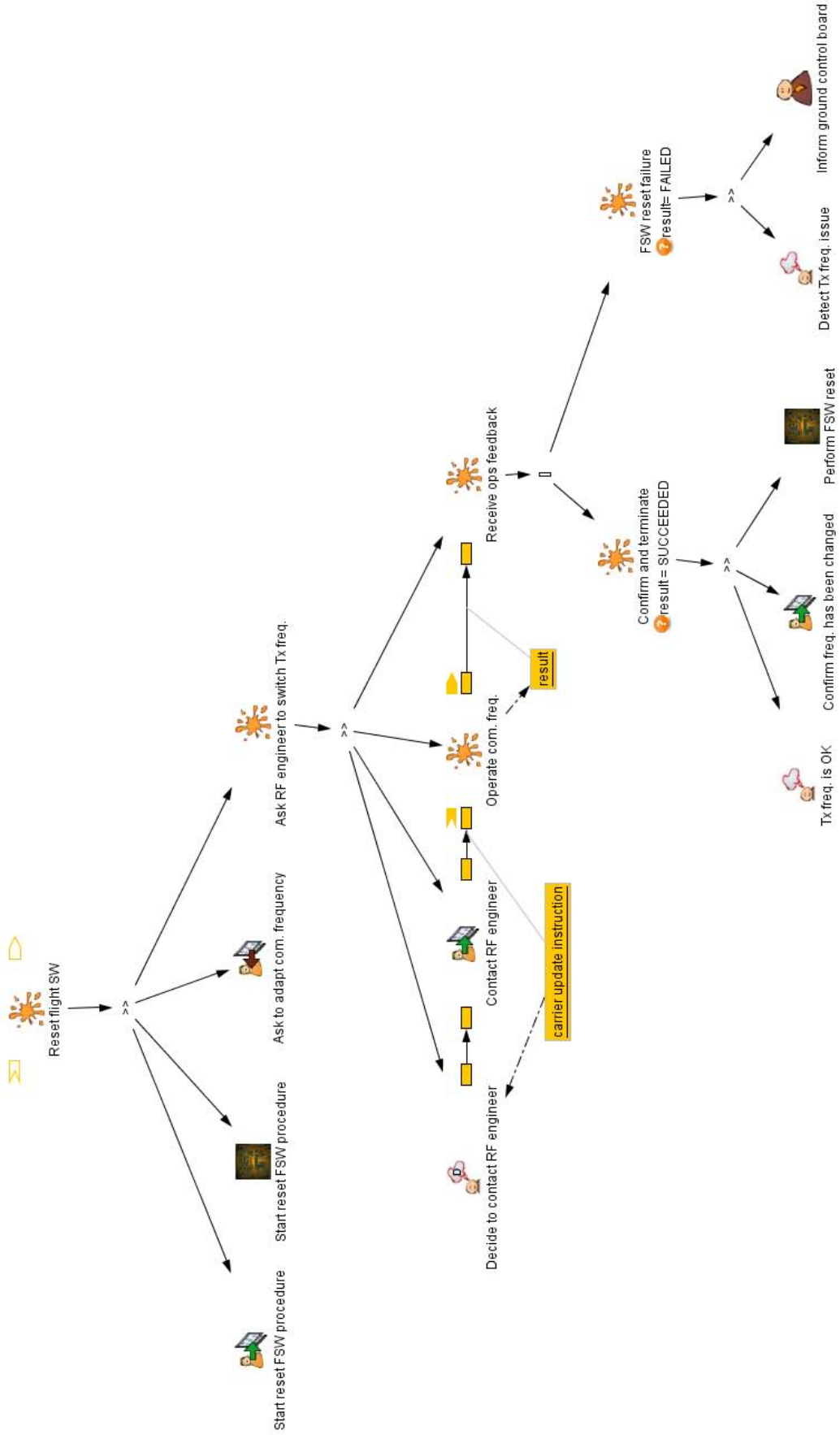


Figure 122. Modèle de tâches de la *subroutine* de réinitialisation du logiciel de vol

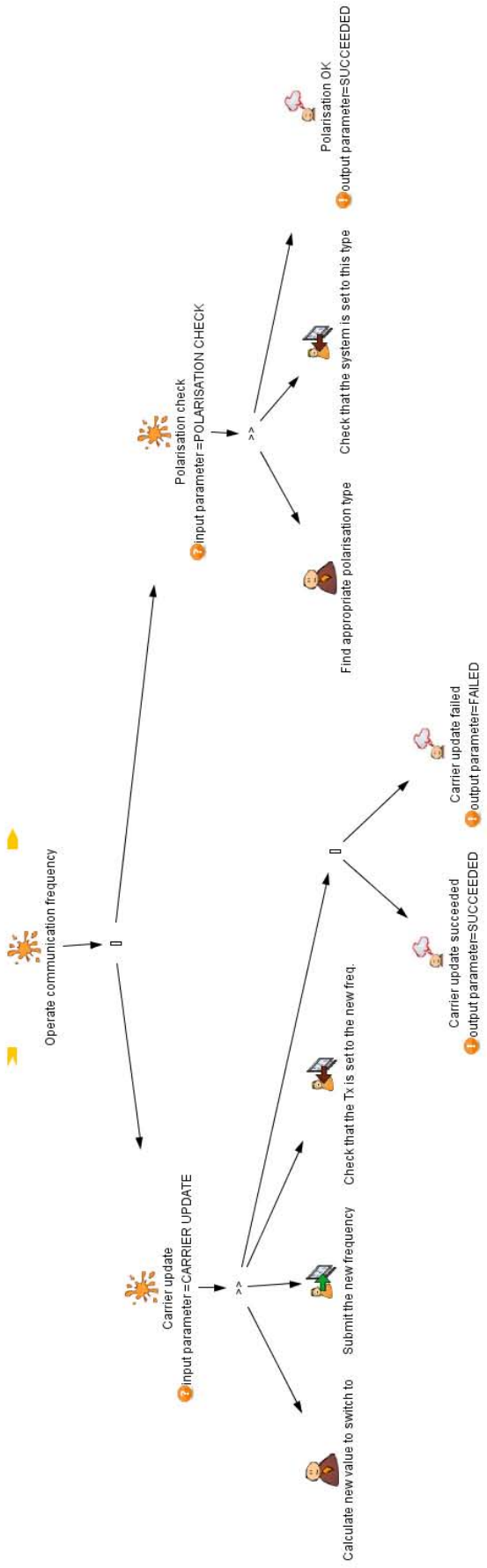


Figure 123. Modèle de tâches de la *subroutine* de gestion du lien de communication entre la station sol et le satellite

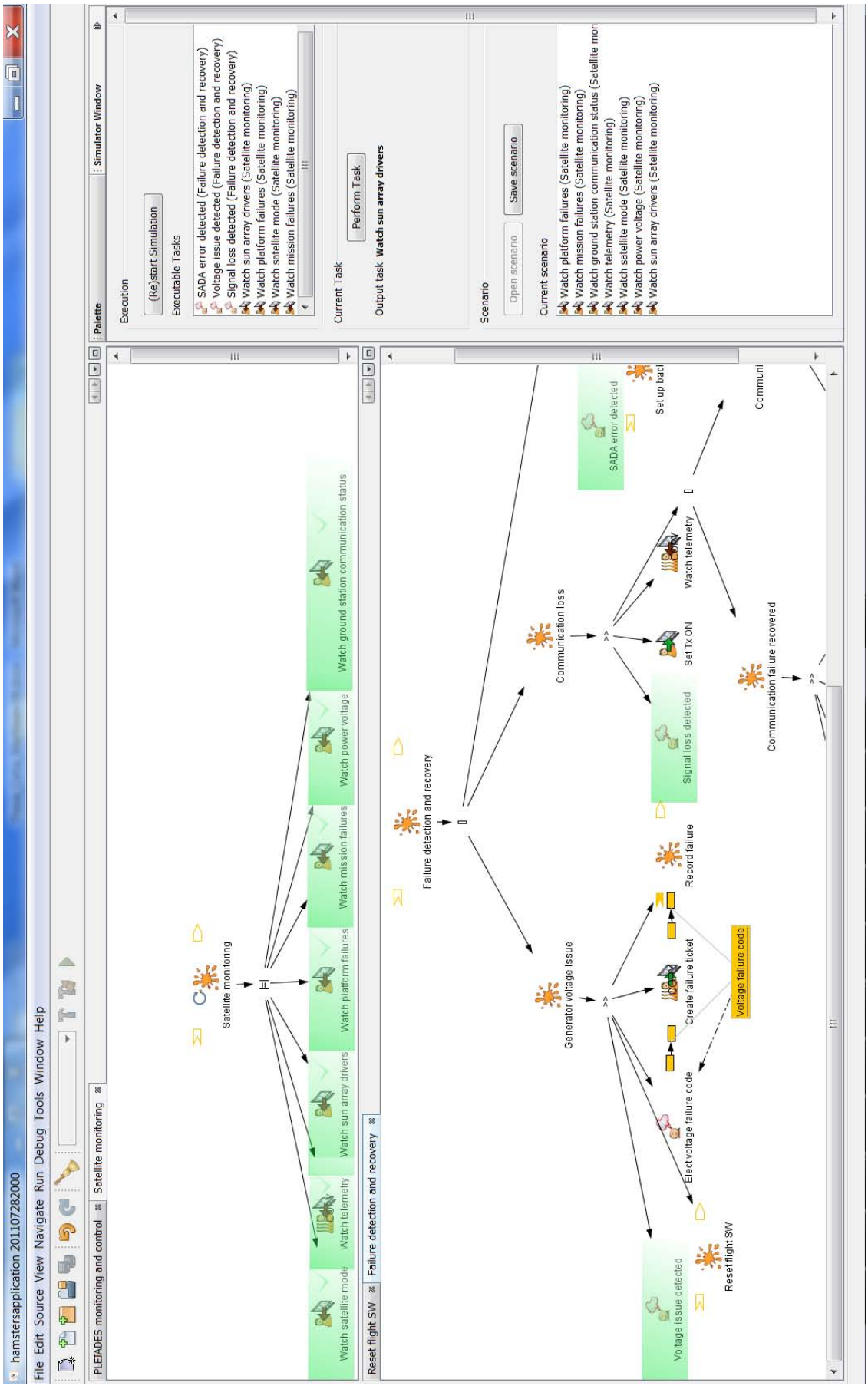
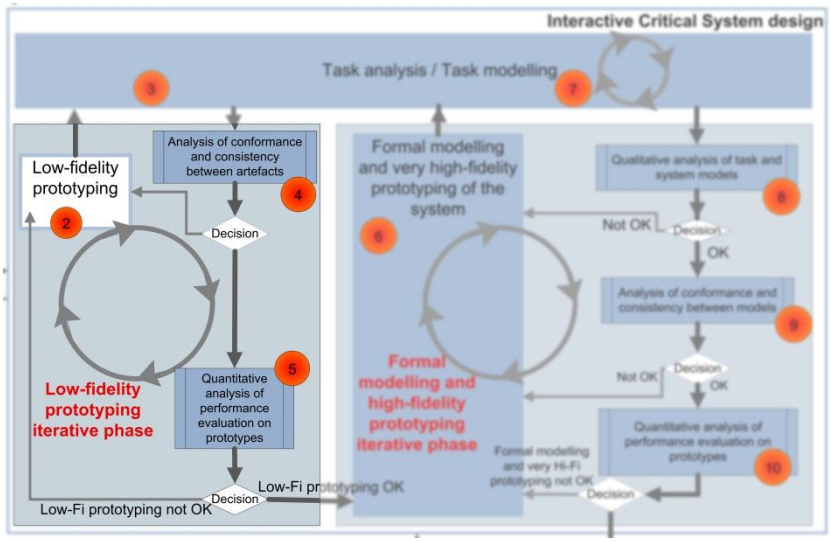


Figure 124. Copie d'écran de la simulation concurrente des modèles de tâches HAMSTERS

2.2.2 Phase itérative de prototypage basse-fidélité



La phase de prototypage basse-fidélité utilise la liste des besoins spécifiée à l'issue de la phase d'analyse des besoins ainsi que les autres artefacts produits (scénarios d'utilisation, résultats des entretiens, questionnaires et observations). Un ensemble de prototypes basse-fidélité ont été conçus puis examinés en fonction des scénarios d'utilisation et des modèles de tâches. Ces prototypes ont ensuite été soumis à l'évaluation d'experts métier du CNES. La Figure 125 présente le prototype basse-fidélité retenu (choisi par un expert métier) à l'issue de la phase de prototypage basse-fidélité.

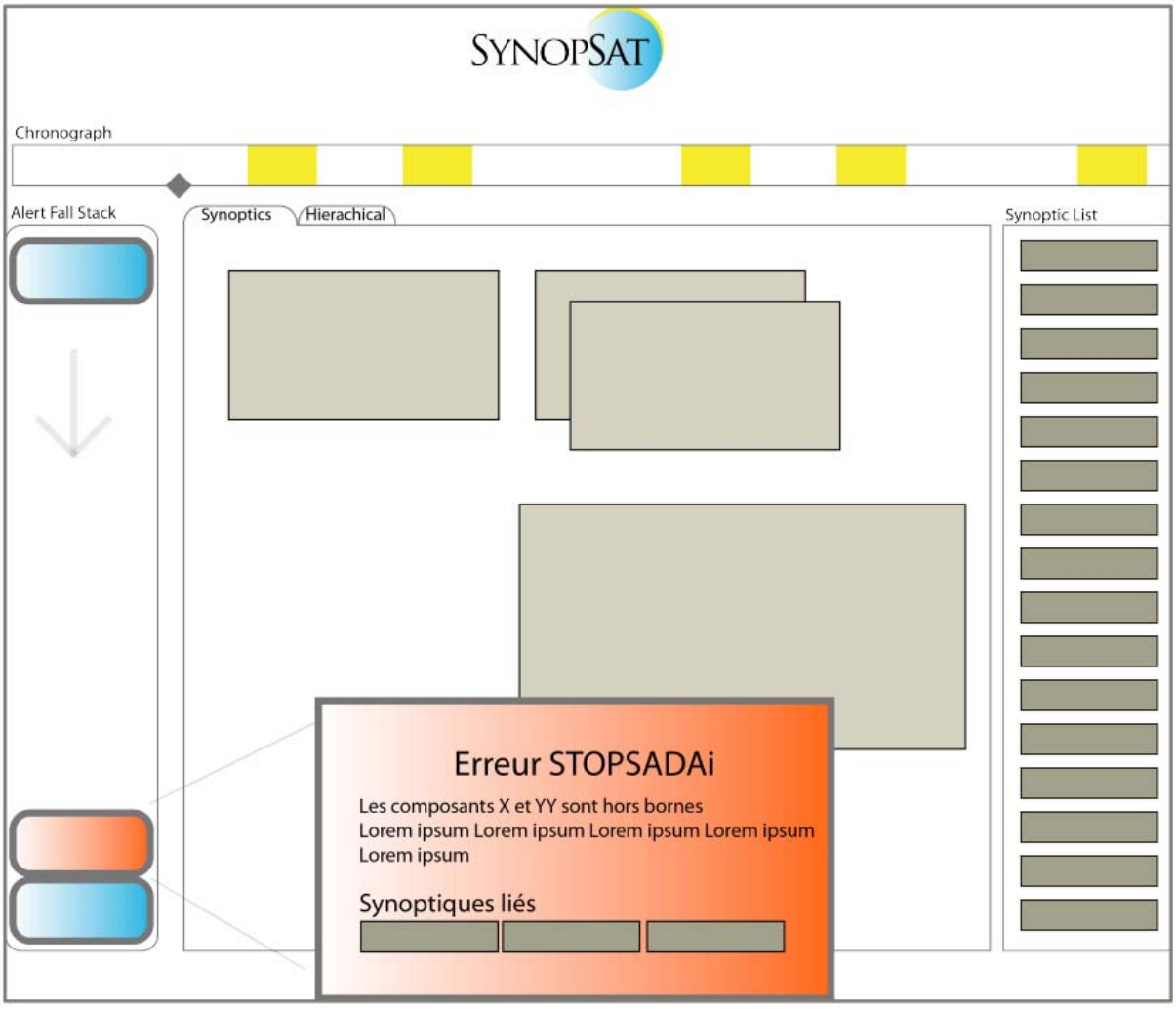
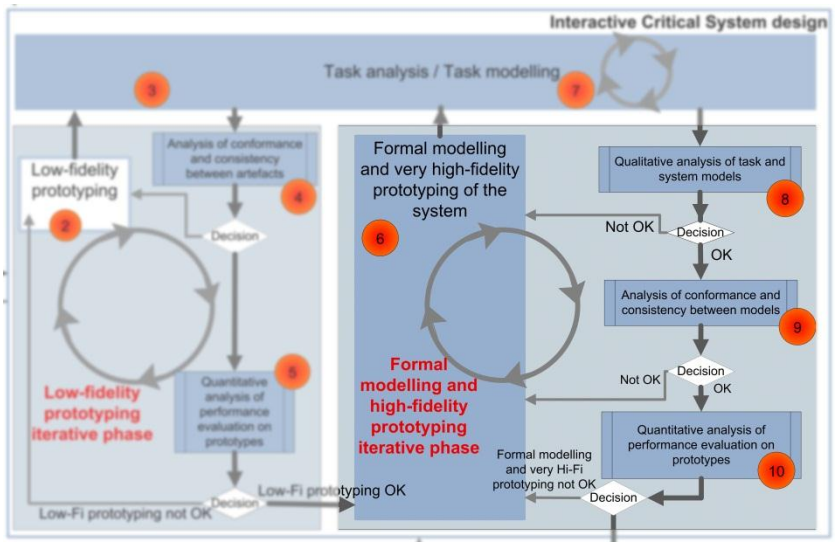


Figure 125. Prototype basse-fidélité retenu à l'issue de la phase de prototypage basse-fidélité

2.2.3 Phase itérative de modélisation formelle et prototypage très haute-fidélité



Cette phase prend en entrée la spécification des besoins utilisateur, le prototype basse-fidélité ainsi que les modèles de tâches. Elle produit les modèles formels du comportement de l'application segment sol et son prototype très haute-fidélité. La mise en correspondance de ces deux types de modèles permet, sur plusieurs itérations, de valider l'adéquation entre les tâches des opérateurs de l'application et le comportement de l'application.

Dans cette section, nous présentons le prototype très haute-fidélité produits avec la technique de modélisation formelle ICO (dont la mise en œuvre a été décrite dans la section 3.3 du chapitre 6). Grâce à cette technique, les interfaces du prototype d'application segment sol et les modèles de comportement des différents éléments de son interface sont produits à l'issue de cette phase de modélisation formelle et prototypage très haute-fidélité. Les interfaces sont les éléments de présentation des modèles ICO et les modèles formels de comportement sont les objets coopératifs des modèles ICO. Les fonctions de rendu et d'activation font le lien entre les parties présentation et les objets coopératifs.

Deux types de modèles formels de comportement ont été produits à l'issue de cette phase :

- Les modèles formels ICO du comportement de l'application segment sol.
- Les modèles formels ICO des parties automatisées des procédures utilisées par les opérateurs. Ces procédures mettent en œuvre des télécommandes pour contrôler le satellite. Les parties non automatisées de ces procédures sont décrites dans les modèles de tâche.

Dans cette section, nous avons choisi de décrire le modèle formel de comportement d'une procédure que les opérateurs peuvent être amenés à utiliser dans le cas d'une résolution de panne du moteur d'entraînement des panneaux solaires (aussi appelés SADA – Solar Array Driver Assembly). Nous avons fait ce choix en fonction des éléments suivants :

- La procédure de résolution de basculement d'un moteur d'entraînement défaillant à un autre correspond à la procédure utilisée pour décrire la mise en œuvre de de la phase de développement du programme de formation (exposée dans la section 4 du chapitre 6). Et nous souhaitons éviter la démultiplication d'exemples afin de ne pas noyer notre contribution dans trop de détails techniques liés au domaine d'application spatial.
- Un exemple de modèle ICO de comportement d'une interface a déjà été présenté dans le chapitre précédent (dans la section 3.3 du chapitre 6).

De plus, cette section ne détaille pas les fonctions de rendu et d'activation (des exemples détaillés sont exposés dans la section 3.3 du chapitre 6).

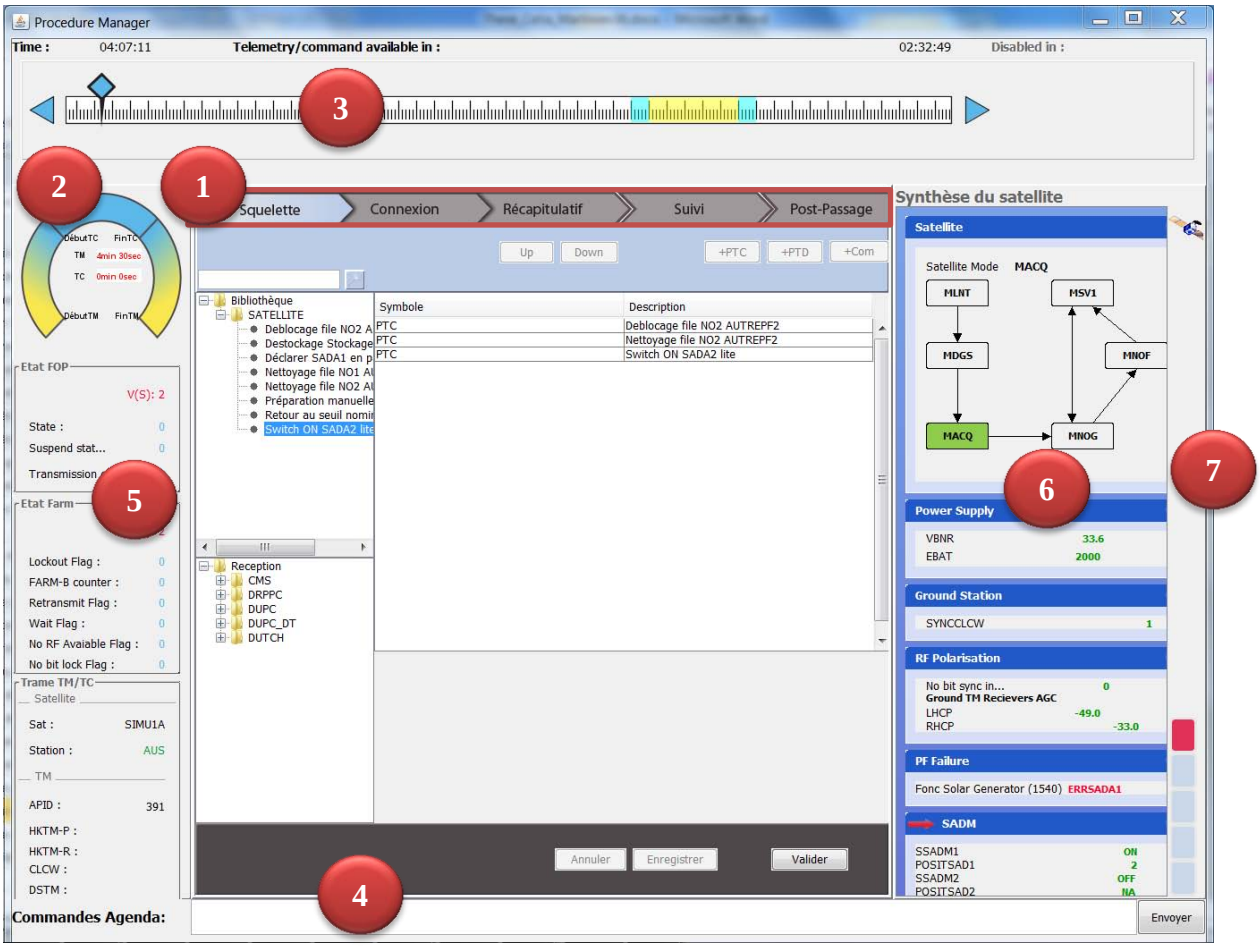


Figure 126. Copie d'écran du prototype haute-fidélité de l'interface de l'application segment sol de contrôle de la mission PLEIADES

La Figure 126 présente l'interface du prototype très haute-fidélité de l'application segment sol. Par rapport à la version actuelle de l'application (comme présentée dans la Figure 115), cette interface contient de nouveaux éléments graphiques permettant de satisfaire les besoins spécifiés par les opérateurs (listés dans la sous-section 2.1). Les lettres par lesquelles ces éléments sont référencés sont les lettres des besoins correspondants dans la sous-section 2.1 :

a) La barre de progression indiquant la séquence de tâches pour préparer un plan de télécommandes, gérer son suivi pendant le passage et effectuer un traitement post-passage « Squelette – Connexion – Récapitulatif – Suivi – Post-passage » (pastille 1) répond au besoin : « L'interface devrait guider l'opérateur par rapport aux tâches qu'il doit effectuer avant et pendant l'intervalle de temps ou le satellite est en visibilité ».

b) La jauge circulaire à gauche de l'interface (pastille 2) répond au besoin « L'interface devrait rendre plus explicite les sous-périodes de visibilité ou la réception de télémesure est possible et où l'envoi de télécommandes est possible ».

c) Le chronographe dans la partie supérieure de l'interface (pastille 3) répond au besoin « L'interface devrait fournir un moyen graphique symbolique pour la visualisation du temps de visibilité restant ».

d) La zone de saisie dans la partie inférieure de l'interface (pastille 4) répond au besoin « L'interface devrait permettre de rechercher une commande, une fonction de l'application ou un paramètre par mot-clé ».

e) La zone située à gauche (pastille 5) contient les paramètres de la station sol utilisée pour la communication et la zone située à droite de l'interface (pastille 6) contient les paramètres vitaux

du satellite. Ces zones répondent au besoin « L'interface devrait permettre de regrouper les paramètres vitaux des différentes parties du système par zones ».

La zone verticale située complètement sur la droite (pastille 7) permet de représenter dynamiquement l'arrivée de la télémesure depuis le satellite. Un paquet de couleur rouge indique qu'une anomalie ou panne est décrite dans ce paquet.

La Figure 127 montre l'environnement logiciel d'exécution du prototype très haute-fidélité de l'application segment sol et les modèles (dans la partie centrale, chaque onglet correspond à un modèle en cours d'exécution) contrôlant son exécution.

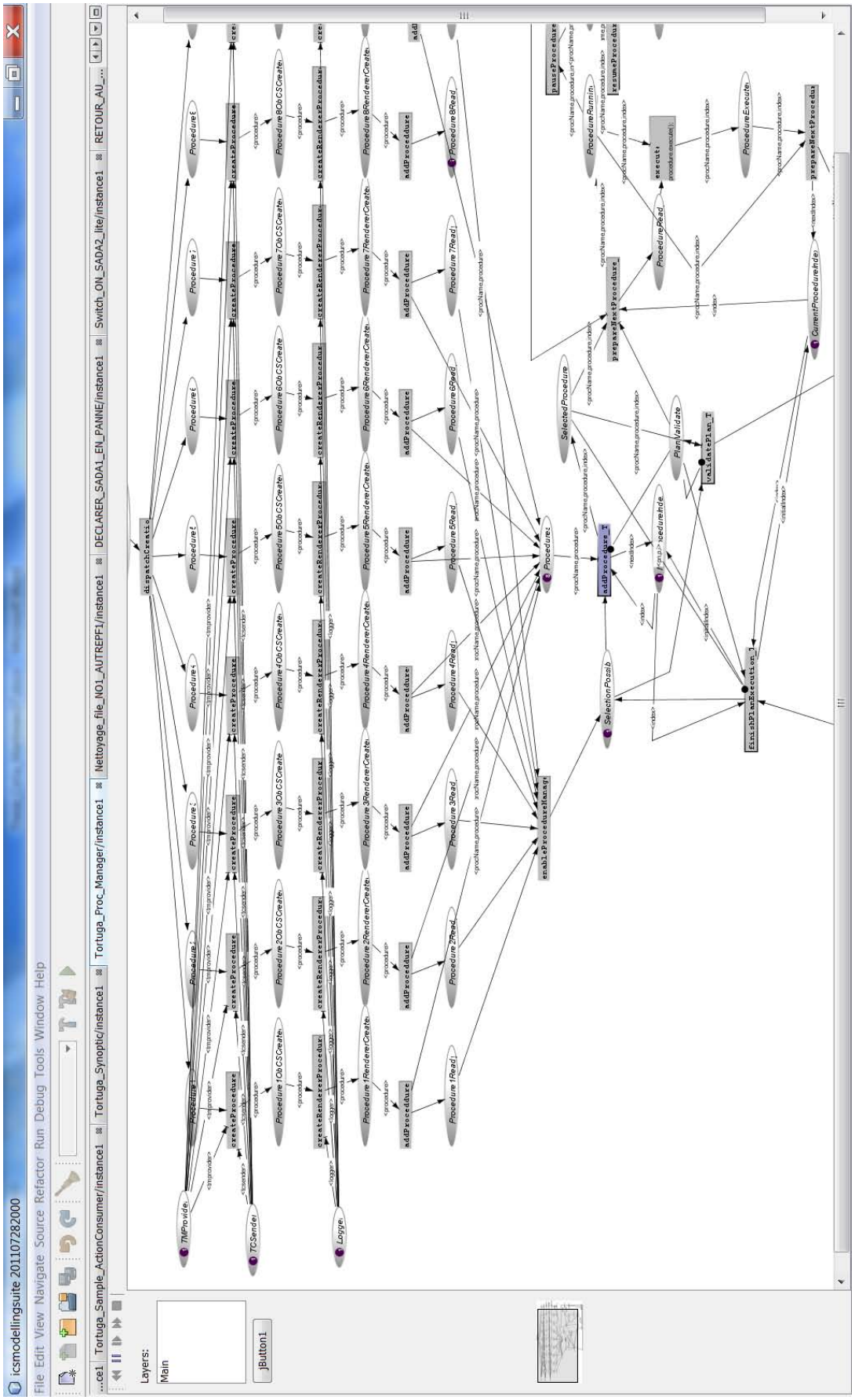


Figure 127. Copie d'écran de l'environnement des modèles formels contrôlant l'exécution du prototype de l'application segment sol

La Figure 128 présente le modèle de comportement de la partie automatisée de la procédure de changement de moteur d'entraînement des panneaux solaires, utilisée pour résoudre une panne SADA. Elle peut être appelée par le modèle ICO de comportement du gestionnaire de procédures grâce au port d'entrée « SIP_execute » et son appel est bloquant, le gestionnaire de procédure doit attendre qu'un jeton soir présent dans le port de sortie « SOP_execute ». Entre autres, elle spécifie l'exécution d'envoi de télécommandes vers le satellite pour. Les étapes principales sont :

- La vérification que le moteur d'entraînement SADA2 est actif (représenté par la transition « checkSSADM2_isON » de la pastille 1).
- L'attente de la confirmation ou de l'infirmation de l'arrêt de la rotation du SADA2 (représentée par les transitions sous la pastille 2), suite à une opération de rotation pour rallier sa position initiale de démarrage. Si l'opérateur infirme l'arrêt de la rotation, la procédure est stoppée, sinon les étapes suivantes sont exécutées. Ces deux transitions sont liées à des fonctions d'activation afin que l'évènement utilisateur de sélection d'un bouton sur l'interface présentée en Figure 130 déclenche l'une des deux transitions.
- La mise en fonctionnement du SADA2 (pastille 3) avec un envoi de télécommande « ONBARRE ».
- La demande de confirmation de bonne réception de paquets de télémesure suite à cette opération de changement (pastille 4). Cette transition est liée à une fonction d'activation afin que l'évènement utilisateur de sélection du bouton sur l'interface présentée en Figure 131 déclenche cette transition.

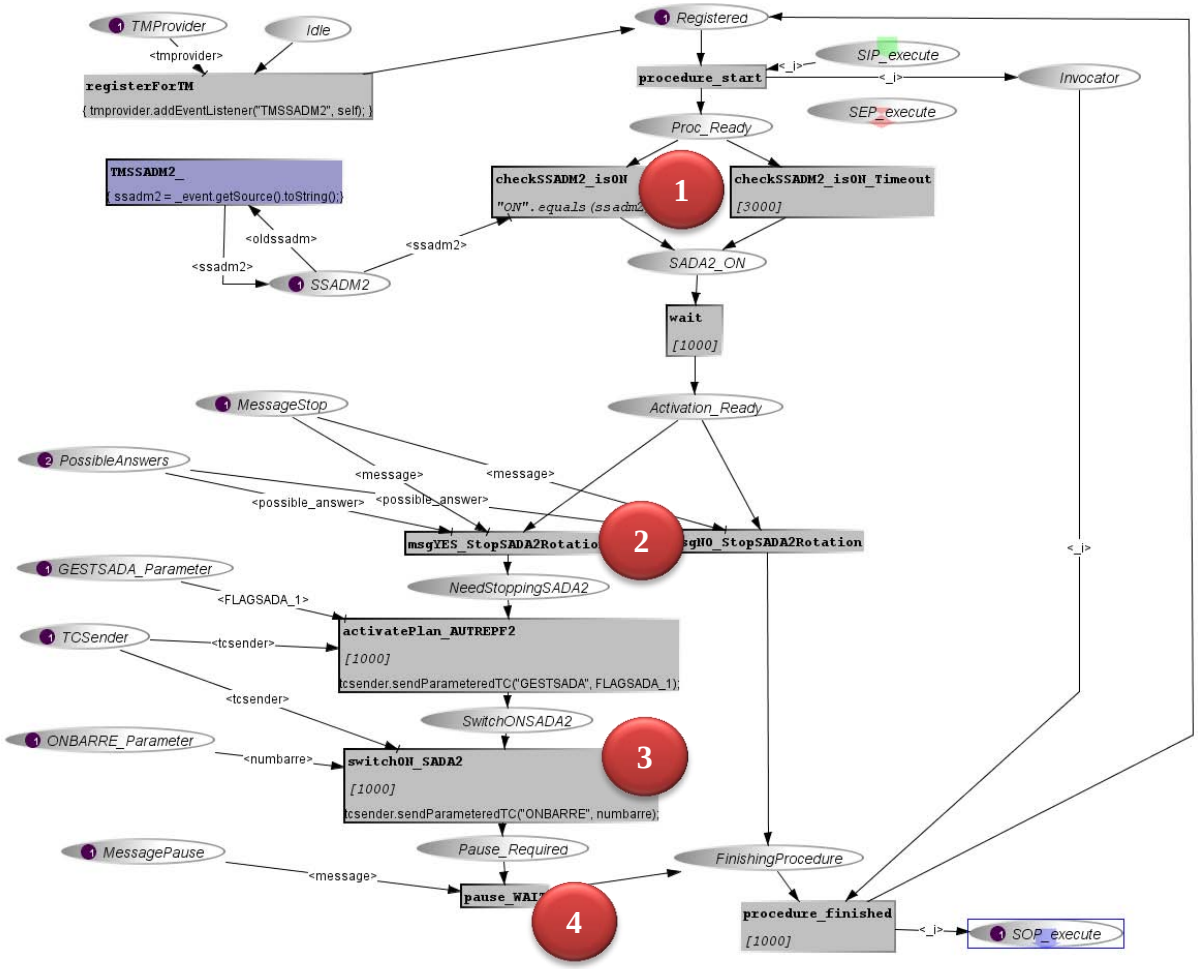


Figure 128. Modèle ICO de la procédure d'envoi de télécommandes pour le changement de moteur d'entraînement des panneaux solaires

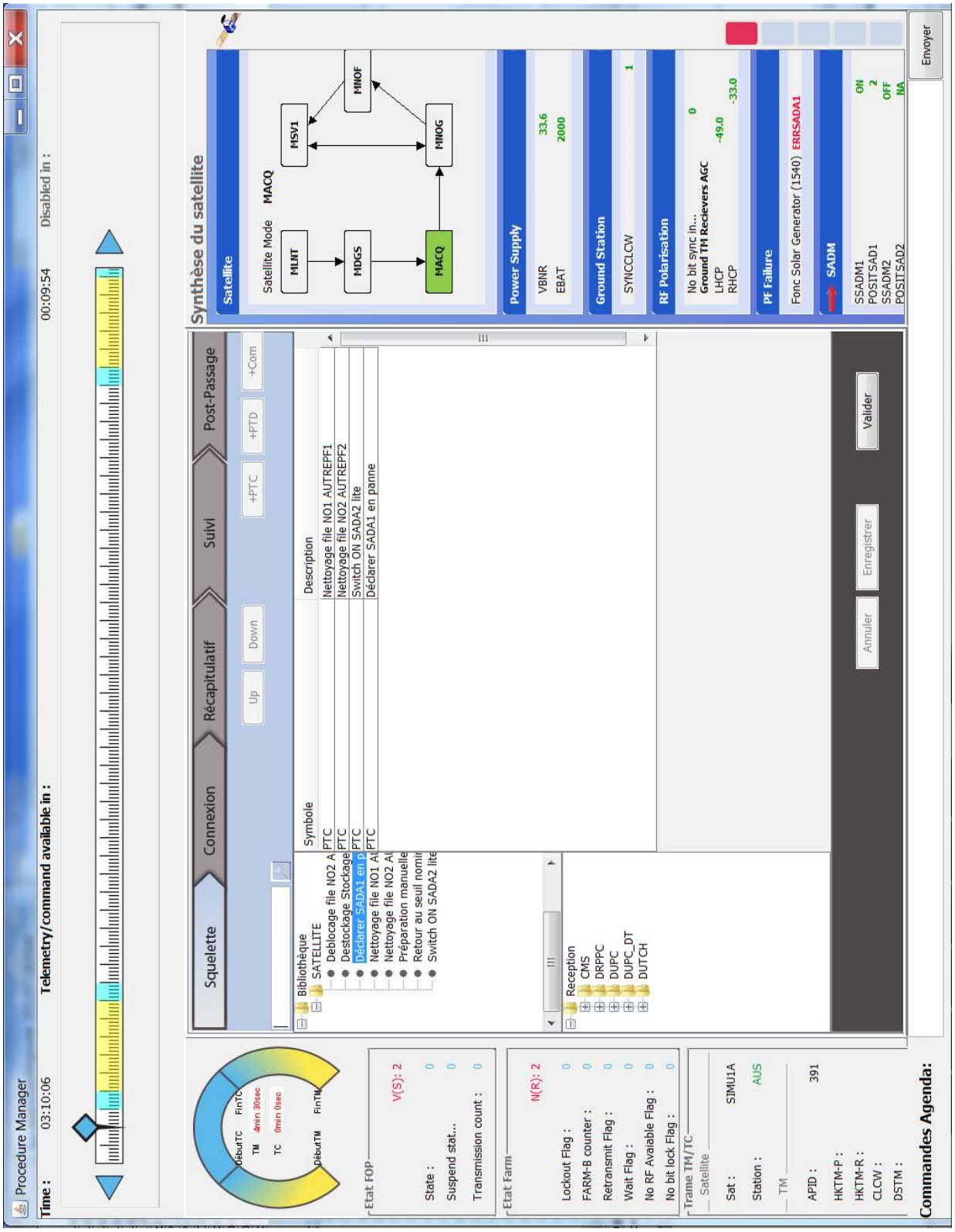


Figure 129. Copie d'écran de l'interface permettant de contrôler le déroulement de la partie automatisée des procédures (plan de télécommandes)

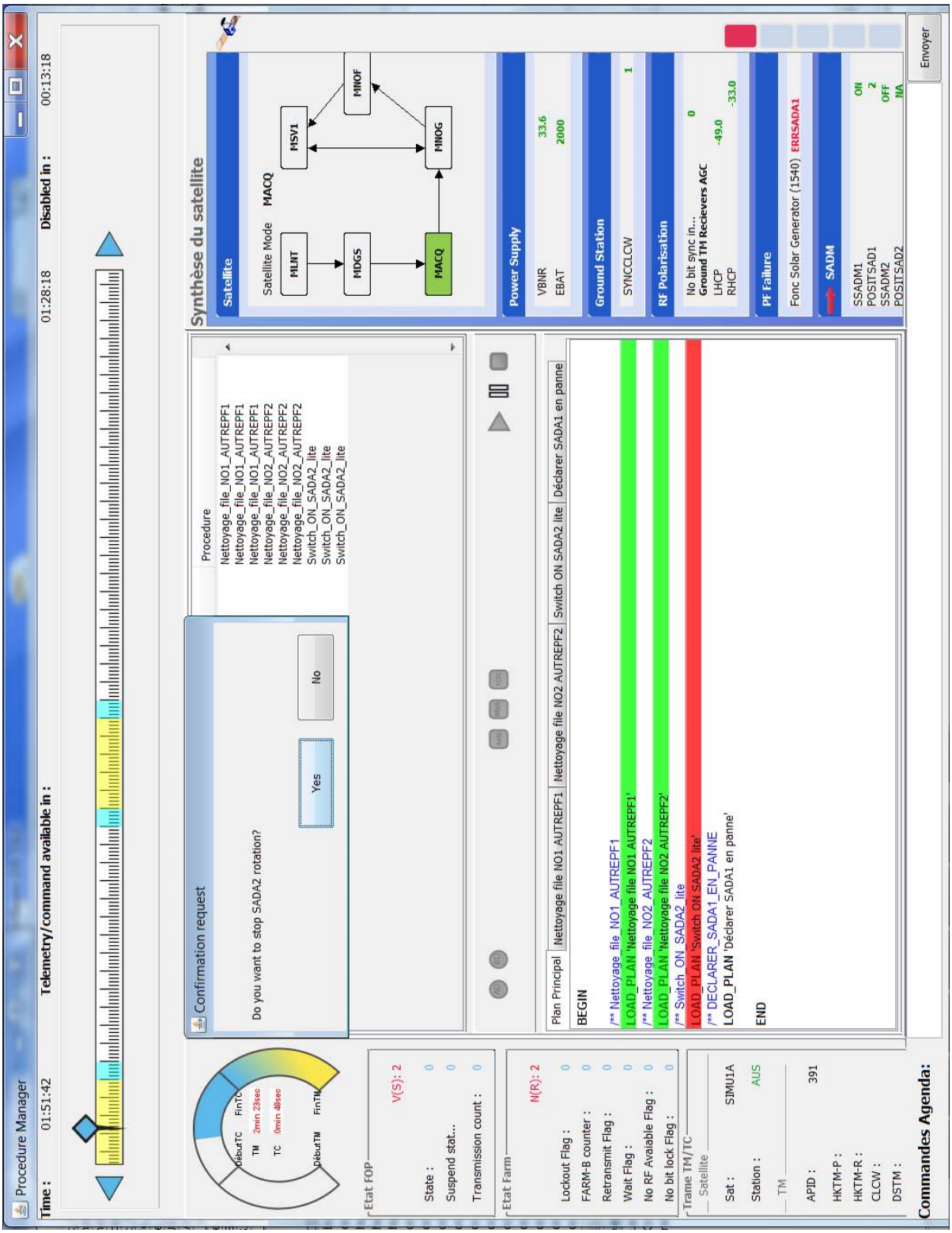


Figure 130. Copie d'écran de l'interface permettant de contrôler la partie automatisée des procédures (demande de confirmation d'arrêt de la rotation du SADA redondant)

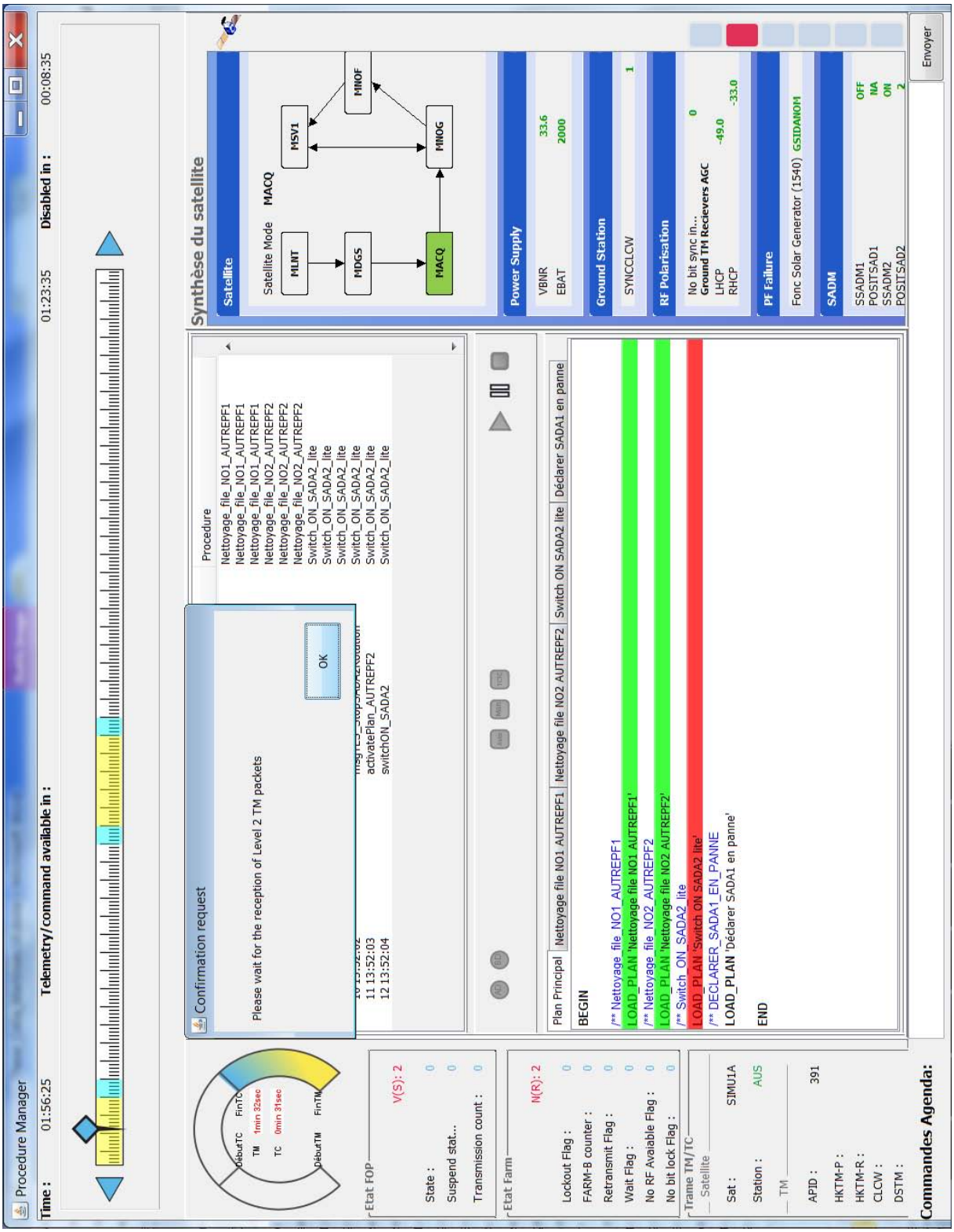


Figure 131. Copie d'écran de l'interface permettant de contrôler la partie automatisée des procédures (demande de confirmation de bonne réception de la télémesure)

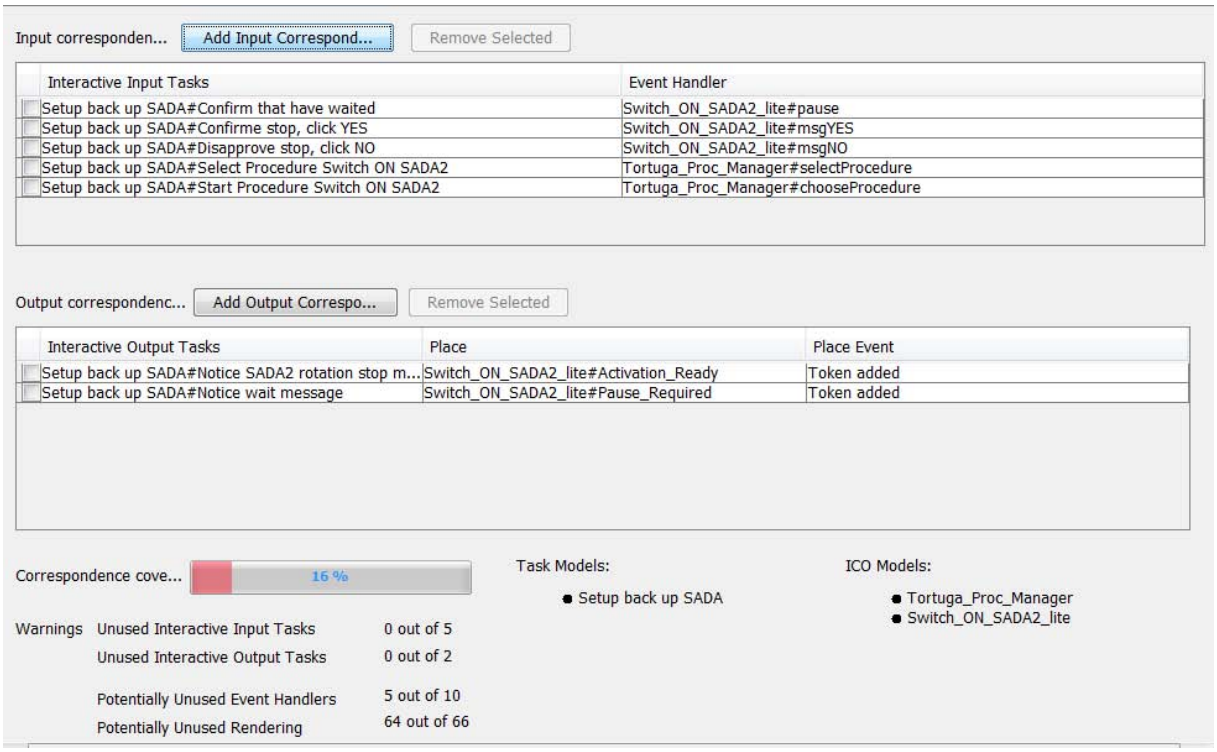


Figure 132. Copie d'écran de l'interface de mise en correspondance entre les modèles de comportement de l'application et des procédures et les modèles de tâches

La Figure 132 présente la mise en correspondance entre :

- Les tâches interactives d'entrée du modèle de tâches de changement de SADA « Setup back up SADA » et les fonctions d'activations du modèle de comportement de l'application segment sol « Tortuga_Proc_Manager » et du modèle de la procédure automatisée « Switch_ON_SADA2_lite ».
- Les tâches interactives de sortie du modèle de tâches de changement de SADA « Setup back up SADA » et les fonctions de rendu du modèle de comportement de l'application segment sol « Tortuga_Proc_Manager » et du modèle de la procédure automatisée « Switch_ON_SADA2_lite ».

Cette mise en correspondance permet la synchronisation entre les modèles de tâches et les modèles du système lors de la co-exécution des modèles.

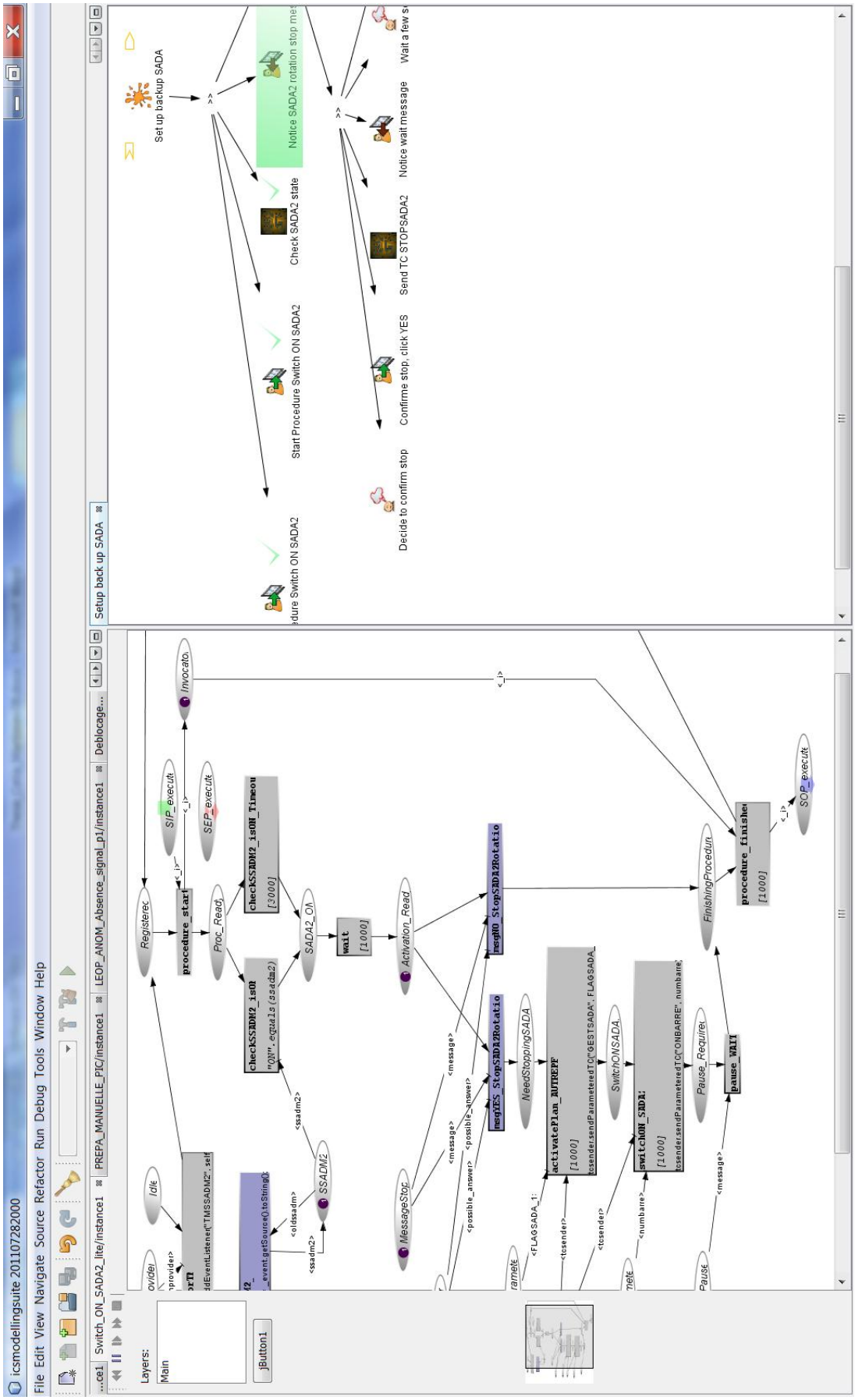


Figure 133. Copie d'écran de la co-exécution du modèle formel de la partie automatisée de la procédure de changement de SADA et du modèle de tâche correspondant

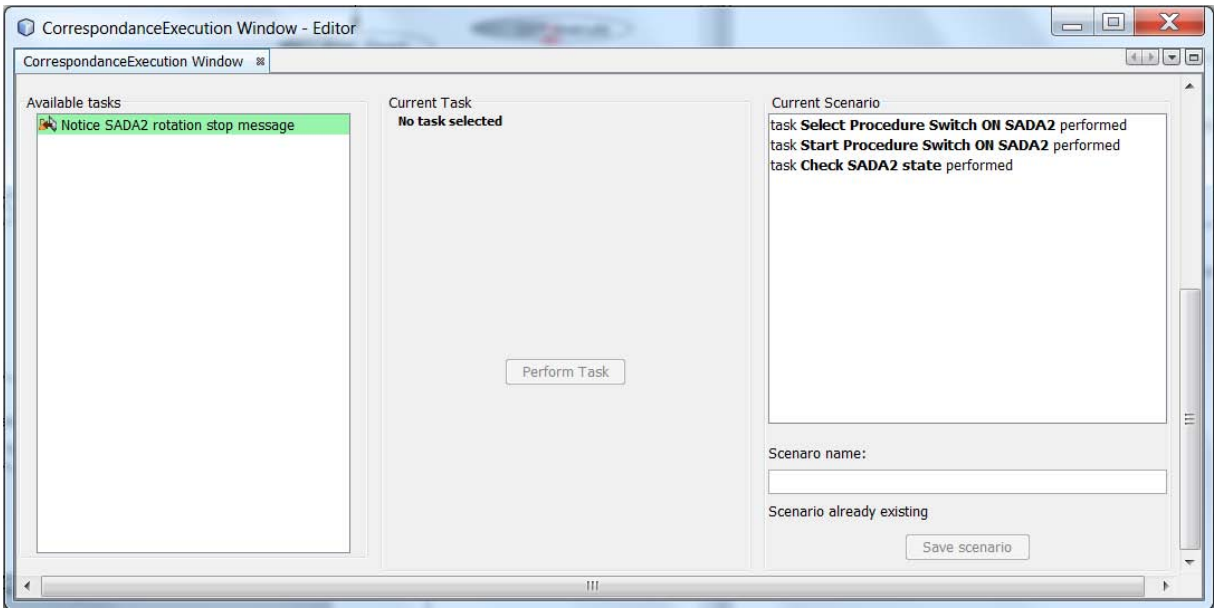


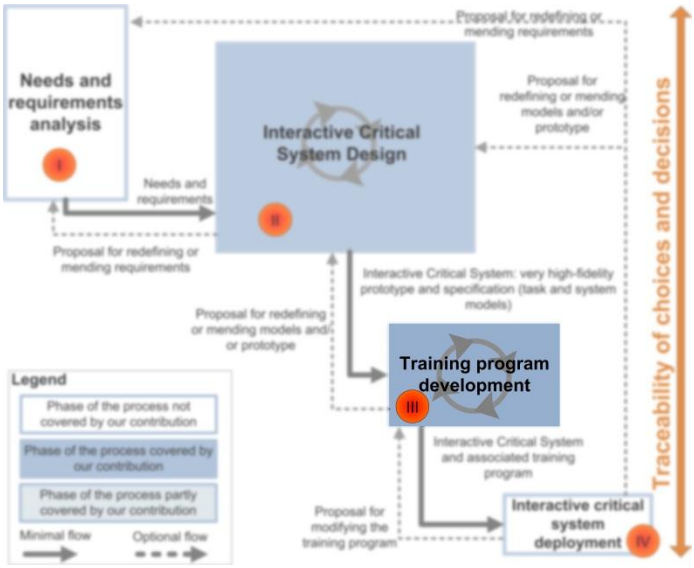
Figure 134. Copie d'écran de l'interface de contrôle de la co-exécution pendant le suivi de la procédure de basculement du SADA

Les Figure 133 et Figure 134 présentent des copies d'écran de l'interface de contrôle de la co-exécution sous-jacente des modèles de tâches et des modèles formels du comportement du système. Cette co-exécution se déroule simultanément avec l'exécution de l'interface du prototype d'application segment sol et permet de valider l'adéquation entre les différents types de modèles mais aussi l'adéquation entre les besoins spécifiés dans les phases amont, les propriétés requises et une version fonctionnelle de l'application.

2.2.4 Analyse quantitative des performances utilisateur

La phase d'analyse quantitative des performances utilisateur ne fait pas l'objet d'une description particulière dans ce chapitre car elle est décrite de manière détaillée dans la section 3.4 du chapitre 6 et les éléments de cette étude de cas n'ajouteraient pas de précisions supplémentaires par rapport à l'approche.

2.3 Phase de développement du programme de formation associé



La phase de développement du programme de formation s'appuie sur :

- L'environnement de conception synergique mis en œuvre dans la phase précédente.
- Le prototype très haute-fidélité.
- Les modèles de tâches et de comportement du système.
- Les outils logiciels pour la préparation, le déroulement et l'analyse des sessions de formation.

La mise en œuvre détaillée du programme de formation grâce aux notations, techniques et outils proposés dans cette

thèse est décrite dans la section 2.3 du chapitre 6. Dans cette section, nous montrons l'application de cette mise en œuvre à la préparation, au déroulement et à l'analyse d'une session de formation à la résolution d'une panne SADA.

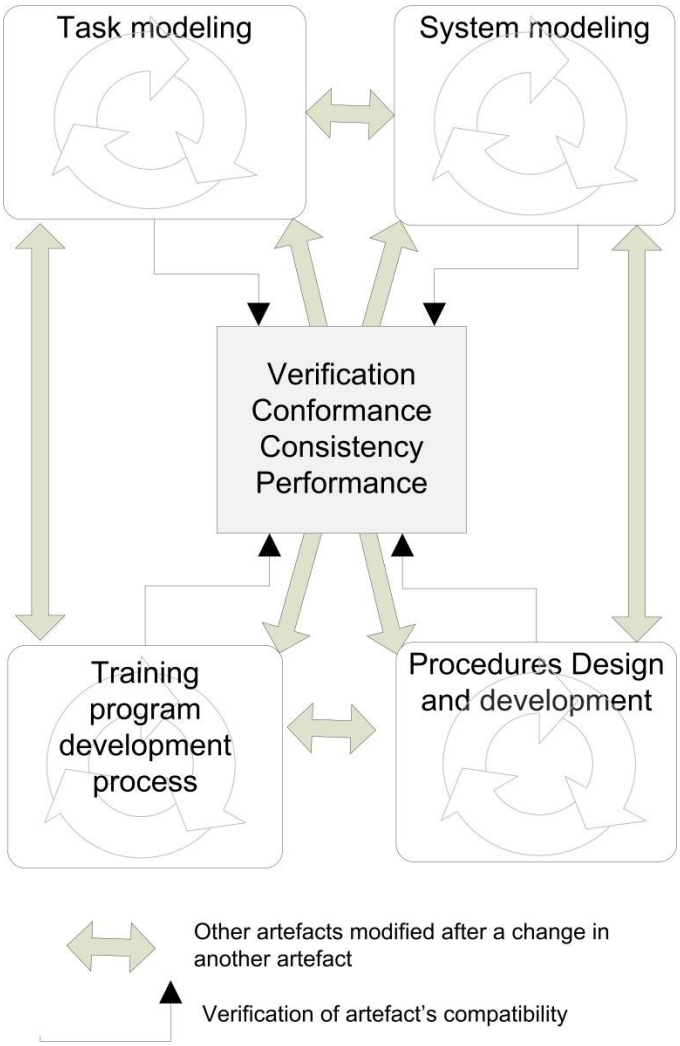


Figure 135. Phase de vérification de la conformité et de l'adéquation entre les modèles de tâches, les modèles du système, les procédures et le programme de formation

La Figure 135 illustre la manière dont notre approche fournit un support à la vérification de la conformité entre les modèles de tâches des activités des opérateurs, les modèles formels du comportement de l'application segment sol, les procédures opérationnelles de gestion de la mission (comme celle de résolution d'une panne SADA) et le programme de formation des opérateurs.

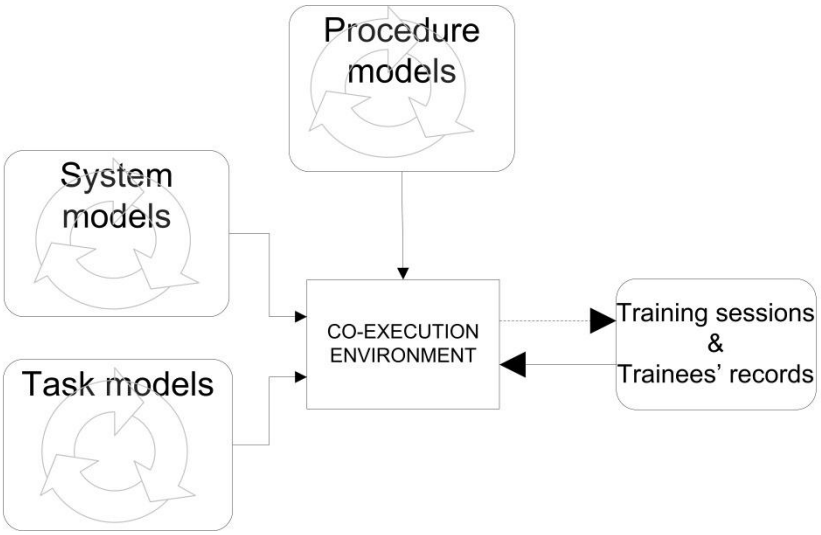


Figure 136. Environnement synergique de co-exécution des modèles comme support à la formation

La Figure 136 illustre la manière dont l'environnement synergique de co-exécution, alimenté par les différents modèles de tâches, du système et des parties automatisées des procédures, fournit un support aux sessions de formation ainsi qu'à l'analyse de performances des opérateurs formés suite aux sessions de formation. Nos propositions basées sur l'environnement synergique de modélisation, le prototype très haute-fidélité pour fournir un support à la formation des opérateurs de systèmes interactifs critiques a fait l'objet d'une publication et est détaillée dans (Martinie, Palanque, Navarre, Winckler, & Poupart, 2011). Elles permettent de réaliser :

- La préparation des sessions de formation avec différents scénarios d'utilisation générés grâce à l'environnement synergique PetShop-HAMSTERS et validés en fonction des scénarios d'utilisation recueillis dans les phases d'analyse des besoins et des tâches, en amont du processus.
- Les sessions de formation avec le prototype haute-fidélité, les scénarios d'utilisation et les modèles de tâches décrivant les activités à savoir mener.
- L'évaluation des performances opérateurs suite à une ou plusieurs sessions de formation.

Pour illustrer ces étapes, nous décrivons l'environnement de mise en œuvre du plan de formation pour les opérateurs du segment sol de surveillance et contrôle du satellite PLEIADES.

Dans la suite de cette section, nous décrivons un exemple de conception d'une session de formation concernant la gestion d'une panne d'un moteur d'entraînement d'un panneau solaire (aussi appelé SADA – Solar Array Driver Assembly). La Figure 138 montre une copie d'écran de l'environnement logiciel intégré. La partie gauche de la copie d'écran correspond au navigateur de programmes de formation.

Chaque sous-section suivante présente un échantillon de ces éléments produits pour le cas d'étude de la résolution d'une panne de moteur d'entraînement des panneaux solaires (panne SADA).

2.3.1 Préparation d'une session

La Figure 137 présente la préparation de scénarios d'utilisation de l'application segment sol pour résoudre une panne SADA. L'outil de simulation HAMSTERS y est principalement mis en œuvre et permet de simuler l'ensemble des tâches décrites dans les modèles et sous-modèles de tâches présentés dans la section 2.2.1 de ce chapitre. La partie droite de la Figure 137 montre la fenêtre de contrôle de la simulation des modèles. Comme décrite dans la section 6 du chapitre 4, elle contient la liste des

tâches disponibles à l'exécution (liste de la partie supérieure) ainsi que la liste des tâches déjà exécutées (liste de la partie inférieure). En plus des tâches disponibles et des tâches effectuées, ces listes contiennent, entre parenthèses après chaque tâche, le nom du modèle desquelles elles sont issues.

Le modèle de tâche associé à l'activité de résolution d'une panne SADA est exposé dans la partie droite de la Figure 138 : « Set up back up SADA ». Il décrit les activités de l'utilisateur pour rendre fonctionnel le SADA redondant (afin de remplacer le SADA défaillant) :

- La sélection de la procédure automatisée à exécuter par le segment sol (tâche interactive « Select procédure Switch ON SADA2 »).
- Le lancement de l'exécution de cette procédure (tâche interactive « Start procedure Switch ON SADA2 »).
- La vérification par le système que le moteur d'entraînement SADA2 est actif (fonction système « Check SADA2 »).
- L'attente du système d'une confirmation de l'opérateur pour arrêter de la rotation du SADA2 (tâche interactive de sortie « Notice SADA2 rotation stop message »).

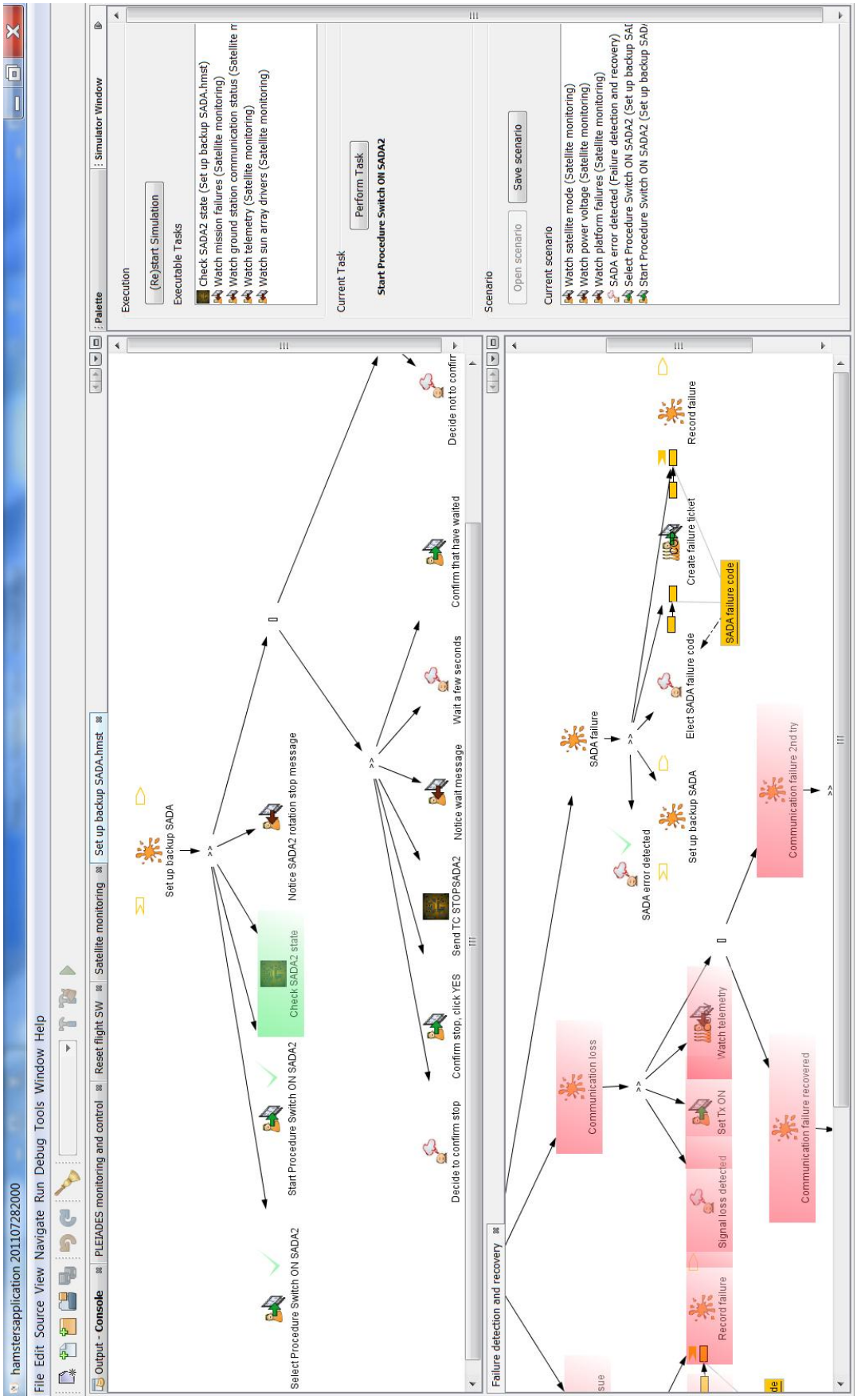


Figure 137. Préparation d'un scénario d'utilisation pour la résolution d'une panne SADA

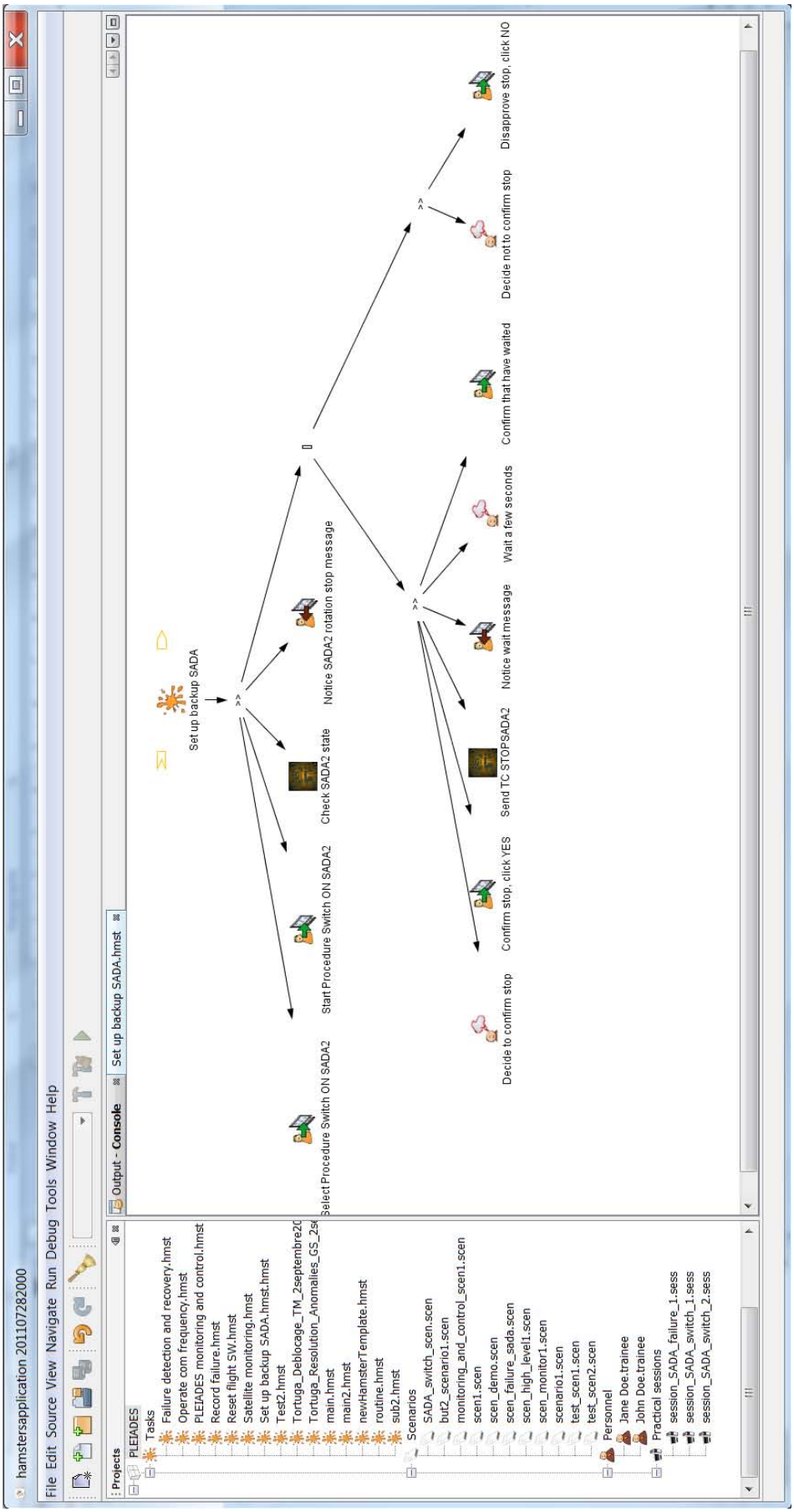


Figure 138. Copie d'écran du navigateur de projets de sessions de formation dans l'environnement logiciel intégré d'exploitation synergique

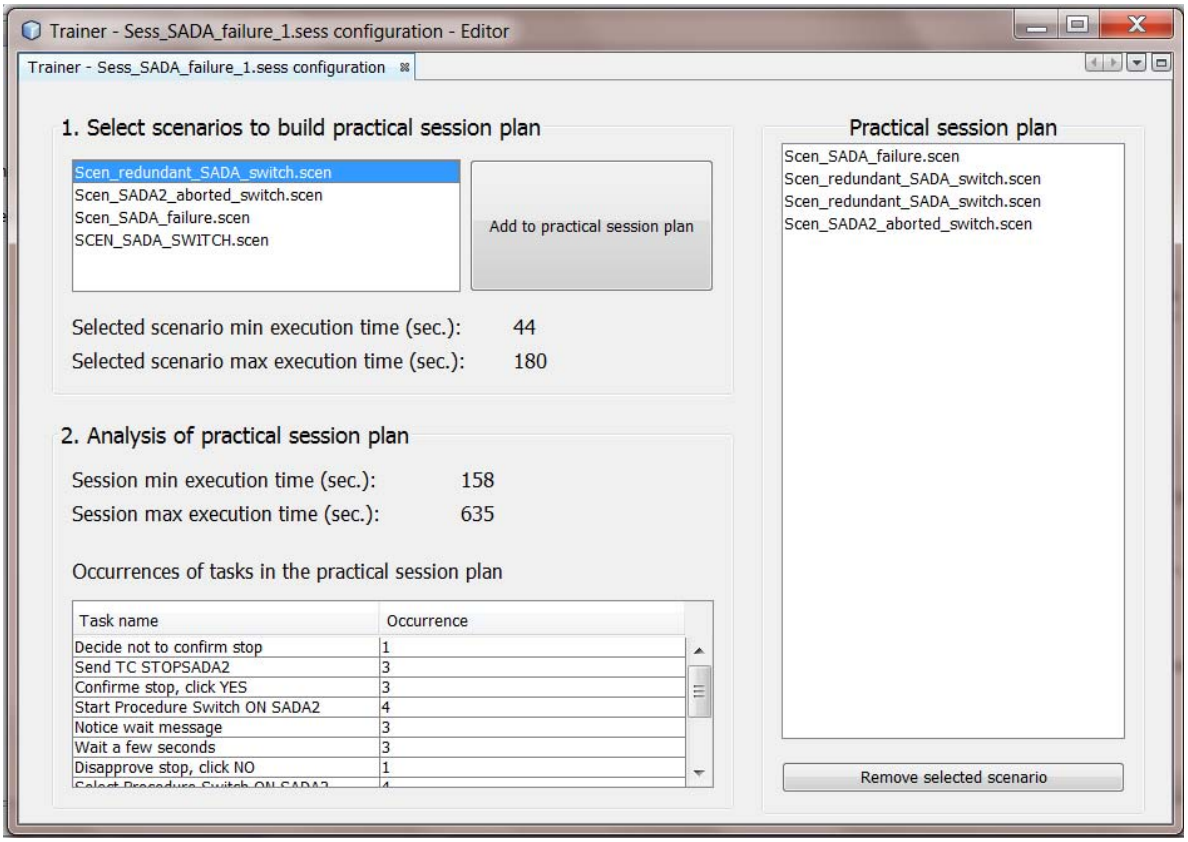


Figure 139. Copie d'écran de l'interface d'édition d'une session de formation par un formateur

La Figure 139 présente une copie d'écran de l'interface de préparation d'une session de formation à la résolution d'une panne SADA. L'interface est décomposée en trois parties (détaillée dans la section 2.3 du chapitre 6) :

- La zone de sélection des scénarios (« 1. Select scenarios to build practical session plan »).
- La zone d'analyse du contenu de session pour déterminer la couverture des tâches par la version actuelle du plan (« 2. Analysis of practical session plan»).
- La liste des scénarios à effectuer pendant la session ou plan de session (partie droite de l'interface, « Practical session plan »).

2.3.2 Déroulement d'une session

Le déroulement d'une session de formation s'effectue avec les outils logiciels suivants :

- Le simulateur de satellite (présenté sur la Figure 140)
- Le prototype très haute-fidélité conçu dans la phase précédente du processus de développement (présenté sur la Figure 141) et les modèles de tâches et du comportement du système s'exécutant simultanément.
- L'interface d'exécution d'une session de formation (présentée sur la Figure 142) où l'opérateur consulte les scénarios qu'il/elle a à effectuer lors de cette session et les met en pratique en s'aidant du simulateur et du prototype très haute-fidélité.

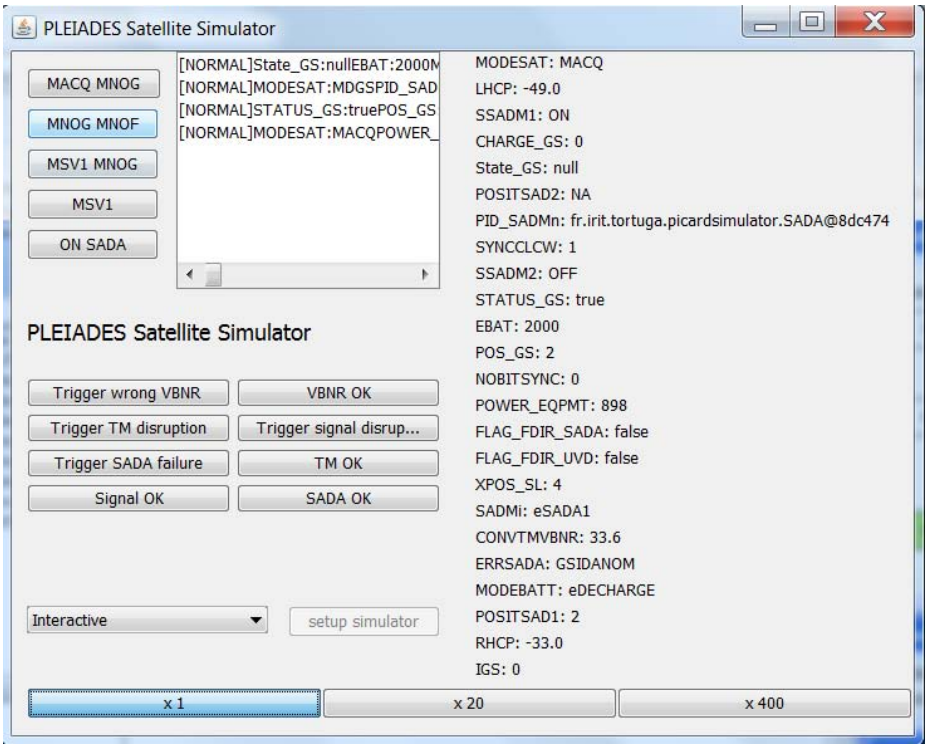


Figure 140. Copie d'écran de l'interface de contrôle du simulateur de satellite utilisé pour la session de formation

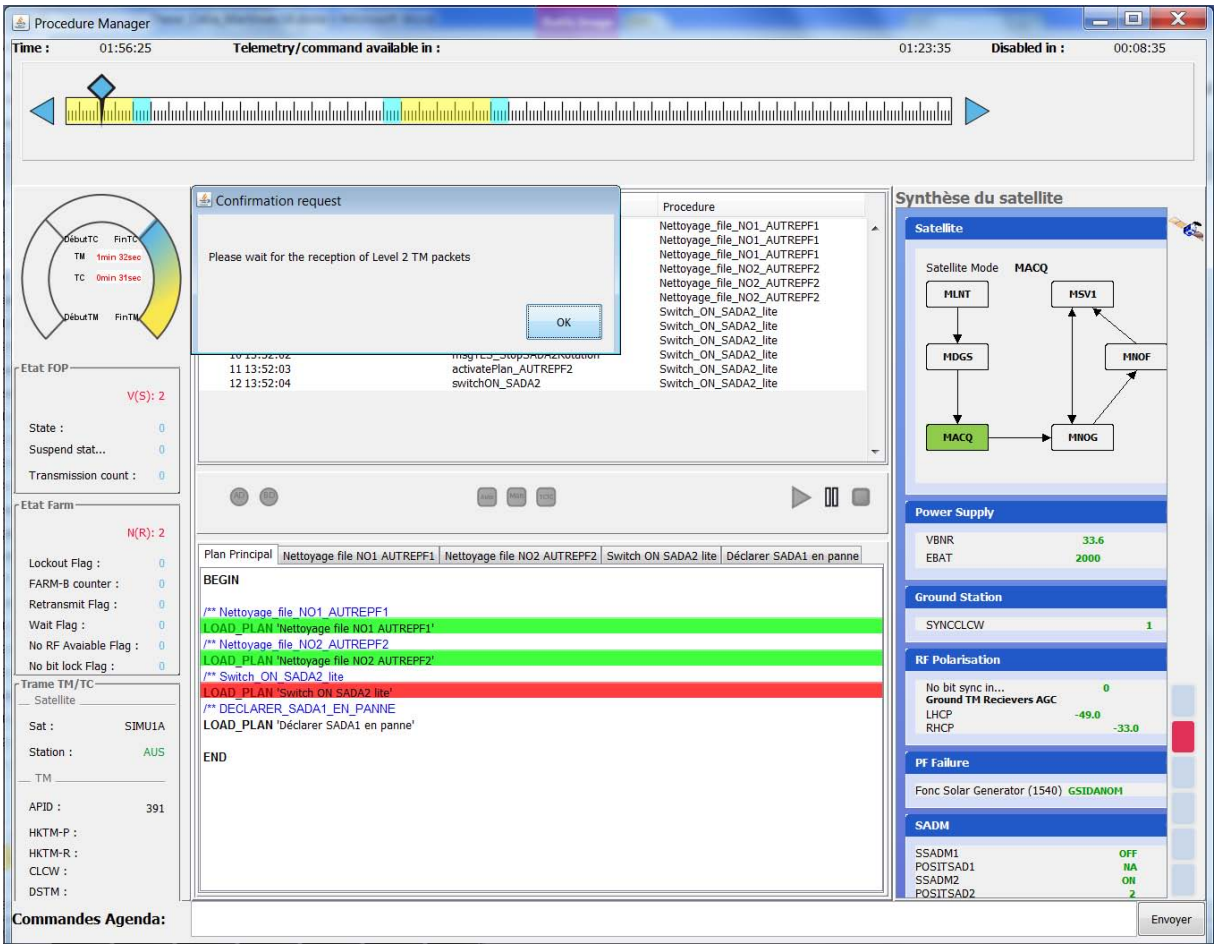


Figure 141. Copie d'écran du prototype de l'application segment sol utilisé pour la session de formation

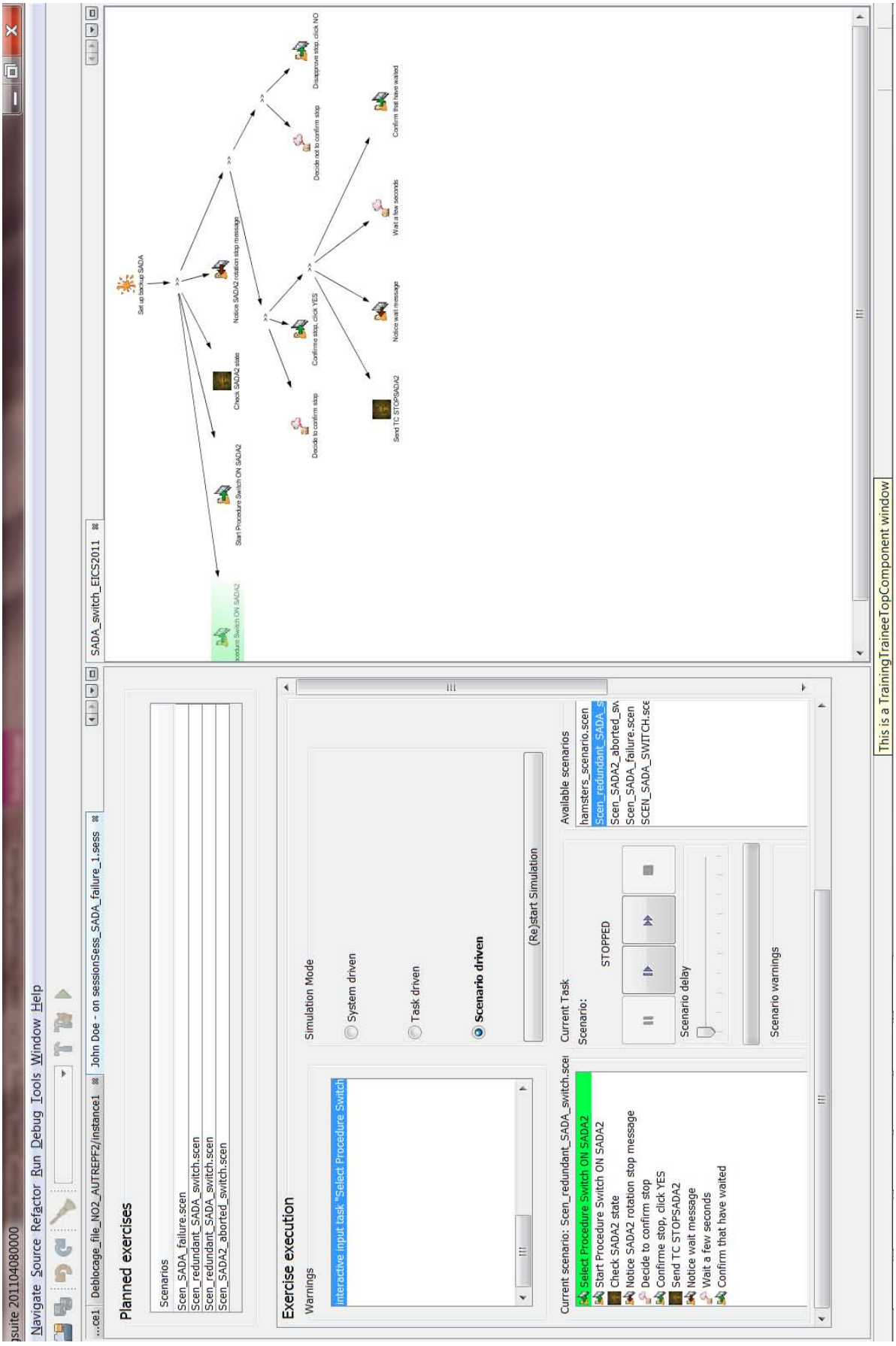


Figure 142. Copie d'écran de l'interface d'exécution d'un plan de session de formation

2.3.3 Analyse des performances post-session

La Figure 143 présente l'interface d'analyse des résultats des sessions effectuées par l'opérateur « John Doe ». Cette interface d'analyse peut être utilisée par l'opérateur lui-même pour se préparer aux qualifications et elle peut aussi être utilisée par le formateur. Cette interface (détaillée dans la section 2.3 du chapitre 6) permet de mettre en valeur que lors de la première session effectuée par « John Doe » (surlignée en bleu dans la liste des sessions effectuées répertoriées par leur nom et la date de leur déroulement), les trois premiers scénarios ont été effectués avec succès contrairement au quatrième. De plus, nous pouvons constater que globalement pour toutes les sessions, John Doe a pratiqué plusieurs fois les scénarios « Scen_SADA_failure.scen » et « Scen_redundant_SADA_switch.scen » mais n'a pas pratiqué du tout le scénario « Scen_SADA2_aborted_switch.scen ». Le tableau de la partie inférieure permet à l'opérateur et/ou au formateur de connaître les temps d'exécution pour chaque scénario pratiqué.

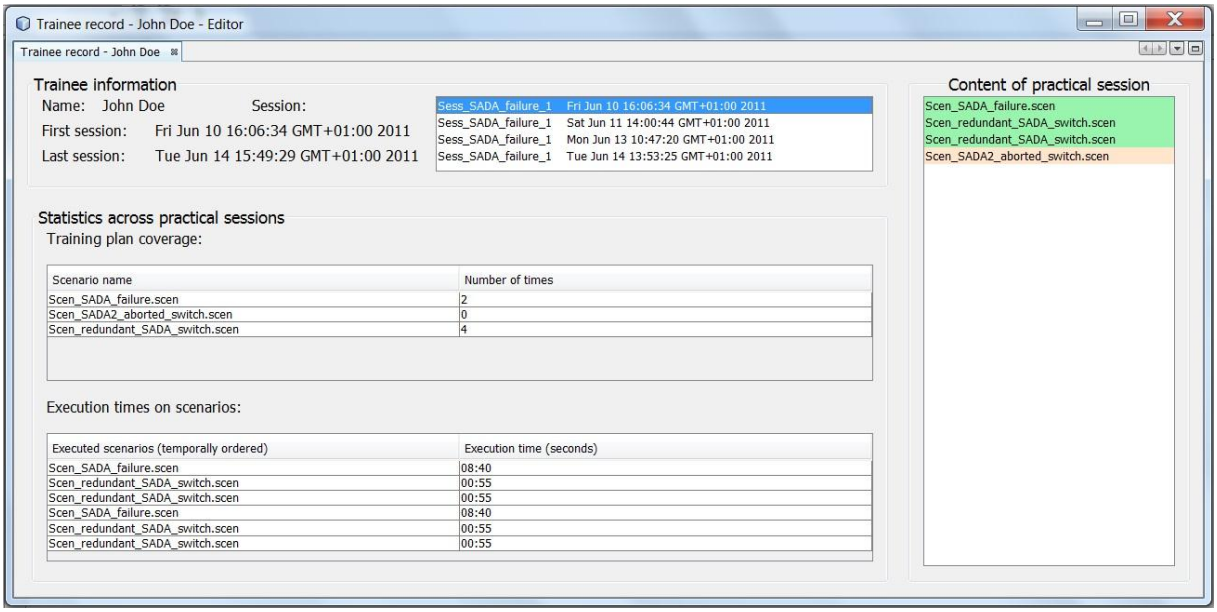


Figure 143. Copie d'écran de l'interface d'évaluation des résultats de l'exécution des sessions de formation pour un membre du personnel formé

2.4 Traçabilité des choix de conception

La traçabilité des choix de conception et exigences tout au long du processus de développement de l'application segment sol n'est pas décrite dans cette étude de cas. En effet, cette phase transversale à l'approche a fait l'objet d'une présentation détaillée dans le chapitre précédent.

3 Conclusion

L'étude de cas présentée dans ce chapitre permet de montrer la faisabilité de l'approche proposée par cette thèse pour la conception et le développement d'une application interactive complexe et critique :

- Nous avons montré que les applications segment sol et de démontrer leur nature interactive, complexe et critique.
- Nous avons appliqué le processus de développement proposé par cette thèse (décrit au chapitre 5) pour concevoir une application segment sol de supervision de la mission satellitaire PLEIADES et son programme de formation associé. Les techniques de modélisation synergiques décrites au chapitre 6 nous ont permis de mettre en œuvre le développement et la validation du prototype très haute-fidélité et des modèles de tâches et du système. Ce prototype et l'ensemble de ces modèles,

avec les modules logiciels pour la formation (décrits dans la section 4 su chapitre 6) servent ensuite de support au développement du programme de formation associé à cette application segment sol.

Conclusion

Dans le domaine de l'IHM, une grande majorité des travaux s'intéresse à investiguer des techniques d'interaction innovantes et à proposer des méthodes et procédés pour les concevoir et les évaluer (Blanch & Ortega, 2011) (Wherry, 2003) (Apitz, Guimbretière, & Zhai, 2008) (Höök, et al., 2011). Ces travaux sont souvent centrés sur l'utilisabilité des systèmes conçus et sur les performances d'utilisation de ces systèmes. Les études de cas réalisées pour démontrer la faisabilité de ces techniques, méthodes et procédés portent en général sur des tâches simples. Ainsi, les solutions proposées parviennent difficilement à répondre aux exigences intermédiaires permettant leur industrialisation telles que le passage à l'échelle, la fiabilité, l'applicabilité, l'exploitabilité et la tolérance aux erreurs.

Dans le domaine des systèmes interactifs critiques, la prise en compte de ces exigences pour l'industrialisation est nécessaire car le coût d'une erreur, engendrée par un utilisateur ou par le système utilisé, peut largement dépasser le coût de développement du système. Le critère de fiabilité est le critère prépondérant pour développer ce type de systèmes. En conséquence, le développement de systèmes critiques est, la plupart du temps, régit par des standards et recommandations pour assurer leur fiabilité lors de leur mise en exploitation. Le développement de ce type de systèmes est soumis à différentes contraintes telles que la description complète et non ambigüe des fonctionnalités et de leur mise en œuvre (European Organisation for Civil Aviation Equipment, 1992), ainsi que la traçabilité des exigences fonctionnelles par rapport à la description des fonctionnalités et de leur mise en œuvre (EUROCONTROL, 2003). De plus, les utilisateurs de ces systèmes sont formés et qualifiés avant de pouvoir les exploiter dans le cadre de leurs missions. En effet, avec le nombre croissant des fonctionnalités de certains de ces systèmes et de la complexité des tâches et interactions pour les utiliser, il est désormais difficile de les concevoir sans prendre en compte, au même titre que la propriété de fiabilité, les propriétés d'utilisabilité et d'opérabilité de ces systèmes.

Les contributions apportées par cette thèse concernent les moyens de concevoir des systèmes interactifs critiques à la fois utilisables, fiables et opérables. Nos contributions se situent sur deux plans :

- Sur le plan théorique, nous proposons une approche de développement intégrée pour un système interactif complexe et/ou critique afin qu'il satisfasse simultanément les propriétés d'utilisabilité, de fiabilité et d'opérabilité. Pour mettre en œuvre cette approche, nous proposons un processus de conception et développement du système et de son programme de formation associé.
- Sur le plan pratique, nous proposons des moyens pour mettre en œuvre cette approche et ce processus :
    - Une notation et un outil logiciel pour la modélisation des tâches en support à :
        - L'analyse des tâches utilisateurs complexes et/ou critiques.
        - La validation et vérification de l'adéquation entre les tâches utilisateur et les fonctions du système.
        - L'analyse des besoins en formation, la conception et le développement d'un programme de formation ainsi qu'un support à l'évaluation des utilisateurs formés.
    - Un environnement logiciel de prototypage très haute-fidélité et modélisation synergique (tâches utilisateur et du comportement du système) en support à :
        - La validation et la vérification de l'adéquation entre les tâches utilisateur et les fonctions du système.

        - Des moyens de formation assistée par ordinateur (exercices pratiques d'entraînement, connexion à un simulateur,…).
    - Un procédé pour l'évaluation des performances temporelles de l'utilisateur.
    - Une démarche de traçabilité des exigences tout au long du processus de développement soutenue par une notation et un outil logiciel (TEAM et DREAMER).

Les **chapitres 1 et 2** nous ont permis de mettre en valeur la complexité et les contraintes de conception et d'exploitation des systèmes interactifs critiques et/ou complexes. Ces caractéristiques nous ont permis de comprendre les besoins particuliers pour la conception et le développement d'un système simultanément utilisable, fiable et opérable. Notamment, nous avons montré qu'il était nécessaire de recourir à des techniques outillées de description formelles du comportement du système, et qu'il était nécessaire de recourir à des processus et techniques fournissant un support à la description de tâches nombreuses et complexes. Plus particulièrement, le chapitre 2 nous a permis de montrer les avantages et limites des approches existantes et la nécessité d'une nouvelle approche intégrant le développement du programme de formation.

Le **chapitre 3** nous a permis de mettre en valeur les étapes systématiques de développement d'un programme de formation pour les systèmes critiques, l'aspect central de l'analyse de tâches dans ce développement ainsi que les différents moyens existant pour mettre en œuvre les formations.

Le **chapitre 4** a présenté une nouvelle notation de modélisation de tâches afin de résoudre les limitations des notations existantes et de s'intégrer à notre approche de développement.

Le **chapitre 5** a décrit en détail les différentes étapes, les avantages, et les limites de notre approche de conception et développement de systèmes interactifs critiques et/ou complexes et de leurs programmes de formation associés.

Le **chapitre 6** nous a permis de décrire la mise en œuvre l'ensemble de étapes de notre approche avec la notation outillée de modélisation de tâches HAMSTERS, la notation outillée de modélisation du comportement du système ICO-PetShop, la notation outillée de traçabilité des choix et exigences TEAM-DREAMER. Nous avons montré l'exploitation synergique des modèles dans un environnement de conception, spécification et développement logiciel.

Le **chapitre 7** a eu pour objectif de prouver la faisabilité de cette approche sur une étude de cas industrielle, dans le cadre de la conception et du développement d'une application sol de commande et contrôle d'une mission satellitaire et de son programme de formation associé.

Ces différents chapitres nous ont permis de mettre en valeur les éléments essentiels de notre approche :

- L'intégration de différentes vues portant sur la collaboration entre un homme et un système (modèles de tâches et modèles du comportement du système).
- La description synergique des tâches de l'utilisateur et du comportement du système.
- La conception synergique du système et de la formation lui étant associée.
- La formation et le maintien des connaissances et compétences des utilisateurs basés sur les modèles du système qu'ils exploitent.

L'approche proposée nous permet ainsi de montrer qu'il est possible de fournir un support :

- A la conception et au développement de systèmes interactifs critiques de manière à ce qu'ils soient à la fois utilisables, fiables et opérables tout au long de leur mise en exploitation.
- A la conception et au développement d'un programme de formation des utilisateurs de ces systèmes critiques pour améliorer la fiabilité humaine.

Perspectives

Des perspectives à court terme peuvent être dégagées à partir des limitations de notre approche (telles qu'exposées lors de la conclusion du chapitre 5), mais aussi à partir des résultats de la mise en œuvre de notre approche sur des études de cas.

Tout d'abord, la notation de modélisation de tâches utilisateur, qui est au cœur des travaux présentés dans cette thèse, car utilisée dans différentes étapes du processus de développement proposé, peut être étendue moyennent l'ajout de nouveaux éléments :

- Le raffinement des types tâches utilisateur. Une étude approfondie des différents travaux de psychologie cognitive pourrait permettre de raffiner la notation HAMSTERS, en fonction des différentes phases de traitement de l'information, comme celles proposées dans les travaux de (Barnard & Teasdale, 1991). D'autre part, le raffinement des tâches perceptives et motrices pourrait aussi fournir un support à la modélisation des interactions multimodales, comme l'ont montré les travaux de (Jourde, Laurillau, & Nigay, 2010).
- Les différents types d'objets manipulés par l'utilisateur, par le système, et transitant entre l'utilisateur et le système. Cette différenciation permettrait de décrire plus précisément les données, les variables du système ainsi que les objets interactifs. Par la suite, ces éléments de notations pourraient être utilisés pour mieux comprendre la nature des données manipulées par l'utilisateur. Ils fourniraient ainsi un support :
    - A l'analyse des erreurs probables dues par exemple à une divergence entre les données fournies par le système et celles perçues par l'utilisateur, ou entre les données que l'utilisateur pense transmettre au système et celles que le système prend en compte effectivement.
    - Aux choix d'automatisation et à l'analyse de tâches de l'utilisateur dans le cas d'une dégradation du service fourni par le système.
    - Au développement des programmes de formation, en permettant de caractériser finement les connaissances à acquérir par les utilisateurs (Martinie, et al., 2011).
- Les aspects coopératifs et collaboratifs. Comme indiqué dans l'introduction de ce mémoire, cet aspect n'a pas fait l'objet de recherches détaillées dans le cadre de cette thèse mais il est important de le prendre en compte et de l'intégrer à la notation et à l'outil de modélisation. En effet, cette problématique croît dans le domaine des systèmes critiques complexes car l'utilisation de ces systèmes s'effectue de plus en plus dans ce type de contexte.

Ensuite, et de manière générale, l'applicabilité de notre approche aux utilisations collaboratives doit être étudiée pour prendre en compte les problématiques de systèmes hétérogènes et distribués mais aussi le besoin de description de la répartition et de l'ordonnancement des flux de travail entre différents intervenants.

Concernant la modélisation des erreurs humaines pouvant se produire lors de l'utilisation du système et les impacts d'une ou plusieurs défaillances du système sur son utilisation, la phase de modélisation des tâches pourrait être raffinée. La modélisation des tâches utilisateur est bien entendu une étape centrale mais il devrait être décrit si cette étape particulière de modélisation de cas non nominaux reste intégrée dans le processus de développement tel que décrit dans ce mémoire ou si d'autres étapes particulières sont nécessaires. Afin d'améliorer le support fourni à cette activité d'analyse d'erreurs, l'environnement logiciel pourrait intégrer des fonctionnalités de visualisation sélective des modèles d'erreurs d'utilisation associés à un modèle d'utilisation dans le cas nominal.

Concernant la modélisation du système, il reste à étudier comment décrire plus précisément la partie présentation du système. A l'heure actuelle, la notation ICO ne permet pas de la décrire et une solution en cours d'étude est l'utilisation d'un langage de description comme USIXML (Limbourg Q. , Vanderdonckt, Michotter, Bouillon, & Lopez-Jaquero, 2004). D'autre part, la modélisation du système requiert des compétences dans le domaine des méthodes formelles et la modélisation d'un système au comportement complexe et aux nombreuses fonctionnalités peut engendrer des coûts très importants. Ainsi, il devrait être étudié comment sélectionner, lors du processus de développement, les parties du système nécessitant absolument cette modélisation pour optimiser les temps de conception et de développement.

Enfin, comme détaillé dans le chapitre 6, l'outil logiciel de conception rationalisé, DREAM, est extérieur à l'environnement de modélisation synergique et prototypage. Cet outil sera intégré à l'environnement de modélisation afin de fournir un lien direct dans le logiciel, pour le concepteur et pour le développeur, entre les éléments des diagrammes de conception rationnalisée et les éléments des différents modèles. Cette fonctionnalité faciliterait l'exploration directe des choix de conception dans les modèles de tâches et du système correspondants.

La Table 17 récapitule les problématiques décrites pour ces perspectives à court terme en permettant de visualiser leurs composantes en termes d'impact sur le processus de développement, sur les notations et sur l'environnement logiciel de modélisation synergique et prototypage.

Table 17. Récapitulatif des perspectives à court terme

Problématique	Processus de développement	Notations	Environnement logiciel
Gestion des connaissances des utilisateurs dans le cadre de leur formation		Différenciation des types de données pour consigner et contrôler les connaissances à acquérir par l'utilisateur.	Différenciation des types de données dans lors de l'édition des modèles de tâches et pour les contrôler lors de l'exécution des modèles.
Description de la partie Présentation du système		La notation ICO ne permet pas actuellement de décrire la partie présentation du système. Il pourrait être envisagé d'utiliser des notations telles qu'USIXML pour pallier à ce manque.	Edition de la description de la partie présentation du système avec la notation retenue.
Choix de conception pour l'allocation des tâches utilisateur et des fonctions du système		Différenciation des types de données et des types d'objets pour analyser les cas de dégradation du service fourni par le système.	Différenciation des types de données (édition et contrôle lors de l'exécution).
Opérations collaboratives	Investigation sur la nécessité d'ajouter des étapes de développement complémentaires ou d'utiliser des techniques de modélisation complémentaires pour le développement (flux de travail).	Extension des notations HAMSTERS et ICO pour fournir un support à la modélisation d'activités et de systèmes collaboratifs.	Extension des outils logiciels HAMSTERS et ICO pour fournir un support à l'édition et à la simulation de modèles d'activités et de systèmes collaboratifs.
Analyse des erreurs humaines potentielles	Investigation sur la manière d'intégrer l'étape de modélisation des tâches pour les erreurs d'utilisation dans le processus de développement.	Différenciation des types de données et d'objets pour analyser les divergences entre les informations transitant de l'utilisateur vers le système et/ou du système vers l'utilisateur.	Différenciation des types de données (édition et contrôle lors de l'exécution). Fournir un support pour visualiser les modèles d'erreurs potentielles d'utilisation associés au modèle de tâche d'utilisation nominale.
Traçabilité des exigences et choix de conception			Intégration de l'outil logiciel DREAMER dans l'environnement de modélisation synergique et prototypage pour une exploration directe des choix de conception dans les modèles de tâches et du système.

Les perspectives de recherche à plus long terme nous permettent de mettre en valeur les différents domaines, représentés sur la Figure 144, étudiés pour cette thèse.

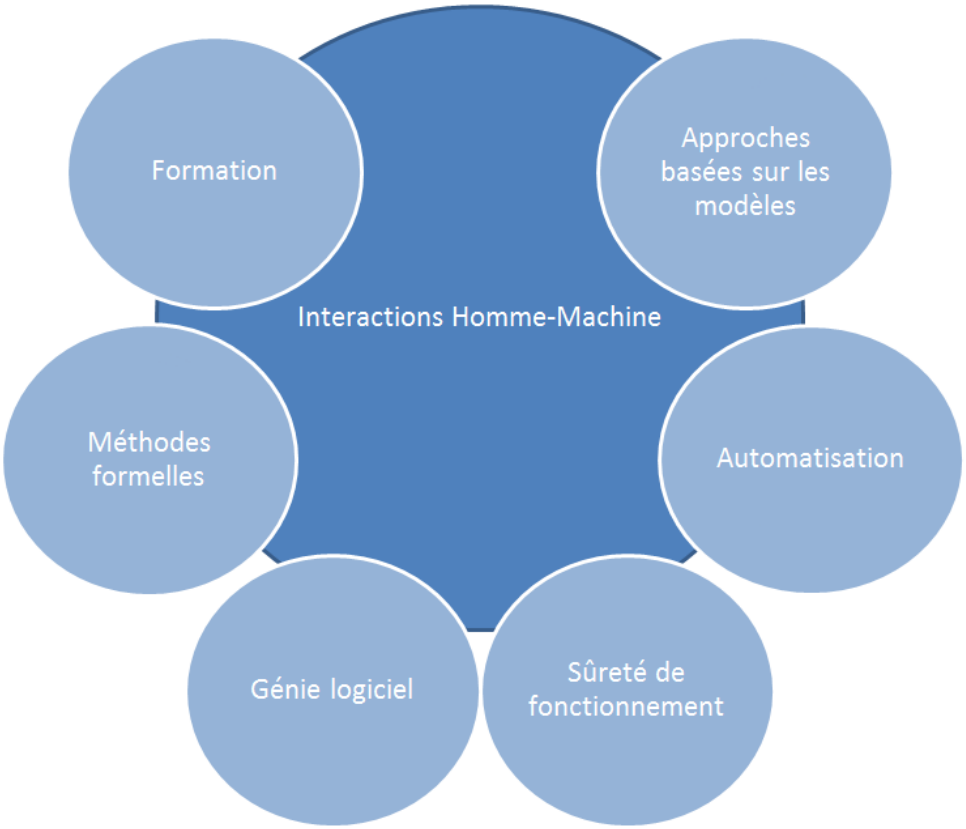


Figure 144. Vue schématique des domaines de recherche explorés au cours de la thèse

Chacun de ces domaines peut faire l'objet d'études intéressantes dans le prolongement des travaux décrits par cette thèse.

Sur le thème du **génie logiciel**, pour pallier au problème des compétences requises pour la modélisation formelle du comportement du système et du temps requis pour modéliser un système complexe et/ou possédant de nombreuses fonctionnalités, il pourrait être investigué comment distinguer les sous-parties du système, en fonction de leur criticité, afin de choisir les sous-parties pour lesquelles une modélisation formelle est nécessaire. Cette distinction pourrait par exemple s'effectuer grâce aux résultats d'analyses de risques effectuées pour des systèmes existants proches en termes de fonctionnalité et de comportement.

Sur le thème de la **sûreté de fonctionnement**, plusieurs axes pourraient être investigués pour améliorer notre approche. Les deux premiers axes décrits ci-après correspondent à une source potentielle d'erreur d'après :

- Comment fournir un support pour éviter les fautes de développement, terme issue de la classification de (Avizienis, Laprie, Randell, & Landwehr, 2004) (détaillée dans la section 2.2 du chapitre 1), pouvant engendrer des interruptions partielles ou totales du service fourni par le système. Une solution envisageable est l'utilisation de composants graphiques tolérants aux fautes, telle que proposée par les travaux de (Tankeu-Choitat, et al., 2011). D'autre part, la vérification formelle du comportement du système est aussi une solution envisageable. Dans notre approche, l'étape de modélisation formelle du système est certes une première étape indispensable mais la vérification de

ces modèles formels ne peut actuellement être que partielle, à cause de la complexité à exprimer mathématiquement des formules de vérification de propriétés sur les Objets Coopératifs. De plus, la vérification est actuellement possible sur les réseaux de Petri sous-jacents mais aucun support n'est fourni par notre environnement intégré de conception et prototypage très haute-fidélité.

- Comment fournir un support pour éviter les fautes d'utilisation. La notation et l'outil de modélisation HAMSTERS fournissent un support à l'analyse des erreurs humaines mais il est nécessaire d'investiguer comment intégrer précisément cette analyse aux phases de notre approche (comme indiqué dans les perspectives à court terme), mais il serait aussi nécessaire d'investiguer comment intégrer des étapes dans notre approche permettant de garantir la propriété de sécurité pour les systèmes interactifs critique. Les étapes d'analyse de risques, de modélisation de barrières de sécurité, et d'analyse des rapports d'accidents et incidents pourraient être intégrées à notre processus de développement moyennant une étude approfondie des notations et outils à utiliser et/ou intégrer à notre environnement de modélisation synergique et prototypage.

Sur le thème de la **formation**, un soin particulier a été accordé aux tâches importantes pour pouvoir qualifier un opérateur avant de l'autoriser à utiliser le système. Mais, les aspects cognitifs liés à l'apprentissage pourraient être investigués plus précisément. Les différents types d'architectures cognitives existantes, comme par exemple ACT-R (Anderson, Matessa, & Douglass, 1995), pourraient fournir un support au développement et à l'évaluation de sessions de formation. Par exemple, ils seraient une aide pour estimer les performances humaines lorsque tout ou partie d'un système est complètement nouveau et qu'un expert ne peut juger de certains intervalles de temps pour accomplir une tâche. Ils pourraient aussi fournir matière à investiguer les types de sessions de formation « intelligentes », guidant en temps réel l'ordonnancement des scénarios à exécuter par la personne formée en fonction du modèle cognitif de l'utilisateur.

Sur le thème des **méthodes formelles**, notre approche répond aux besoins de description complète et non ambigüe du comportement du système ainsi qu'aux besoins de traçabilité des exigences par rapport à la description des différents éléments de comportement. Cependant, il serait utile d'investiguer comment fournir un support aux activités industrielles de certification. Par exemple, il pourrait être analysé précisément comment intégrer les étapes de test et de vérification formelle à notre processus de développement, et comment nos outils de mise en œuvre pourraient supporter ces étapes.

Sur le thème de l'**automatisation**, notre approche permet actuellement d'analyser l'adéquation entre les fonctions exécutées par le système et les tâches exécutées par l'utilisateur. Les perspectives à court terme sur les éléments de notations pour la modélisation des tâches présentés au début de cette section sont une première étape. A plus long terme il pourrait être investigué comment améliorer notre approche pour permettre de concevoir un système autorisant plusieurs degrés d'automatisation, reconfigurable en utilisation (Navarre, Palanque, Barboni, Ladry, & Martinie, 2011), selon les évènements externes ou internes au système, et les besoins de l'utilisateur.

Enfin, sur le thème des **approches basées sur les modèles**, nous avons montré que plusieurs types de modèles sont nécessaires pour qu'un système puisse satisfaire plusieurs propriétés et que leur utilisation de manière synergique optimise leur potentiel. Les types de modèles du système utilisés dans cette thèse ont pour objet les composants logiciels du système interactif. Il pourrait être envisagé de trouver quels modèles pourraient être utilisés pour décrire la partie électronique du système et comment les lier de manière synergique avec les modèles de notre approche. L'intégration de ce type de modèles pourrait améliorer le support à la fiabilité du système et à son opérabilité en permettant de gérer plus en amont les défaillances liées au matériel pouvant engendrer des erreurs humaines.

Récapitulatif des articles publiés dans le cadre de la thèse

Celia Martinie, Philippe Palanque, Marco Antonio Winckler. **Structuring and Composition Mechanism to Address Scalability Issues in Task Models,** IFIP TC13 Conference on Human-Computer Interaction (**INTERACT 2011**), Lisbonne, Portugal, 05/09/2011-09/09/2011, Springer, 2011.

Célia Martinie, Philippe Palanque, Marco Winckler, David Navarre & Erwann Poupart, **Model-Based Training: An Approach Supporting Operability of Critical Interactive Systems: Application to Satellite Ground Segments,** ACM SIGCHI Conference Engineering Interactive Computing Systems (**EICS 2011**), Pise, Italie, 13/06/2011-16/06/2011, ACM SIGCHI, 2011.

Philippe Palanque, Eric Barboni, Celia Martinie, David Navarre, Marco Antonio Winckler, **A Tool Supported Model-based Approach for Engineering Usability Evaluation of Interaction Techniques,** ACM SIGCHI Conference Engineering Interactive Computing Systems (**EICS 2011**), Pise, Italie, 13/06/11-16/06/11, ACM SIGCHI, 2011.

Celia Martinie, Philippe Palanque, Eric Barboni, Marco Antonio Winckler, Martina Ragosta, Alberto Pasquini, Paola Lanzi, **Formal Tasks and Systems Models as a Tool for Specifying and Assessing Automation Designs,** International Conference on Application and Theory of Automation in Command and Control Systems (**ATACCS 2011**), Barcelone, Espagne, 26/05/2011-27/05/2011, IRIT Press, 2011.

Celia Martinie, Philippe Palanque, Eric Barboni, Martina Ragosta. **Task-Model Based Assessment of Automation Levels: Application to Space Ground Segments**. IEEE International Conference on Systems, Man and Cybernetics (**SMC 2011**), Anchorage, 09/10/2011-12/10/2011, IEEE Computer Society - Conference Publishing Services, 2011.

Célia Martinie, Philippe Palanque, David Navarre & Marco Winckler, **A formal approach supporting effective and efficient training program for improving operators' reliability,** Safety and Reliability for managing Risk (**ESREL 2010**), Rhodes Grece, 05/09/2010-09/09/2010, Taylor & Francis Group, p. 234-243, 2010.

Célia Martinie, Philippe Palanque, Marco Winckler, Stéphane Conversy, **DREAMER: a Design Rationale Environment for Argumentation, Modelling and Engineering Requirements,** ACM International Conference on Design of Communication (**ACM SIGDOC'2010**), Sao Paulo Brésil, september 26-29 2010.

Célia Martinie, Jean-François Ladry, David Navarre, Philippe Palanque & Marco Winckler, **Embedding Requirements in Design Rationale to Deal Explicitly with User eXperience and Usability in an "intensive" Model-Based Development Approach,** 5th International Workshop on Model-Driven Development of Advanced User Interface, (**MDDAUI 2010**).

David Navarre, Philippe Palanque, Célia Martinie, Sandra Steere. **Formal Description Techniques for Human-Machine Interfaces - ModelS-Based Approaches for the Design and Evaluation of Dependable Usable Interactive Systems,** The Handbook of Human-Machine Interaction, A Human-Centered Approach, Edited by Guy A. Boy, Florida Institute for Human and Machine Cognition, USA , Ashgate, ISBN: 978-0-7546-7580-8, 2011.

Philippe Palanque, Célia Martinie. **Contextual Help for Supporting Critical Systems' Operators: application to space ground segments,** International workshop and academic/industrial consortium on Activity Context Representation: Techniques and Langages, (**AAAI 2011**), San Francisco, USA, august 7-8th 2011.

David Navarre, Philippe Palanque, Eric Barboni, Jean-François Ladry, Celia Martinie. **Designing for resilience to hardware failures in interactive systems: A model and simulation-based approach**. Elsevier's Safety Science, Elsevier, Special issue : Reliability Engineering and System Safety, Vol. 96, p. 38-52, january 2011.

Références

Africha, H., Haffane, I., Lassalle, A., & Riegert, R. (2011). *Synopsat: interface de supervision satellite segment sol*. Toulouse, France.

AGILE. (2001). *Manifesto for Agile Software Development, http://agilemanifesto.org/*.

Aguinis, H., & Kraiger, K. (2009). Benefits of Training and Development for Individuals and Teams, Organizations, and Society. *Annual Review of Psychology*, pp. 451-475, vol. 60.

Airlines Electronic Engineering Committe. (2002, Avril 22). ARINC 661 specification: Cockpit Display System Interfaces To User Systems. Aeronautical Radio, Inc.

Amokrane, K., Lourdeaux, D., Burkhardt, J.-M., & Barthès, J.-P. (2008). An Intelligent Tutoring System for Training and Learning in a Virtual Environment for High-Risk Sites. *IEEE International Conference on Tools with Artificial Intelligence, ICTAI '08*. Dayton, Ohio: IEEE.

Anderson, J., Matessa, M., & Douglass, S. (1995). The ACT-R theory and visual attention. *Seventeenth Annual Conference of the Cognitive Science Society* (pp. 61-65). Hillsdale, NJ: Lawrence Erlbaum Associates.

Anett, J. (2004). Hierarchical Task Analysis. Dans S. N. Diaper Dan, *The Handbook of Task Analysis for Human-Computer Interaction* (pp. pp. 67-82). Mahwah, New Jersey: Lawrence Erlbaum Associates.

Annett, J., & Duncan, K. D. (1967). Task analysis and training design. *Journal of Occupational Psychology*(41).

Apitz, G., Guimbretière, F., & Zhai, S. (2008, May). Foundations for designing and evaluating user interfaces based on the crossing paradigm. *ACM Transactions on Computer Human Interactions*.

Avizienis, A., Laprie, J.-C., Randell, B., & Landwehr, C. (2004). Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on dependable and secure computing, 1*(1).

Azzouzi, Y., André, F., & Hoarau, R. (2009). *Edition centrée utilisateur de modèle: application au Modèle de tâches.* Toulouse.

Bailey, G. (1993). Iterative methodology and designer training in human-computer interface design. *INTERACT and CHI conference on Human factors in computing systems.* Amsterdam: ACM.

Balbo, S., Ozkan, N., & Paris, C. (2004). Choosing the Right Task Modeling Notation: A Taxonomy. Dans D. D. N., *The Handbook of Task Analysis for Human-Computer Interaction* (pp. 445-465). Mahwah, New Jersey: Lawrence Erlbaum Associates.

Barboni, E., Bastide, R., Lacaze, X., Navarre, D., & Palanque, P. (2003). Petri Net Centered versus User Centered Petri Nets Tools. *10th Workshop Algorithms and Tools for Petri Nets, AWPN 2003.* Eichstätt, Allemagne: Springer.

Barboni, E., Conversy, S., Navarre, D., & Palanque, P. (2006). Model-Based Engineering of Widgets, User Applications and Servers Compliant with ARINC 661 Specification. *DSVIS 2006*, (pp. 25-38).

Barboni, E., J-F., L., D., N., Palanque, P., & M., W. (2010). Beyond Modelling: An Integrated Environment Supporting Co-Execution of Tasks and Systems Models. *ACM SIGCHI conference Engineering Interactive Computing Systems (EICS 2010).* Berlin, Allemagne: ACM SIGCHI.

Barboni, E., Navarre, D., Palanque, P., & Bazalgette, D. (2006). PetShop: A Model Based Tool for the Formal Modelling and Simulation of Interactive Safety Critical Embedded Systems. *International Conference on Human Computer Interaction in Aeronautics, HCI Aero.* Seattle: ACM.

Barnard, P., & Teasdale, J. (1991). Interacting cognitive subsystems: A systemic approach to cognitive-affective interaction and change. *Cognition and Emotion, vol. 5*, pp. 1-39.

Barthet, M.-F., & Tarby, J. C. (1996). The Diane+ method. *Computer-aided design of user interfaces*, (pp. 95-120).

Basnyat, S. (2006). A generic integrated modelling framework for the analysis, design and validation of interactive safety-critical and error-tolerant systems. *PhD Thesis*.

Basnyat, S., Palanque, P., Bernhaupt, R., & Poupart, E. (2008). Formal Modeling of Incidents and Accidents as a Means for Enriching Training Material for Satellite Control Operations. *Joint ESREL 2008 and 17th SRA-Europe Conference. 22 - 25 September, Valencia.*

Bass, L., Little, R., Pellegrino, R., Reed, S., Seacord, R., Sheppard, S., & Szezur, M. (1991). The Arch model: Seeheim revisited. *User Interface Developpers' workshop.*

Bastide, R., & Palanque, P. (1995). A Petri Net Based Environment for the Design of Event-Driven Interfaces. *16th International Conference on Application and theory of Petri Nets (ATPN'95).* Turin, Italie: Springer Verlag.

Bastide, R., & Palanque, P. (2001). Modeling a Groupware Editing Tool with Cooperative Objects. Dans G. Agha, & F. De Cindio, *Concurrent Object-Oriented Programming and Petri Nets.* Springer-Verlag.

Bastide, R., & Palanque, P. (2003). UML for Interactive Systems: What is missing. *Workshop on software engineering and HCI, INTERACT 2003.* Zurich, Suisse: Springer verlag LNCS.

Bastien, C., & Scapin, D. (1993). *Ergonomic Criteria for the Evaluation of Human-Computer interfaces, rapport technique n°156.* Rocquencourt, France: Institut National de Recherche en Informatique et en Automatique.

Baudisch, P., Cutrell, E., Robbins, D., Czerwinski, M., Tand-ler, P., Berderson, B., & Zierlinger, A. (2003). Drag-and-Pop and Drag-and-Pick: Techniques for Accessing Remote Screen Content

on Touch-and-Pen-operated Systems. *IFIP TC 13 International Conference on Human Computer Interactions (INTERACT 2003)* (pp. 57-64). Zurich, Suisse: IOS Press.

Baumeister, L., John, B., & Byrne, M. (2000). A Comparison of Tools Building GOMS Models Tools for Design. *ACM International Conference on Human Factors in Computing Systems (CHI).* La Hague: ACM Press.

Beaudouin-Lafon, M. (1997). *Ingénierie des Systèmes Interactifs*. Consulté le septembre 14, 2011, sur Michel Beaudouin-Lafon: http://www.lri.fr/~mbl/ENS/IHM/ecole-in2p3/Cours/cours1.html

Beck, K. (1999). *Extreme Programming Explained.* Palo Alto, CA: Addison-Wesley.

Bedwell, W., & Salas, E. (2010, September). Computer-based training: capitalizing on lessons learned, vol. 14, issue 3. *International Journal of Training and Development*, pp. 239-249.

Bernhaupt, R., Navarre, D., Palanque, P., & Winckler, M. (2007). Model-Based Evaluation: A New Way to Support Usability Evaluation of Multimodal Interactive Applications. Dans T. H. Law E., *Maturing Usability: Quality in Software, Interaction and Quality (chapter 5).* Springer Verlag.

Bertin, J. (1967). *Sémiologie Graphique - Les diagrammes - les réseaux - les cartes.* Gauthier-Villars et Mouton & Cie.

Bias, R., & Mayhew, D. (2005). *Cost-Justifying Usability, Second Edition: An Update for the Internet Age (Interactive Technologies).* Morgan Kaufmann.

Blanch, R., & Ortega, M. (2011). Benchmarking pointing techniques with distractors: adding a density factor to Fitt's pointing paradigm. *International conference on human factors in computing systems (CHI'11)* (pp. 1269-1638). Vancouver, Canada: ACM.

Blanch, R., Guiard, Y., & Beaudoin-Lafon, M. (2004). Semantic pointing: improving target acquisition with control-display ratio adaptation. *International Conference on Human Factors in Computing Systems* (pp. 519-526). Vienne, Autriche: ACM.

Blumendorf, M., Lehman, G., Feuerstack, S., & Albayrak, S. (2008). Executable Models for Human-Computer Interaction. *DSV-IS. 5136.* Kingston, Canada: Springer.

Bodart, F., Hennebert, A., Leheureux, J., & Vanderdonkt, J. (1994). Towards a dynamic strategy for computer-aided visual placement. *Workshop on Advanced Visual interfaces.* Bari, Italie.

Boehm, B. (1986). A spiral model of software development and enhancement. *SIGSOFT Software Engineering Notes, XI*(4, pp.14-24).

Branson, R., & Grow, G. (1987). Instructional systems development. Dans R. Gagné, *Instructional Technologies: Foundations* (pp. 397-428). Hillsdale, NJ: Lawrence Erlbaum.

Branson, R.K., Rayner, G.T., Cox, J.L., Furman, J.P., King, F.J., & Hannum, W.H. (1975). *Interservice procedures for instructional systems development. Phases I, II, III, IV, V, and executive summary. (TRADOC Pamphlet 350-30).* Fort Monroe, VA: U.S. Army Training and Doctrine Command.

Breedvelt, I., Paterno, F., & Severiins, C. (1997). Reusable structures in task models. *DSVIS* (pp. 251-265). Grenade, Espagne: Springer-Verlag.

Brown, M., & Leveson, N. (1998). Modeling controller tasks for safety analysis. *International Workshop on Human Error and System Development.* Seattle.

Cacciabue, P. (2004). Human error risk management for engineering systems: a methodology for design, safety assessment, accident investigation and training. *Reliability Engineering and System Safety*(83, pp. 229–240).

Caffiau, S., Scapin, D., Girard, P., Baron, M., & Jambon, F. (2010). Increasing the expressive power of task analysis: Systematic comparison and empirical assessment of tool-supported task models. *Interacting with Computers 22,* pp. 569-593.

Calvary, G., Coutaz, J., Thévenin, D., Bouillon, L., Florins, L., Limbourg, Q., . . . Santoro, C. (2002). *The CAMELEON Reference framework, R&D Project IST 2000 30104.*

Calvary, G., Coutaz, J., Thévenin, D., Limbourg, Q., Bouillon, L., & Vanderdonckt, J. (2003). A unifying reference framework for multi-target user interfaces. *Interacting with Computers, 15*(3).

Card, S. K., Newell, A., & Moran, T. P. (1983). *The Psychology of Human-Computer Interaction.*

Chen, P. (1976). The Entity-Relationship Model - Toward a Unified View of Data. *ACM Transactions on Database Systems, 1*(1).

Clark, D. R. (2004). *Instructional System Design Concept Map.* Consulté le juillet 2011, sur http://nwlink.com/~donclark/hrd/ahold/isd.html

Cockburn, A. (2002). *Agile Software Development.* Boston, MA: Addison-Wesley.

Collins, D. (1995). *Designing Object-Oriented user interfaces.* Readwoods City, CA: Benjamin/Cummings Publishing, Inc.

Coutaz, J., Nigay, L., Salbert, D., Blandford, A., May, J., & Young, R. (1995). Four Easy Pieces for Assessing the Usability of Multimodal in Interaction : the CARE Properties. *Human Computer Interaction, INTERACT'95.* Lillehammer: Chapman & Hall (IFIP).

Curtis, B., & Hefley, B. (1994). A WIMP no more: the maturing of user interface engineering. *Interactions, 1*(1).

Dennis, A., & Valachich, J. (1993). Computer Brainstorms: More Heads are Better than One. *Journal of Applied Psychology, 78*(4).

Diaper, D., & Stanton, N. (2004). *The Handbook of Task Analysis for Human-Computer Interaction.* Lawrence Erlbaum Associates.

Diehl, M., & Stroebe, W. (1987). Productivity Loss in Brainstorming Groups: Towards a Solution of a Riddle. *Journal of Personality and Social Psychology, 53*(3).

Dix, A. (1995). Formal methods: an introduction to and overview of the use of formal methods within HCI. Dans M. A., & G. N., *Perspectives on HCI: diverse approaches.* Academic Press.

Dix, A., Finlay, J., Abowd, G., & Beale, R. (2003, 3rd edition). *Human-Computer Interaction.* Harlow: Pearson Education.

DREAMER. (s.d.). *DREAM*. Récupéré sur http://www.irit.fr/recherches/ICS/softwares/dream/

Duncan, J., & Annett, K. (1967). Task Analysis and Training Design. *Journal of Occupational Psychology, 41*, 211-221.

El-Chaar, J., Boer, C., Pedrazzoli, P., Mazzola, S., & Dal Maso, G. (2011). Interactive 3D Virtual Environments for Industrial Operation Training and Maintenance. *International Conference on Reliability, Maintainability and Safety, ICRMS'2011* (pp. 1376-1381). Guiyang: IEEE.

Esteban, O., Chatty, S., & Palanque, P. (1995). Whizz'ed: a Visual Environment for building Highly Interactive Software. *IFIP International Conference on Human-Computer Interaction (Interact'95)* (pp. 121-127). Lillehammer, Norvège: Springer.

EUROCONTROL. (2003, Novembre 6). *Eurocontrol Safety Regulatory Requirement (ESARR) 6, Software in ATM Systems*.

EUROCONTROL. (2003). European Air Traffic Management Program. *ATCO Basic Training - Training plans*. European Organisation for the Safety of Air Navigation.

EUROCONTROL. (2004). European Air Traffic Management Program. *EATM Training Progression and Concepts, HRS/TSP-006-GUI-07*. European Organisation for the Safety of Air Navigation.

European Aviation Safety Agency. (2007, septembre 19). *Certification Specifications for Large Aeroplanes, CS-25, Amendment 3*. Cologne, Allemagne: EASA.

European Cooperation for Space Standardization. (2008). Space Engineering, Ground Systems and Operations, ECSS-E-70C.

European Organisation for Civil Aviation Equipment. (1992). DO-178B, Software Consideration in Airborne Systems and Equipment Certification. EUROCAE.

European Space Agency. (1994). ESA PSS-05-0. Software Engineering Standards, Issue 2, Revision 1. Noordvijk, The Netherlands: ESA Publications Division.

Feary, M., Sherry, L., Polson, P., & Palmer, E. (2000). Evaluation Of A Formal Methodology For Developing Aircraft Vertical Flight Guidance Training Materia. *International Conference on Human-Computer Interaction in Aeronautics*. Toulouse, France.

Federal Aviation Administration. (2011, Septembre). *Training*. Récupéré sur FAA Human Factors: http://www.hf.faa.gov/webtraining/Training/Training1.htm

Fu, W.-T., Bothell, D., Douglass, S., Haimson, C., Sohn, M.-H., & Anderson, J. (2006). Toward a real-time model-based training system. *Interacting with Computers, vol. 18*, pp. 1215-1241.

Gaffar, A., Sinnig, D., Seffah, A., & Forbrig, P. (2004). Modeling patterns for task models. *International Conference on TAsk MOdels and DIAgrams* (pp. 99-104). New-York, NY: Springer-Verlag.

Gagné, R. (1985). *The Conditions of Learning and the Theory of Instruction.* New York: Holt, Rinehart, and Winston.

Gagné, R. (1987). *Instructional Technology: Foundations.* Hillsdale, NJ: Lawrence Erlbaum.

Gagné, R., Wager, W., Golas, K., & Keller, J. (2005). *Principles of Instructional Design.* Florence, KY: Thomson Wadsworth.

Giese, M., Mistrzik, T., Pfau, A., Szwillus, G., & von Detten, M. (2008). AMBOSS: A Task Modelling Approach for Safety-Critical Systems. *HCSE/TAMODIA. 5247, pp. 98-109.* Pise, Italie: Springer.

Gore, B., Hooey, B., Haan, N., Bakowski, D., & Mahlstedt, E. (2011). A Methodical Approach to Build Valid Human Performance Models of Flight Deck Operations. *International Conference in Human Computer Interaction, HCII 2011.* Orlando, Florida.

Graham, C., & Cockton, G. (1996). *Design principles for Interactive Software.* London: Chapman & Hall.

Grossman, T., & Balakrishnan, R. (2005). The bubbe cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. *International Conference on Human Factors in Computing Systems (CHI'05)* (pp. 281-290). Portland, Oregon: ACM.

Gulliksen, J., & Goransson, B. (2003). Usability Design: Integrating User Centered System Design in the Software Development Process. *IFIP International Conference on Human-Computer Interaction, INTERACT.* Zurich, Suisse: Springer.

Harel, D. (1987). StateCharts: A Visual Formalism for Complex Systems. *Science of Computer Programming, vol. 8, n. 3*, pp. 231-274.

Hartson, H., & Hix, D. (1989). Human-computer interface development: concepts and systems for its management. *ACM Computing Surveys, 21*(1).

Hassenzahl, M. (2003). The thing and I: understanding the relationship between user and product. Dans M. Blythe, C. Overbeeke, A. Monk, & P. Wright, *Funology: From Usability to Enjoyment* (pp. 31-42). Dordrecht: Kluwer.

Henry, N., & Fekete, J.-D. (2006). MatrixExplorer: a Dual-Representation System to Explore Social Networks. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization / Information Visualization), 12*(5).

Hewett, Baecker, Card, Carey, Gasen, Mantei, . . . Verplank. (1996). *ACM SIGCHI Curricula for Human-Computer Interaction.* Consulté le 2011, sur ACM SIGCHI: http://old.sigchi.org/cdg/index.html

Hix, D., & Rex Harston, H. (1993). *Developing User Interfaces: ensuring usability through product and process.* Wiley.

Hofer, E., & Ruggiero, F. (1990). Computer-Based Flight Training Evaluation. Dans D. Norrie, & H.-W. Six, *Computer Assisted Learning* (pp. 309-320). Springer Berlin / Heidelberg.

Hollnagel, E. (2006). Task Analysis: The Why, What and How. Dans *Handbook of Human Factors and Ergonomics (3rd Edition) (Ed. G. Salvendy)* (pp. pp. 373-383). New York: Wiley.

Höök, K., Isbister, K., Westerman, S., Gardner, P., Sutherland, E., Vasalou, A., . . . Petta, P. (2011). Evaluation of Affective Interactive Applications. Dans *Emotion-Oriented Systems.* Berlin: Springer.

Hopcroft, J., & Ullman, J. (1990). *Introduction To Automata Theory, Languages, And Computation.* Boston, MA, USA: Addison-Wesley Longman Publishing.

Hussman, H., Meixner, G., & Zuehlke, D. (2011). *Model-Driven Development of Advanced User Interfaces.* Berlin: Springer-Verlag.

International Atomic Energy Agency. (1998). *Experience in the use of Systematic Approach to Training (SAT) for Nuclear Power Plant Personnel, Technical Report, IAEA-TECDOC-1057.*

International Atomic Energy Agency. (2009). IAEA Nuclear Energy Series. *Managing Human Resources in the Field of Nuclear Energy, NG-G-2.1*. Vienna, Austria: IAEA.

International Standard Organisation. (2000). ISO TR 18529:2000 Ergonomics of human system interaction: Human-centred lifecycle process description.

International Standard Organisation. (2001). ISO IEC 9126: Software engineering -- Product quality.

International Standard Organisation. (2008). ISO/IEC 12207:2008(E), Systems and Software Engineering - Software Life Cycle Processes. ISO/IEC-IEEE.

International Standard Organization. (1996). *DIS 9241-11: Ergonomic requirements for office work with visual display terminals (VDT) - Part 11 Guidance on Usability*.

ISO. (1989). ISO 8807 Information Processing Systems - Open Systems Interconnection - LOTOS - A Formal Description Technique Based on temporal Ordering of Observational Behaviour. Genève, Suisse: ISO.

JAA. (2006). *JAR - FCL 1 - Flight Crew Licensing (Aeroplane)*. Joint Aviation Authorities.

Jackson, M. (1983). *System Development.* Englewoog Cliffs, NJ: Prentice Hall.

Jacob, R. (1999). A Software Model and Specification Language for Non-WIMP User Interfaces. *ACM Transactions on Human-Computer Interactions, vol. 6, n. 1*, pp. 1-46.

Johnson, C. (1995). Using Z to support the design of interactive safety-critical systems in Software Engineering Journal. (march).

Johnson, P., & Johnson, H. (1989). Knowledge Analysis of Task; Tast Analysis and Specification for Human-Computer Systems. pp. 119-144.

Jourde, F., Laurillau, Y., & Nigay, L. (2010). COMM Notation for Specifying Collaborative and MultiModal Interactive Systems. *ACM SIGCHI Symposium on Engineering Interactive Computing Systems, EICS 2010* (pp. 125-134). Berlin, Germany: ACM.

Kenney, P., & Loftin, R. (1994). Virtual Environments in Training: NASA's Hubble Space Telescope Mission. *International Conference on Interservice/Industry Training Systems & Education.* Orlando, Florida.

Kincaid, J., & Westerlund, K. (2009). Simulation in Education and Training. *IEEE International Conference on Winter Simulation.* Austin, Texas: IEEE.

Kirkpatrick, D., & Kirkpatrick, J. (2006). *Evaluating Training Programs: The Four Levels, 3rd Edition.* Berrett-Koehler.

Kruchten, P. (2004). *The Rational Unified Process: an Introduction* (Vol. 3rd edition). Boston, MA: Pearson Education, Inc.

Lacaze, X. (2005). La conception rationalisée pour les systèmes interactifs. *PhD Thesis*.

Lacaze, X., Palanque, P., Barboni, E., Bastide, R., & Navarre, D. (2006). From DREAM to Reality : Specificities of Interactive Systems Development With Respect to Rationale Management. Springer Verlag.

Ladry, J.-F. (2010). *Une notion et un processus outillé pour le développement de systèmes interactifs multimodaux critiques.* Toulouse: Université de Toulouse.

Ladry, J.-F., Navarre, D., & Palanque, P. (2009). Formal Description Techniques to Support the Design, Construction and Evaluation of Fusion Engines for SURE (Safe Usable, Reliable and Evolvable) Multimodal Interfaces. *International Conference on Multimodal Interfaces*. 2009: ACM.

Lakos, C. (1991). *Language for Object-Oriented Petri Nets, #91-1*. University of Tasmania: Department of Computer Science.

Lakos, C., & Christensen, S. (1994). A General Systematic Approach to Arc Extensions for Coloured Petri Nets. *15th International Conference on Application and Theory of Petri Nets. LNCS no. 815*. Berlin: Springer.

Leveson, N., & Brown, M. (1998). Modeling Controller Tasks for Safety Analysis. *HESSD 98.*

Lim, K., Long, J., & Silcock, N. (1992). Integrating human factors with the Jackson System Development method: an illustrated overview. *Ergonomics, 35*(10).

Limbourg, Q., & Vanderdonckt, J. (2004). Comparing Task Models for User Interface Design. Dans D. D. N., *The Handbook of Task Analysis for Human-Computer Interaction* (pp. 135-154). Mahwah, New Jersey: Lawrence Erlbaum Associates.

Limbourg, Q., Pribeanu, C., & Vanderdonckt, J. (2001). Towards Uniformed Task Models in a Model-Based Approach. *DSV-IS '01*, (pp. 164-182).

Limbourg, Q., Vanderdonckt, J., Michotter, M., Bouillon, L., & Lopez-Jaquero, V. (2004). USIXML: A Language Supporting Multi-path Development of User Interfaces. *In proceedings of EHCI-DSVIS 2004.* LNCS.

Loftin, R., & Kenney, P. (1994). Virtual Environments in Training: NASA's hubble space telescope mission. *International Conference on Interservice/Industry Training Systems & Education Conference.* Orlando, Florida: NTSA.

Lu, S., Paris, C., & Vander Linde, K. (1998). Towards the automatic generation of task models from object oriented diagrams. *IFIP Working Conference on Engineering for Human-Computer Interaction.* Crète, Grèce: Kluwer academic publishers.

Lu, S., Paris, C., Vander Linde, K., & Colineau, N. (2003). Generating UML Diagrams From Task Models. *CHINZ.* Dunedin, Nouvelle Zélande.

MacLean, A., Young, R. M., Bellotti, V. M., & Moran, T. P. (1991). Questions, options, and criteria: elements of design space analysis. *Hum.-Comput. Interact., 6*(3), 201-250.

Marquardt, N., Jota, R., Greenberg, S., & Jorge, J. (2011). The Continuous Interaction Space: Interaction Techniques Unifying Touch and Gesture On and Above a Digital Surface. *International Conference on Human Computer Interaction (INTERACT'11)* (pp. 461-476, vol. 6948). Lisbon: Springer.

Martinie, C., & Palanque, P. (2011). A Multi-Models Based Development Process for Critical Interactive Systems Integrating Formal and Informal Approaches. *Workshop on Combining Models and Design for interactive systems (ComDeisMoto 2011), IFIP TC 13 conference on Human Computer Interaction (INTERACT 2011).* Lisbonne, Portugal.

Martinie, C., Palanque, P., & Winckler, M. (2011). Structuring and Composition Mechanism to Address Scalability Issues in Task Models. *INTERACT.* Lisbonne, Portugal: Springer.

Martinie, C., Palanque, P., Barboni, E., & Ragosta, M. (2011). Task-Model Based Assessment of Automation Levels: Application to Space Ground Segments . *IEEE International Conference on System, Man and Cybernetics, SMC 2011.* Anchorage, Alaska: IEEE.

Martinie, C., Palanque, P., Barboni, E., Winckler, M., Ragosta, M., Pasquini, A., & Lanzi, P. (2011). Formal Tasks and Systems Models as a Tool for Specifying and Assessing Automation Designs . *International Conference on Application and Theory on Automation in Command and Control Systems, ATACCS 2011.* Barcelone, Espagne: IRIT Press.

Martinie, C., Palanque, P., Navarre, D., & Winckler, M. (2010). A formal approach supporting effective and efficient training program for improving operators' reliability. *European Safety and Reliability Conference.* Rhodes, Grèce: ESRA.

Martinie, C., Palanque, P., Navarre, D., Winckler, M., & Poupart, E. (2011). Model-based Training: An Approach Supporting Operability of Critical Interactive Systems: Application to Satellite Ground Segments. *ACM SIGCHI International Conferences on Engineering Interactive Computing Systems.* Pise, Italie: ACM Press.

Martinie, C., Palanque, P., Winckler, M., & Conversy, S. (2010). DREAMER: a Design Rationale Environment for Argumentation, Modelling and Engineering Requirements. *ACM International Conference on Design of Communication, SIGDOC.* Sao Paulo, Brésil: ACM Press.

Mavin, A., & Maiden, N. (2003). Determining Socio-Technical Systems Requirements: Experiences with Generating and Walking Through Scenarios. *11th International Conference on Requirements Engineering.* Monterey Bay, CA, USA: IEEE Computer Society Press.

McConnell, S. (1996). *Rapid Development: Taming Wild Software. Schedules.* Redmond, WA: Microsoft Press.

McDermid, J., & Ripken, K. (1983). Life cycle support in the Ada environment. *ACM SIGAda Ada Letters, III*(1).

Memon, A., & Xie, Q. (2005). Studying the Fault-Detection Effectiveness of GUI Test Cases for Rapidly Evolving Software. *IEEE Transactions on Software Engineering, vol. 31, no. 10*, pp. 884-896.

Miller, S., Tribble, A., Whalen, M., & Heimdal, M. (2006). Proving the shalls - Early validation of requirements through formal methods. *Software Tools for Technology Transfer, 8*(4).

Myers, P., Watson, B., & Watson, M. (2008). Effective Training Programs Using Instructional Systems Design and E-Learning. *Process Safety Progress, 27*(2).

Narumi, T., Nishizaka, S., Kajinami, T., Tanikawa, T., & Hirose, M. (2011). Augmented reality flavors: gustatory display based on edible marker and cross-modal interaction. *International Conference on Human Factors in Computing Systems (CHI'11)* (pp. 93-102). Vancouver: ACM.

Navarre, D., Palanque, P., & Winckler, M. (2009). Task Models and System Models as a Bridge between HCI and Sofware Engineering. Dans *Human-Centered Software Engineering Software Engineering Models, Patterns and Architectures for HCI.* Springer.

Navarre, D., Palanque, P., Barboni, E., & Mistrzik, T. (2007). On the Benefit of Synergistic Model-Based Approach for Safety Critical Interactive System Testing. Dans Springer (Éd.), *TAMODIA, LNCS 4849.* Toulouse, France.

Navarre, D., Palanque, P., Barboni, E., & Mistrzyk, T. (2007). On the Benefit of Synergistic Model-based Approach for Safety Critical Interactive System Testing. *TAMODIA2007.*

Navarre, D., Palanque, P., Barboni, E., Ladry, J.-F., & Martinie, C. (2011, Janvier). Designing for resilience to hardware failures in interactive systems: A model and simulation-based approach. *Elsevier's Safety Science, Special issue : Reliability Engineering and System Safety, Vol. 96*, pp. 38-52.

Navarre, D., Palanque, P., Ladry, J.-F., & Barboni, E. (2009). ICOs: a Model-Based User Interface Description Technique dedicated to Interactive Systems Addressing Usability, Reliability and Scalability. *Transactions on Computer-Human Interaction, 16*(4).

Navarre, D., Palanque, P., Martinie, C., & Steere, S. (2011). Formal Description Techniques for Human-Machine Interfaces - ModelS-Based Approaches for the Design and Evaluation of Dependable Usable Interactive Systems. Dans G. A. Boy (Éd.), *The Handbook of Human-Machine Interaction. A Human-Centered Approach* (Vol. ISBN 978-0-7546-7580-8). Ashgate.

Neitzel, D. (2006, May, june). How to develop an effective training program. *IEEE Industry Applications Magazine*(May-June).

Nielsen, J. (1994). Heuristic evaluation. Dans J. Nielsen, & R. Mack, *Usability Inspection Methods.* New York, NY: John Wiley and Sons.

Norman, D. A. (2002). *The design of everyday things.* Basic Books.

Norman, D., & Drapper, S. (1986). *User Centred System Design.* U.S.: L. Erlbaum.

Nunes, N., & Cunha, J. (2000). Towards an UML profile for interactive systems development: the Wisdom approach. *UML conference.* Kent, UK: Springer Verlag LNCS.

Object Management Group. (2011). OMG Unified Modeling Language (OMG UML), Infrastructure, Version 2.4.

Okazaki, T., Muromura, M., & Kaynano, J. (2011). A Study on Evaluating Maneuvering skill and Developing Support Tool for Marine Pilot Trainees Berthing a Ship. *IEEE International Conference on Systems, Man and Cybernetics, SMC'2010* (pp. 3087-3093). Istanbul, Turquie: IEEE.

Palanque, P. (1992). *Modélisation par objets coopératifs interactifs d'interfaces homme-machine dirigées par l'utilisateur.* Toulouse: Université Toulouse 1.

Palanque, P. (2011). Cours sur l'ingénierie des systèmes interactifs.

Palanque, P., & Basnyat, S. (2004). Task Patterns for Taking Into Account in an Efficient and Systematic Way Both Standard and Erroneous User Behaviours. *International Conference on Human Error, Safety and Systems Development, HESSD 2004* (pp. 109-130). Toulouse, France: Springer.

Palanque, P., & Bastide, R. (1994). A Formalism for Reliable User Interfaces. *Workshop on Software Engineering and Human Computer Interaction associated with the 16th IEEE ICSE conference.* Sorrento, Italy.

Palanque, P., & Bastide, R. (1997). Synergistic Modelling of Tasks, Users and Systems Using Formal Specification Techniques. *Interacting With Computers, 9*(2).

Palanque, P., & Lacaze, X. (2007). DREAM-TEAM: A Tool and a Notation Supporting Exploration of Options and Traceability of Choices for Safety Critical Interactive Systems. *INTERACT.* Rio, Brésil: Springer Verlag.

Palanque, P., & Martinie, C. (2011). Contextual Help for Supporting Critical Systems' Operators: application to space ground segments. . *International workshop and academic/industrial consortium on Activity Context Representation: Techniques and Langages, (AAAI 2011).* San Francisco, USA.

Palanque, P., Basnyat, S., Blandford, A., Bernhaupt, R., Boring, R., Johnson, C., & Johnson, P. (2007). Beyond usability for safety critical systems: How to be SURE (safe, usable, reliable, and evolvable). *International Conference on Human Factors in Computing Systems (CHI'07)* (pp. 2133-2136). San Jose, California: ACM.

Palanque, P., Bastide, R., & Dourte, L. (1993). Contextual Help for Free with Formal Dialogue Design. *HCII 93.* Orlando.

Palanque, P., Bastide, R., & Paternò, F. (1997). Formal Specification As a Tool for the Objective Assessment of Safety Critical Interactive Systems. *INTERACT.* Sydney, Autralie: Chapman et Hall.

Palanque, P., Bastide, R., & Sengès, V. (1995). Validating Interactive System Design Through the Verification of Formal Task and System Models. *IFIP Working Conference on Engineering for Human-Computer Interaction.* Grand Targhee Resort, Wyoming, USA: Chapman et Hall.

Palanque, P., Ladry J-F., Navarre, D., & Barboni, E. (2009). High-Fidelity Prototyping of Interactive Systems can be Formal too. *13th International Conference on Human-Computer Interaction.* San Diego, CA, USA.

Palanque, P., Long, J., Tarby, J., Barthet, M., & Lim, K. (1994). Conceptions d'applications ergonomiques: une méthode pour informaticiens et une méthode pour ergonomes. *Ergonomie et Informatique Avancée.* Biarritz.

Palanque, P., Navarre, D., & Gaspard-Boulinc, H. (2000). *MEFISTO method version 1, ESPRIT Reactive LTR 24963, WP2-7.*

Parasuraman, R., Sheridan, T., & Wickens, C. (2000, May). A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Mans and Cybernetics, Part A: Systems and Humans, vol. 30, n.3,* pp. 286-297.

Paterno, F. (2004). ConcurTaskTrees: An Engineered Notation for Task Models. Dans D. Diaper, & N. Stanton, *The Handbook of Task Analysis for Human Computer Interaction.* Lawrence Erlbaum Associates.

Paterno, F., & Zini, E. (2004). Applying information visualization techniques to visual representations of task models. *TAMODIA.* New York, NY, USA: ACM.

Paternò, F., Breedvelt-Schouten, I. M., & de Koning, N. (1999). Deriving Presentations from Task Models. *Proceedings of the IFIP TC2/TC13 WG2.7/WG13.4 Seventh Working Conference on Engineering for Human-Computer Interaction*, (pp. 319-337).

Paterno, F., Mancini, C., & Meniconi, S. (1997). ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models. *INTERACT.* Sidney, Autralie: Chapman & Hall.

Patrick, J. (1992). *Training: Research and Practice.* San Diego, CA: Academic Press.

Petri, C. A. (1962). Kommunikation mit Automaten. *PhD Thesis*. Technical University Darmstadt.

Puerta, A., & Eisenstein, J. (1999). Towards a general computational framework for model-based interface development systems. *IUI.* Los Angeles, CA, USA.

Puerta, A., & Eisenstein, J. (2002). A common representation for interaction data. *In Proc. Of the 7th International Conference on Intelligent User Interfaces.* Santa Fe: ACM press.

Rajeev, A. (1999). Timed Automata. *11th International Conference on Computer Aided Verification.* London, UK: Springer-Verlag.

Ramamoorthy, C., & Ho, G. (1980). Performance Evaluation of Asynchronous Concurrent Systems. *IEEE Transactions of Software Engineering, 6*(5).

Rasmussen, J., & Vicente, K. (1989). Coping with human errors through system design: Implications for ecological interface design. *International Journal of Man-Machine Studies*(31, pp. 517-534).

Rational Software Corporation. (1997). *UML Notation Guide. 1.1*.

Rauterberg, M. (1992). An Iterative-Cyclic Software Process Model. *International Conference on Software Engineering and Knowledge Engineering.* Capri, Italie: IEEE.

Reason, J. (1990). *Human error.* Cambridge: Cambridge University Press.

Reichart, D., Dittmar, A., Forbrig, P., & Wurdel, M. (2008). Tool Support for Representing Task Models, Dialog Models and User-Interface Specifications. *DSVIS. LNCS 5136.* Kingston, Canada: Springer.

Reiser, R. (1987). History. Dans R. Gagné, *Instructional Technology: Foundations*. Hillsdale, NJ: Lawrence Erlbaum.

Reiser, R. (2001). A History of Instructional Design and Technology: Part II: A History of Instructional Design dans Educational Technology Research and Development. *49*(2).

Royce, W. (1970). *Managing the development of large software systems: Concepts and techniques.* WESCON technical paper.

Salas, E., & Cannon-Bowers, J. (2001). The Science of Training: A Decade of Progress. *Annual Review of Psychology*, pp. 471-499.

Santoro, C. (2005). Chapter 6 - Using Task Models to Verify Properties of an Interactive System. Dans C. Santoro, *A Task-Model Based Approach for the Design and Evaluation of Innovative User Interfaces.* Louvain-la-Neuve: Presses Universitaires de Louvain.

Sawyer, J., Minsk, B., & Bisantz, A. (1996). Coupling User Models and System Models: A Modeling Framework for Fault Diagnosis in Complex Systems. *Interacting with computer*.

Scapin, D., & Pierret-Golbreich, C. (1989). Towards a method for task description: MAD. *Work with DisplayUnits WWU'89*, (pp. 27–34).

Schmidt, D. (2006). Model-Driven Engineering. *IEEE Computer, 39*(2).

Scott, D. (2009). Growing a Training System and Culture for the Ares I Upper Stage Project. *Aerospace Conference.* Big Sky, MT: IEEE.

Seamster, T., Boehm-Davis, D., Holt, R., & Schultz, K. (1998, August 1). *Developping Advanced Crew Resource Management (ACRM) Training: A Training Manual*. Washington DC, USA: Federal Aviation Administration.

Sebillotte, S. (1988). Hierarchical planning as method for task analysis: the example of office task analysis. *Behaviour & Information Technology, 7*(3).

Selby, R. (2009). Synthesis, Analysis, and Modeling of Large-Scale Mission-Critical Embedded Software Systems. *International Conference on Software Process: Trustworthy Software Development Process.* Vancouver: Springer-Verlag.

Shepherd, A. (1985). Hierarchical task analysis and training decisions. *Programmed Learning and Educational Technology*(22).

Shwaber, K., & Beedle, M. (2002). *Agile Software Development with SCRUM.* Upper Saddle River, NJ: Prentice-Hall.

Sinnig, D., Forbrig, P., Wurdel, M., Chalin, P., & Khendek, F. (2007). Practical extensions for Task models. *International Conference on TAsk MOdels and DIAgrams, TAMODIA* (pp. 42-55 (vol. 4849)). Heidelberg: Springer.

Sommerville, I. (2006). *Software engineering* (éd. 7th). Addison-Wesley.

Storey, N. (1996). *Safety-critical computer systems.* Addison-Wesley.

Sukaviriya, P., Kovacevic, S., Foley, J., Myers, B., Olsen Jr., D., & Schneider-Hufschmidt, M. (1994). Model-Based User Interfaces: What Are They and Why Should We Care? *ACM Symposium on User Interface Software and Technology* (pp. 133-135). Marina del Rey, CA, USA: ACM.

Tankeu-Choitat, A., Fabre, J.-C., Palanque, P., Navarre, D., Deleris, Y., & Fayollas, C. (2011). Self-Checking Components for Dependable Interactive Cockpits using Formal Description Techniques. *17th Pacific Rim Dependable Computing Conference (PRDC 2011).* Pasadena, US.

Tarby, J., & Barthet, M. (1996). The Diane+ method. *2nd International Workshop on Computer-Aided Design of User Interfaces.* Presses Universitaires de Namur, Vanderdonckt, J.

Tardieu, H., Rochfeld, A., & Colletti, R. (1986). *La méthode MERISE, principes et outils.* Edition d'Organisation.

Thimbleby, H. (2007). User-centered methods are insufficient for safety-critical systems. *Usability and HCI for medicine and healthcare. lncs 4799.* Graz: Springer.

U.S. Army Field Artillery School. (1984). *A System Approach To Training (Course Student textbook). ST - 5K061FD92* .

Van Der Veer, G. C., Lenting, V. F., & Bergevoet, B. A. (1996). Gta: Groupware task analysis - modeling complexity. *Acta Psychologica, 91,* 297-322.

Wherry, E. (2003). Scroll ring performance evaluation. *International Conference on Human Factors in Computing Systems (CHI'03).* Fort Lauderdale: ACM.

Winckler, M., Vanderdonckt, J., Trindade, F., Stan, & Stanciulescu, A. (2008). Cascading Dialog Modeling with UsiXML. *DSVIS. LNCS 5136.* Kingston, Canada: Springer.

Wood, W. (1970). Transition network grammars for natural language analysis. *Communications of the ACM, 13*(10).

TITLE: A Synergistic Models-Based Approach to Develop Usable, Reliable and Operable Interactive Critical Systems.

ABSTRACT

User Centered Design is today a reference to design and develop interactive systems. All the existing techniques, methods and development processes based on this paradigm are targeting to understand and know the user. They aim at designing and developing systems that match their needs, skills and behaviours. These techniques, methods and development processes are improved regularly and are spreading through the mass market industries. The underlying philosophy is to focus on the affordance of the designed system, so that it will be used efficiently and that it will give satisfaction to the user, even at first use. However, in the case of interactive critical systems (in the application domains of aeronautics, aerospace or nuclear energy for example), the cost of a usage error or of a system failure can overcome the cost of the development of the system itself, and can result in loss of life, injury or damage to the system and its environment. User Centered Design techniques, methods and processes are then not sufficient, as they are not handling all of the design and development issues that are associated to interactive critical systems. First of all, these techniques, methods and processes do not enable to guarantee that the system will fulfil both usability and reliability properties. Then, they do not consider training and qualification of the users of the system. At last, they do not provide means for traceability of the needs and requirements through the whole development process.

This thesis states that it is possible to develop an interactive critical system and its associated training program, in an integrated way, in order to guarantee that the system fulfils properties of usability, reliability and operability, and that the users are qualified before using the system. To demonstrate this statement, we analyse existing development processes, techniques and methods and we highlight their advantages and limitations. From the outcome of our study, we propose an approach to develop interactive critical systems that are usable, reliable and operable and we describe the associated conceptual framework of our approach. We propose an implementation of this approach with a development process, notations and a software environment. The development process integrates phases for the development of the associated training program, and it provides support for the traceability of requirements and design choices during the whole phases of the process. This approach takes advantages from the User Centered Design paradigm and uses, in a synergistic way, task models, system's behaviour formal models and training program development model. We also propose a new task modelling notation, HAMSTERS (Human-centered Assessment and Modelling to Support Task Engineering for Resilient Systems), in addition to an extension of the TEAM (Traceability Exploration and Analysis Model) design rationale notation, and a synergistic modelling and prototyping software environment.

The proposed development process, notations and software environment are applied in several examples and in a large case study of the development of a satellite command and control ground segment application.

KEYWORDS

Interactive critical systems, Human Computer Interaction, model-based design, software engineering, formal methods, systematic approach to training.

AUTEUR : Célia MARTINIE DE ALMEIDA

TITRE : Une approche à base de modèles synergiques pour la prise en compte simultanée de l'utilisabilité, la fiabilité et l'opérabilité des systèmes interactifs critiques.

DIRECTEUR DE THESE : Philippe PALANQUE

LIEU ET DATE DE SOUTENANCE : 5 décembre 2011, Université Toulouse 3 (IRIT)

RESUME

Dans le cadre de la conception et du développement de systèmes interactifs critiques, lorsque le coût d'une erreur potentielle d'utilisation ou d'un dysfonctionnement du système peut dépasser le coût de développement de ce système ou se chiffrer en pertes humaines, les techniques, méthodes et processus actuellement proposés dans le domaine de l'IHM sont difficilement exploitables. D'une part, ils ne permettent pas de garantir simultanément les propriétés d'utilisabilité et de sûreté du système développé. D'autre part, la formation et la qualification des utilisateurs du système avant sa mise en opération n'est pas envisagée. Enfin, ces techniques, méthodes et processus ne fournissent pas les moyens de traçabilité exigés pour le développement de systèmes critiques.

L'argumentaire de cette thèse s'appuie sur les avantages et limitations des approches existantes en termes de processus et notations de modélisation. Nous proposons une approche et montrons sa réalisation à travers un processus de développement d'un système interactif critique et de son programme de formation associé. Ce processus fournit un cadre conceptuel, une association d'étapes, des notations, et un environnement logiciel pour: le développement d'un système utilisable et sûr, le développement du programme de formation associé ainsi que la traçabilité des exigences et des choix de conception tout au long des différentes étapes. Il utilise certains principes de la conception centrée utilisateur et exploite de manière synergique les modèles des tâches, les modèles formels du comportement du système et le modèle de développement du programme de formation.

MOTS-CLES

Systèmes interactifs critiques, Interaction Homme Machine, génie logiciel, méthodes formelles, ingénierie des modèles, approche systématique à la formation.

DISCIPLINE ADMINISTRATIVE

Informatique

INTITULE ET ADRESSE DE L'U.F.R. OU DU LABORATOIRE :

Institut de Recherche en Informatique de Toulouse (IRIT)

118, route de Narbonne 31062 Toulouse Cedex 9