Identification systématique et représentation des erreurs humaines dans les modèles de tâches

Racim Fahssi

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 *Racim Fahssi*
Le 14 décembre 2018

Titre : *Identification systématique et représentation des erreurs humaines dans les modèles de tâches*

JURY
Káthia Marçal De Oliveira (Maîtresse de conférence, Université de Valenciennes, Rapporteure*)*
Jean Vanderdonckt (Professeur, Université Catholique de Louvain, Rapporteur)
Jean-Claude Tarby (Maitre de conférence, Université Lille 1, Examinateur)
Marco Winckler (Professeur, Université Nice Sophia Antipolis, Président du jury)
Philippe Palanque (Professeur, Université Paul Sabatier, Directeur de thèse)
Célia Martinie (Maître de conférence, Université Paul Sabatier, co-encadrante)
Christine GRIS, (Experte Flight Warning, AIRBUS Group, Membre Invitée)

Ecole doctorale: *Mathématiques Informatique et Télécommunications de Toulouse (MITT)*
Unité de recherche: *IRIT – UMR 5505*
Directeur(s) de Thèse: *Philippe Palanque et Célia Martinie*
Rapporteurs: *Káthia Marçal De Oliveira et Jean Vanderdonckt*

Remerciements

Il me sera très difficile de remercier tout le monde car c'est grâce à l'aide de nombreuses personnes que j'ai pu mener cette thèse à son terme.

Je voudrais tout d'abords remercier Célia Martinie et Philippe Palanque, grâce à qui j'ai eu l'opportunité de préparer cette thèse, et la chance de découvrir le monde de la recherche académique. Qu'ils soient aussi remerciés pour leur gentillesse, leur disponibilité quasi permanente, pour les nombreux encouragements qu'ils m'ont prodigués et surtout de leur patience.

La préparation de cette thèse a aussi été rendue possible grâce au financement de AIRBUS, notamment par le biais du projet IKKY. Ainsi, je remercie Christine Gris et Yannick Deleris membres de ce projet, pour la confiance qu'ils nous ont accordée, pour leur aide, ainsi que pour leur avis constructif.

Les travaux présentés dans cette thèse sont issus de nombreux échanges et discussion avec Eric Barboni et David Navarre, qui m'ont aussi guidé et orienté à travers leurs conseils. Travailler avec eux a été énormément enrichissant pour moi et très plaisant. Je les remercie pour leurs soutiens, leurs encouragements. Aussi, je tiens à remercier particulièrement Marco, pour tout le soutien, aide, conseil, encouragement et son humour apportés avant et pendant la thèse, et que j'ai eu le regret de le voir quitter notre équipe.

J'adresse tous mes remerciements à Káthia Marçal De Oliveira et Jean Vanderdonckt, de l'honneur qu'ils m'ont fait en acceptant d'être les rapporteurs de mon mémoire, en consacrant ainsi de leur temps pour étudier cette thèse et me prodiguer leurs avis et conseils. Je remercie également Marco Winckler et Jean-Claude Tarby d'avoir accepté de faire partie du jury de ma thèse.

Je remercie également les membres de l'équipe ICS, actuels et anciens, pour leur soutien et encouragements, et en particulier Camille Fayolas, Martina Ragosta, Regina Bernhaupt, Elodie Bouzekri, Arnaud Hamon, Martin Cronel, Cédric Bach, Caio Stein Dagostini, Bastien Gatellier, Antoine Desnos, François Manciet, José Luis Silva, Oussama Sabra, Jean Luc Hak, Allexandre Battut, Alexandre Canny, Dimitri Drouet, Guillaume Pottier, Thiago Rocha Silva, et tous ceux que j'ai peut-être oublié et qui ont apporté joie et bonne humeurs au sein de l'équipe.

Je tiens aussi à remercier tous les membres de l'équipe SIERA, qui m'ont apporté un grand soutien durant ma thèse, en particulier Messaoud, Rémi, Romain, Cédric, Samer, Arnaud, Pierrick, André, François, Malek, Julien, Thierry, Michel, Manu, Daniel, Franck, Patrice et Bachar.

Je suis très reconnaissant à mes parents et ma sœur, sans eux, je ne serai jamais arrivé à là où j'en suis, ainsi qu'à mon entourage proche et familial qui m'ont soutenu de près ou de loin. Je remercie Mamer, Adel, Mahiou, Karim, Mehdi, Seb, Alexis, Alex, Mohamed, Amine, Abdou, Yves Cu., Yves Ch., Gildas, Fabien, Giles, Hélène, Mamar, Sami, Domi, Najim pour leur soutien et amitité. Je tiens aussi à remercier ma belle-mère pour son soutien. Je remercie ma femme Aquila, qui ma soutenue jusqu'au bout en m'apportant paix et joie. Enfin, je remercie mes deux garçons Abdelkader et Manil, qui me m'ont donné le courage de mener à bien ma thèse.

Merci à tous les autres que je n'ai pas cité et qui compte dans ma vie.

Table des matières

Liste des figures

Liste des tables

Introduction

Le domaine de l'Interaction Homme-Machine[1] a, au cours des dernières années, atteint un niveau de maturité certain dans le domaine académique avec un portfolio de conférences important, un nombre de laboratoire et d'équipes de recherche et d'enseignant chercheurs en augmentation. A titre d'exemple le nombre d'articles acceptés à la conférence ACM SIGCHI en 2018 est supérieur au nombre total de soumissions à la même conférence en 2000 (avec un taux d'acceptation similaire). Dans le domaine industriel la progression a été plus longue à démarrer mais est maintenant forte augmentation comme le démontre la grande quantité d'offre d'emploi dans le domaine maintenant clairement identifiée UX/UI.

Il semble donc clair que le paradigme de la conception centrée utilisateur (Norman & Drapper, 1986) soit aujourd'hui au cœur des processus de développement des systèmes informatiques interactifs même si les approches agiles (Schawber 2002 - méthode SCRUM) se focalise sur la relation client-développeur plus que sur la relation utilisateur-outil. Dans ces approches centrées utilisateur, 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. 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 » et « 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.

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 facilement mesurable et que cette valeur a un impact direct sur la performance (à la fois en termes de production et de coût financier) des entreprises qui les emploient. L'aspect « effectiveness » est moins considéré car les systèmes interactifs grand public (e. g. sites webs), qui concentrent une grande partie des développements actuels, se limitent souvent à un nombre de tâches utilisateur réduit et relativement simple (remplir un formulaire, chercher une information, effectuer un paiement, …).

Le domaine de l'Interaction Homme-Machine se concentre principalement sur les moyens d'augmenter la performance des utilisateurs principalement au moyen de techniques d'interaction innovantes comme par exemple le drag and pop (Baudisch, et al., 2003) et le buble cursor, (Grossman & Balakrishnan, 2005) présentés en Figure 1.

[1] Que l'on peut assimiler à « Special Interest Group on Human-Computer Interaction - SIGCHI » de l'ACM www.sigchi.org

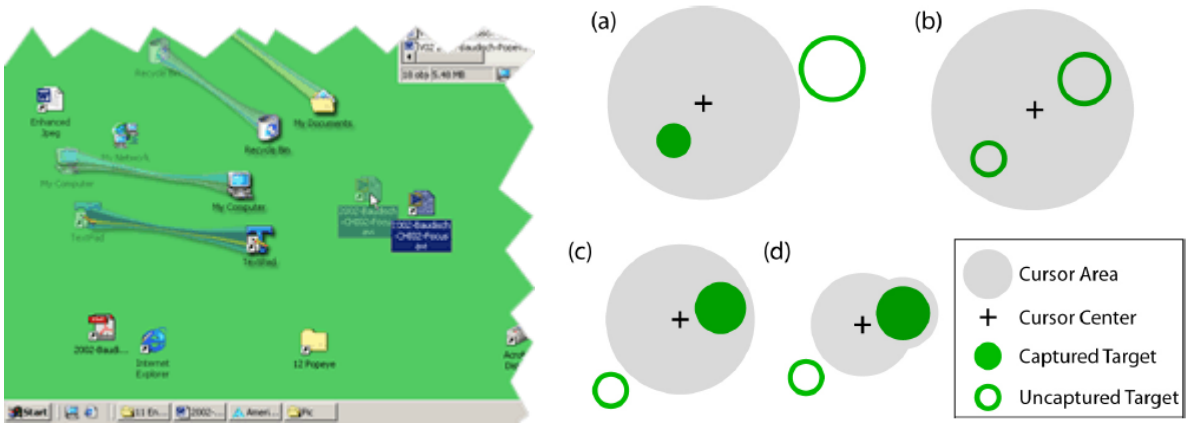


Figure 1. Illustration des techniques d'interaction « Drag and Pop » (Baudisch, et al., 2003) et « Bubble cursor » (Grossman & Balakrishnan, 2005)

Pour ce travail sur l'amélioration des performances, le domaine de l'Interaction Homme-Machine trouve sa source dans des travaux fondamentaux comme ceux de Stuart Card (Card, Newell, & Moran, 1983) et (Card, Moran, & Newell, 1980). Dans (Card, Moran, & Newell, 1980), on voit explicitement la prise en compte des erreurs pour le calcul des performances, alors que dans (Card, Newell, & Moran, 1983) le schéma d'interaction entre l'utilisateur et la machine, (voir Figure 2) montre une interaction sans erreur allant de la perception des informations présentées par le système interactifs à la manipulation de systèmes d'entrée et en passant par un traitement cognitif des informations perçues.

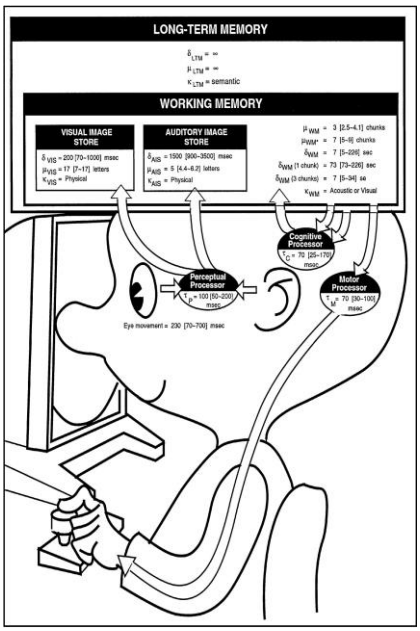


Figure 2. Représentation schématique de l'interaction entre un utilisateur et un système interactif (connu sous le nom de modèle du processeur humain) – extrait de (Card, Newell, & Moran, 1983)

Cette vision sans erreur suivie largement par la communauté en Interaction Homme-Machine est très différente de celle de la communauté en Facteur Humain[2] qui depuis ses débuts s'intéresse à comprendre les causes des erreurs humaines et leur impact sur la performance mais

[2] Que l'on peut assimiler à la société Human Factors and Ergonomics society https://www.hfes.org/

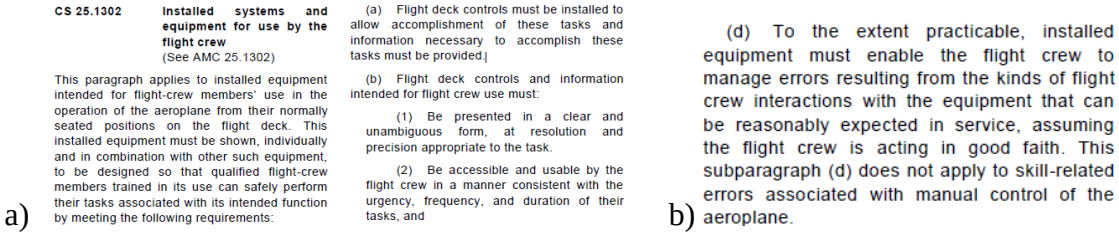
aussi sur des aspects majeurs comme la sureté de fonctionnement et la fiabilité des utilisateurs et de leur travail. On peut par exemple citer les travaux de (Reason, 1990) et (Hollnagel, 1993) qui ont proposé des classifications d'erreurs humaines permettant à la fois de parler de ces erreurs (en définissant un vocabulaire spécifique) mais aussi, entres autres, d'identifier leurs causes (via la structure de la classification et les relations entre erreurs). 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) ainsi que, à la fois la mise en place de barrières pour prévenir leur occurrence (e. g. messages de confirmation) et la mise en place de mécanismes de correction des erreurs lorsqu'elles se sont produites e. g. commande undo/redo).

Ce problème d'erreur prend une toute autre dimension dans le domaine des systèmes interactifs critiques, où une défaillance du système peut se chiffrer en nombre de vies humaines ou en coûts matériels exorbitants. Actuellement, la complexité de ces systèmes et des informations qu'ils manipulent engendre une forte demande pour l'intégration de techniques d'interaction évoluées (ou à tout le moins plus évoluées que les techniques d'interaction actuellement déployées). Par exemple, la Figure 3, présente une vue 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 le cas de ces systèmes, la certification est une phase du processus de développement majeure. 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 (FAA) ou la European Aviation Safety Agency (EASA) dans le domaine aéronautique) n'autoriseront pas l'exploitation du système. En ce qui concerne les systèmes interactifs les spécifications de certification CS 251302 (EASA, 2007) exigent des garanties très précises. Les plus proches de ce travail de thèse concernent la section 13.02 de la CS 25 intitulée « Installed

systems and equipment for use by the flight crew » qui définit les contraintes d'opération pour le pilote et le co-pilote. Dans cette section de la CS 25 (voir Figure 4) on peut remarquer que la certification se base sur une compréhension exhaustive des tâches des pilotes ainsi que des informations nécessaires à l'accomplissement de ces tâches.



a)

CS 25.1302 **Installed systems and equipment for use by the flight crew**
(See AMC 25.1302)

This paragraph applies to installed equipment intended for flight-crew members' use in the operation of the aeroplane from their normally seated positions on the flight deck. This installed equipment must be shown, individually and in combination with other such equipment, to be designed so that qualified flight-crew members trained in its use can safely perform their tasks associated with its intended function by meeting the following requirements:

(a) Flight deck controls must be installed to allow accomplishment of these tasks and information necessary to accomplish these tasks must be provided.

(b) Flight deck controls and information intended for flight crew use must:

(1) Be presented in a clear and unambiguous form, at resolution and precision appropriate to the task.

(2) Be accessible and usable by the flight crew in a manner consistent with the urgency, frequency, and duration of their tasks, and

b)

(d) To the extent practicable, installed equipment must enable the flight crew to manage errors resulting from the kinds of flight crew interactions with the equipment that can be reasonably expected in service, assuming the flight crew is acting in good faith. This subparagraph (d) does not apply to skill-related errors associated with manual control of the aeroplane.

Figure 4. Extrait de la CS 25 section 13.02 (EASA, 2007). Partie a) de la figure concerne la description de tâches. Partie b) la prise en compte des erreurs

La partie b) de la Figure 4 (issue aussi de la CS 25 section 13.02) met en avant la nécessité d'offrir des mécanismes de gestion et de récupération des erreurs éventuelles produites par les opérateurs. Ceci démontre l'importance d'aller au-delà des pratiques actuelles dans le domaine de l'Interaction Homme-Machine et prenant en compte de façon explicite la capacité des opérateurs à faire des erreurs. Ceci nécessite donc l'intégration des connaissances sur les erreurs humaines (issues du domaine des Facteurs Humains) et de les intégrer avec les aspects utilisabilité, non seulement au niveau performance (ce qui est pris en compte par exemple au niveau des tests d'utilisabilité) mais aussi au niveau tâches à réaliser ce qui n'est malheureusement qu'abordé de façon superficielle dans des travaux comme (Palanque & Basnyat, 2004) où des patrons d'erreurs sont présentés, ou (Paternò & Santoro, 2001) qui propose une identification des erreurs humaines en utilisant la méthode HAZAOP (Lawley, 1973) sur des modèles de tâches.

Hypothèses de la thèse

Dans cette thèse notre objectif est de démontrer qu'il est possible de décrire de façon systématique les tâches des opérateurs ainsi que les informations et les connaissances nécessaire à la réalisation de ces tâches. Partant de ce premier objectif, notre second objectif est de démontrer qu'il est possible de décrire de façon systématique les erreurs pouvant survenir lors de l'accomplissement de ces tâches.

L'hypothèse de cette thèse est centrée autour de la notion de modèles de tâches et d'erreurs humaines. En effet, le contenu de ce mémoire a pour objectif de démontrer qu'une approche **à base de modèles de tâches associée à un processus de description des erreurs humaines supportée** par un **ensemble d'outils** permet d'atteindre ces objectifs.

Structure du mémoire

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

Le **chapitre 1** décrit les critères de sélection habituellement utilisés pour la modélisation des tâches dans le domaine des systèmes interactifs, ainsi que les critères que nous mettrons en valeurs pour l'aide à l'identification des erreurs humaines.

Le **chapitre 2** présente les concepts principaux utilisés pour caractériser les erreurs humaines, ainsi que les taxonomies utilisées pour les classer.

Le **chapitre 3** présente un état de l'existant des différentes techniques et méthodes d'identification d'erreurs humaines.

Le **chapitre 4** présente les taxonomies retenues pour l'identification et la représentation des erreurs humaines dans les modèles de tâches, ainsi que les extensions apportées à la notation et à l'outil HAMSTERS pour l'identification et la représentation des erreurs humaines dans les modèles de tâches.

Le **chapitre 5** présente un processus outillé qui permet d'identifier et de décrire de manière systématique les erreurs humaines en s'appuyant sur les modèles de tâches.

Le **chapitre 6** présente l'état de la notation et de l'outil HAMSTERS suite aux modifications apportées pour le support à l'approche systématique d'identification et de représentation des erreurs humaines dans les modèles de tâches. Ce chapitre présente aussi les résultats de la mise en œuvre de ces modifications à une étude de cas industrielle dans le domaine d'application de l'aéronautique.

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.

PARTIE I – État de l’art

Chapitre 1. – Techniques de descriptions des tâches utilisateurs

L'analyse des tâches est une méthode qui a longtemps été utilisée comme étape dans le processus de la conception des systèmes interactifs, du début de la spécification des exigences jusqu'à l'évaluation finale du système (Stuart & Penn, TaskArchitect: taking the work out of task analysis, 2004). Ainsi pour exprimer les besoins des utilisateurs lors du processus de conception, les modèles de tâches ont été proposés. Ces modèles rarement outillés ne permettent pas d'effectuer des raisonnements sur l'ensemble des aspects concernant l'activité de l'utilisateur : les tâches, les objets et les erreurs humaines et système ayant une influence directe sur le déroulement de cette activité.

Chaque approche de l'analyse de la tâche a sa propre syntaxe et la méthode d'affichage des résultats. Ils ont été développés pour guider l'analyste dans démêler les problèmes de performance humaine pertinents pour une utilisation dans les activités de conception. Toutefois, ils ajoutent également aux travaux à réaliser- la syntaxe doit être maintenue grâce à des contrôles ; les résultats à nouveau affichés comme des modifications sont apportées. En dépit de cette tête, l'analyse de la tâche continue d'être utilisée par un grand nombre de praticiens.

Ce chapitre décrit les critères de sélection habituellement utilisés dans le domaine des systèmes interactifs ainsi que les critères que nous mettrons en valeurs pour l'aide à l'identification des erreurs humaines.

1 Définition de la description de tâches

Les modèles de tâches ont été créés pour décrire l'activité de l'utilisateur d'un système. Ils sont apparus en ergonomie dans les années 1960 (Annett & Duncan, 1967) en réponse à la complexité croissante de l'activité des opérateurs dans les industries dites lourdes (métallurgie, pétrochimie, …). Les modèles de travail d'alors étaient adaptés au travail à la chaîne mais ne permettaient pas de prendre en compte les tâches "cognitives", ou mentales, comme la supervision ou l'application de procédure. Ils ont ensuite naturellement été adaptés pour l'informatique avec la "numérisation" des situations de travail.

Concrètement, les modèles de tâches offrent un formalisme permettant la description de l'activité nécessaire à une personne pour atteindre un but. Ils privilégient souvent une structure hiérarchique (but principal, buts intermédiaires, actions élémentaires) et sont le plus souvent représentés sous forme arborescente.

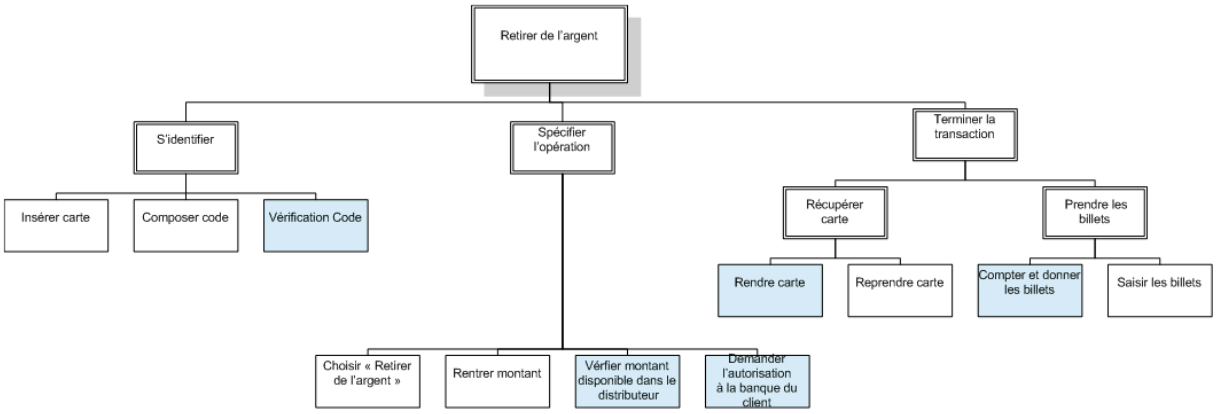


Figure 5. Exemple de modèle de tâche pour retirer de l'argent dans un distributeur automatique de billets

La Figure 5 décrit les tâches à réaliser pour retirer de l'argent à un distributeur automatique. Il se lit de haut en bas, puis de gauche à droite. Les éléments à double cadre représentent les buts intermédiaires à réaliser pour achever le but principal (la racine de l'arbre), les éléments à simple cadre les tâches élémentaires. Les tâches élémentaires bleutées décrivent les tâches que doit réaliser le système (ici, le DAB), les taches blanches, celles de l'utilisateur.

Les différents formalismes offrent une notation plus complexe, avec l'introduction d'opérateurs temporels ou d'ordonnancement (ISO, 1989) (permettant de décrire des tâches parallèles, des répétitions, des choix) ainsi que plus de types de tâches. On retrouve souvent le découpage tâche utilisateur, qui décrit les tâches mentales ou motrices, tâches d'interaction pour les actions de l'utilisateur sur le système et tâches système pour les traitements système. Les buts intermédiaires regroupant des sous-buts ou tâches de types différents sont dit "abstraits".

Les Modèles de tâche permettent de faciliter la compréhension des utilisateurs et les tâches qui sont effectuées dans un domaine donné. Ces modèles permettent l'identification et l'analyse de la conception de nouveaux besoins ainsi que les besoins de formation des utilisateurs (Johnson, Drake, & Wilson, A framework for integrating UIMS and user task models in the design of user interfaces., 1991). Ils permettent aussi d'examiner les connaissances nécessaires pour faire fonctionner un système (Hoppe, 1987).

Pour les systèmes critiques, l'analyse des tâches doit décrire les procédures pour le fonctionnement en situation normal, les procédures de maintenance du système et des procédures pour des situations d'urgence (Rajan & Redmill, 1997). La description des procédures de fonctionnement et d'entretien en situation normal doivent inclure des mesures de recouvrement par lequel les erreurs de l'utilisateur sont détectées et corrigées pour éviter un accident (Kirwan, 1994).

2 Domaine d'utilisation

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) (Kirwan & Ainsworth, 1992). Elle peut être utile comme outil de conception d'un système ou comme aide

à l'analyse de tâches. Dans le premier cas, le plus fréquent, le modèle a pour client le concepteur. Dans le second cas, il s'adresse au consultant en facteurs humains.

Les résultats de cette thèse s'adressent aux deux types d'utilisateurs des techniques de modélisation de tâches. Pour le premier type d'utilisateurs, offrir une notation qui répond à un certain nombre de critère de modélisation de tâches. Pour le deuxième type, offrir une notation et technique d'identification systématique des erreurs humaines.

2.1 Les concepteurs de systèmes interactives critiques (les usines chimiques, le contrôle de trafic aérien, les avions, les systèmes militaires)

Dans ce cas de systèmes, l'analyse de tâches est une exigence dans le processus de conception. L'analyse est utilisée comme une étape fondamentale dans l'intégration homme-système, par exemple l'évaluation des besoins de formation. Souvent, les résultats sont importés dans d'autres outils d'analyse, tels que des outils de modélisation et des simulations cognitives du système. Ces utilisateurs sont spécialistes des facteurs humains, et travaillent souvent dans des équipes multidisciplinaires avec des experts en la matière et spécialistes de la simulation.

2.2 Consultants en facteurs humains

Il s'agit d'un environnement plus varié, dans lequel les analystes appliquent leur méthode préférée pour la collecte des besoins et l'analyse des facteurs humains. Les projets peuvent exigent le respect du style interne du client, mais il est plus probable que le résultat final sera le thème du travail, plutôt que de la méthode utilisée. Vitesse d'apprentissage d'un outil d'analyse des tâches et de l'adaptation à une série de problèmes est essentielle. Analyses vont généralement de taille petite à moyenne. Les utilisateurs peuvent être décrits comme des facteurs humains généralistes.

3 Notations et outils de modélisation de tâches

Les techniques de modélisation des tâches fournissent un support pour structurer et stocker les informations recueillies à partir de l'analyse des tâches à travers leurs propres éléments de représentation tels que les arbres hiérarchiques, les organigrammes, etc. Les modèles résultants sont capables de capturer et de représenter certains aspects particuliers des travaux pertinents pour l'analyste qui sélectionne la technique de modélisation des tâches la plus adaptée à ses besoins.

Pour exécuter des tâches et atteindre un but, les utilisateurs peuvent avoir besoin de manipuler des objets ou des informations sur la situation actuelle du système et de son environnement, de savoir quelles actions exécuter et comment les exécuter. Lors de la conception de systèmes interactifs, la phase d'analyse des tâches se concentre généralement sur : l'identification des objectifs à atteindre, le regroupement des activités à accomplir, la compréhension de l'ordre d'exécution de ces activités et l'identification des objets nécessaires à l'exécution des tâches. Les techniques d'analyse et les notations existantes ne permettent pas de décrire explicitement

et distinctement les concepts liés aux notions d'objet, de connaissance et d'information. Bien que certains d'entre eux fournissent un support pour décrire les objets manipulés, la plupart de ces notations sont axées sur la représentation des procédures et des méthodes pour atteindre un objectif plutôt que sur les connaissances, les informations et les objets impliqués.

Dans cette section, nous allons faire une analyse détaillée d'un certain nombre de notation de modélisation de tâches, et mettre l'accent sur HAMSTERS (Human-centered Assessment and Modelling to Support Task Engineering for Resilient Systems), car grâce à un à un processus structuré, il permet de modéliser des activités humaines complexes et de représenter des informations et des objets. De plus, il s'inspire de la notation CTT (Concur Task Trees) (Paternò, 2004), et a donc été conçu pour rester compatible avec celui-ci. En outre, HAMSTERS implique des éléments de notation tels que les conditions associées aux exécutions de tâches, le flux de données à travers les modèles de tâche, etc., étendant sa puissance d'expression au-delà de celle de CTT. HAMSTERS est disponible au public, avec un simulateur de tâches et une API dédiée pour l'observation des événements d'édition et de simulation.

3.1 Un aperçu des techniques de modélisation des tâches et des méthodes associées

L'analyse et la modélisation de tâches remonte à un demi-siècle en arrière. L'une des raisons de sa longévité est que les techniques et méthodes de modélisation des tâches ont été adaptées à l'évolution des systèmes socio-techniques qu'elles sont censées décrire. Cela a donné naissance à plus d'une centaine de méthodes et de techniques d'analyse de tâches (Diaper & Stanton, The handbook of task analysis for human-computer interaction, 2003). La communauté scientifique a activement développé et amélioré le pouvoir expressif de l'analyse des tâches et des techniques de modélisation présentées dans (Caffiau, Scapin, Girard, Baron, & Jambon, 2010). Cette puissance expressive permet par exemple de mieux représenter et analyser les activités humaines tout en interagissant avec les nouvelles technologies (Jourde, Laurillau, & Nigay, 2010) ou de représenter explicitement les informations utilisées pour effectuer des tâches comme dans (Villaren, Coppin, & Leal, Modeling task transitions to help designing for better situation awareness., 2012). Nous fournissons ici après un aperçu des techniques représentatives de modélisation des tâches et de leurs méthodes connexes lorsqu'elles sont disponibles. Ils sont présentés dans l'ordre chronologique.

Le modèle HTA (Hierarchical Task Analysis) (Annett, 2003) (Annett & Duncan, 1967) a été proposé pour la première fois vers la fin des années 60 comme une approche générale pour l'analyse des tâches (Shepherd, 2014). La notation HTA structure les tâches sous la forme d'un arbre. La racine correspond au but de l'utilisateur, les nœuds partageant un même parent correspondent à la liste des sous-buts ou activités à atteindre pour réaliser le parent. L'enchaînement entre des tâches "sœurs" est décrit par un plan, rattaché au parent. Ce plan prend la forme d'un algorithme, ce qui laisse une grande liberté à l'analyste quant à l'organisation de son arbre (arbre peu profond et plans complexes, ou l'inverse). Les tâches sont numérotées (hiérarchiquement) ce qui permet de différencier facilement un sous arbre vis à vis d'un arbre global (utile dans le cas d'une répartition de l'arbre sur plusieurs feuilles de papier). La modélisation de tâches HTA est supporté par l'outil Task Architect (Stuart & Penn, 2004).

Le modèle GOMS (Goals, Operators, Methods and Selection rules) (Card, Newell, & Moran, 1983) (Card S. K., 2017) a pour objectif de faire de la prédiction de performance vis-à-vis d'un utilisateur face à un système particulier. GOMS est avant tout un modèle cognitif de l'utilisateur, la tâche en elle-même n'étant pas le concept central du modèle, en lieu et place de la méthode qui décrit comment s'accomplit la tâche. Le plus bas niveau dans un modèle de GOMS correspond à une tâche élémentaire précise (qu'on pourra évaluer en terme de performance). L'inconvénient d'un modèle GOMS est qu'il suppose implicitement que l'interface utilisateur existe déjà. Il ne s'agit pas d'une décomposition évidente des tâches mais plutôt d'une organisation hiérarchique des méthodes utilisées pour résoudre les tâches. GOMS est utile pour un très bas niveau dans la modélisation des tâches mais ne procure aucun apport dans la conception d'une interface utilisateur. L'objectif de GOMS concerne plutôt l'optimisation de la performance d'utilisation du système, via l'évaluation du temps d'accomplissement, les tâches mémorielles et d'apprentissage. Son principal intérêt réside donc dans l'évaluation de la performance humaine. Il existe plusieurs outils de support, mais le plus récent est le CAT-HCI (Williams K. E., 2005).

La méthode TKS (Task Knowledge Structure) (Johnson, Johnson, Waddington, & Shouls, 1988) repose sur l'hypothèse que les gens possèdent dans leur mémoire des structures de connaissances liées aux tâches. L'approche représente des connaissances déclaratives et procédurales pour décrire les rôles, les objectifs, les plans et les procédures composés d'actions et d'objets. Une TKS est associée à chaque tâche qu'un agent doit accomplir, ces tâches sont déterminées par le rôle que l'agent doit jouer. Un agent peut avoir plusieurs rôles, et un rôle peut être joué par plusieurs agents. Si des tâches peuvent sembler identiques, mais dont les rôles associés sont différents, alors elles ne le sont pas ; il y a une relation appelée « similaire » pour caractériser cette situation. La TKS détient des informations à propos du but, qui est un état de la situation que peut amener une tâche particulière. Un but particulier est accompli par une tâche particulière et est décomposé en sous-buts. Chaque sous-but correspond à une sous-tâche de la structure et inversement. Un même but peut être atteint par différents sous-buts, ce qui conduit à l'élaboration d'un plan, le concept de plan correspondant à un ensemble de sous-buts et procédures permettant d'accomplir un but précis. L'objectif de TKS est de permettre de générer un prototype d'interface utilisateur et de faciliter la conception en apportant un modèle de tâches plus détaillé, décrivant la hiérarchie des tâches, les objets utilisés et aussi les structures de savoir. Même si la connaissance fait partie de l'acronyme, elle n'est pas explicitement représentée dans les modèles qui peuvent être édités en utilisant l'outil ADEPT (Johnson, Wilson, Markopoulos, & Pycock, 1993).

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. Cette récente version est basé sur l'extension de MAD et appelé K-MAD (Kernel of Model for Activity Description) ou en français N-MDA (Noyau du Modèle de Description de l'Activité). K-MAD est un formalisme proposé par (Lucquiaud, 2005). Il constate que de nombreux modèles existent, qu'ils ont chacun leurs spécificités et utilités sur des points précis, et que jusqu'à présent, les

gens les utilisent pour profiter des avantages de chacun. Lucquiaud fait également le constat que les formalismes en eux-mêmes sont décrits avec relativement peu de précision (difficulté de trouver un méta-modèle, incohérence entre les différentes descriptions). Il base donc son analyse sur les outils (Tamot pour DIANE+, CTTE pour CTT, IMAD pour MAD et Euterpe pou GTA) dont les caractéristiques (attribut entier, booléen, ou textuel) sont plus aisées à extraire. Il déduit de cette analyse qu'il est impossible de convertir systématiquement les modèles d'un formalisme à un autre. Néanmoins, il retrouve un certain nombre d'aspects dans tous les outils qu'il étudie. Il propose donc de définir un noyau, qui aspire à devenir une référence, et d'étendre ce noyau via des extensions pour recouvrir des objectifs particuliers, actuellement couverts par différents formalismes. Il définit son noyau autour de 3 éléments : l'utilisateur, les évènements et les tâches. Il différencie l'ordonnancement local (optionnel, itératif, interruptible) de l'ordonnancement global (lien avec les autres tâches). Pour ce dernier, il utilise 7 opérateurs : séquentiel, parallèle, élémentaire, inconnu, alternatif, pas d'ordre et alphanumérique (laissé à la discrétion de l'analyste). La notation K-MAD est supporté par l'outil K-MADe (Baron, Lucquiaud, Autard, & Scapin, 2006).

UAN (User Action Notation) (Siochi & Hartson, 1989) (Hartson, Siochi, & Hix, 1990) a été conçu pour formaliser la communication entre les concepteurs de l'interface utilisateur (UI) et l'équipe de développement lors de la spécification et l'analyse des détails de technologie. Le concept central est l'abstraction de la tâche, y compris la structure hiérarchique et le séquençage. Les actions de l'utilisateur, le retour d'interface et les changements d'état sont utilisés pour construire la description de la tâche, qui peut ensuite être utilisée comme une action à des niveaux d'abstraction plus élevés. Cette évaluation fournit principalement un support à la modélisation formelle des tâches interactives.

GTA (Groupware Task Analysis) (Van Der Veer, Lenting, & Bergevoet, 1996) a été développé dans l'optique d'apporter un moyen de modéliser la complexité de tâches dans un environnement coopératif. GTA a été pensé à partir du point de vue « ethnographique » dans le sens où il est destiné à permettre la conception de systèmes coopératifs, et aussi à partir de l'activité théorique, ayant défini une distinction claire entre tâche et action. Ses concepts fondamentaux sont : tâche, rôle, objet, agent et évènement. Dans GTA, les tâches complexes sont décomposées en tâches unitaires et tâches de base mais il n'y a pas de relations évidentes par rapport à l'interface utilisateur. L'intérêt de GTA est qu'il permet la représentation de tâches coopératives (groupware) en intégrant le concept de rôle dans le modèle, modèle qui permet alors d'identifier pour une tâche quel rôle en est responsable, mais aussi l'apport dit organisationnel tel que définir les rôles attribués aux différents agents. Néanmoins, certains aspects de GTA font qu'il n'est pas entièrement favorable à la conception d'interfaces utilisateur. Par exemple, les buts et actions sont définis comme des attributs de tâches, et non comme des concepts. GTA permet qu'un but puisse être atteint de nombreuses manières différentes. Aussi, une même action pouvant être appliquée à plusieurs tâches, il aurait été préférable que les actions soient représentées comme des concepts. Ainsi la manipulation d'objets est représentée comme une relation entre actions et objets au lieu de tâches et objets. Il décrit uniquement les connaissances procédurales. Euterpe est l'outil d'aide à la modélisation (van Welie, Van der Veer, & Eliëns, 1998).

Diane + (Tarby & Barthet, 1996) Créé dans le but de résoudre les problèmes des logiciels interactifs (apprentissage, mémorisation), DIANE+ est à la fois un formalisme pour décrire la modélisation de tâches et la méthode pour réaliser cette analyse. DIANE+ couvre toute la phase de spécification. Il repose sur trois représentations de l'application : conceptuelle pour l'analyste, externe pour l'utilisateur, interne pour le programmeur. Pour chaque opération ou tâche, on se pose les questions : qui la déclenche ? Qui la réalise ? Qui vérifie sa réalisation ? Qui contrôle son déroulement ? Le formalisme est basé sur trois types de tâches (automatique, interactive et manuelle) et permet de mettre des contraintes sur ces dernières. On peut subdiviser les tâches, gérer leurs contraintes, poser des pré / post-conditions ou encore des déclencheurs. L'ordre de déroulement des tâches peut être : séquentiel (ordonné ou pas), en boucle ; proposer un choix (libre ou contraint) ou en parallèle. Certaines tâches peuvent même être automatiques. Il est soutenu par plusieurs outils, tels que TAMOT (Paris, Tarby, & Vander Linden, 2001).

VTML (Visual Task Modeling Language) (Brown & Leveson, 1998) est une notation graphique conçue pour analyser formellement les actions d'un opérateur humain en relation avec des problèmes de sécurité. Elle propose d'enregistrer les actions des opérateurs sous la forme d'organigrammes. Il n'y a aucune indication sur la représentation des connaissances. VTML est soutenu par SpecTRM (Lee, Howard, & Anderson, 2002).

CWA (Cognitive Work Analysis) (Vicente, 1999) est un cadre conceptuel centré sur le travail, qui guide la conception de la technologie à utiliser sur le lieu de travail. Il se compose de 5 phases avec une approche dédiée pour chacun: 1) Analyse du domaine de travail (WDA) avec table d'abstraction et de décomposition, 2) Analyse des tâches de contrôle (CTA) supportée par échelle de décision, 3) analyse des stratégies avec des flux d'information, 4) l'analyse de l'organisation sociale et de la coopération dans laquelle vous pouvez utiliser toutes les approches ci-dessus, 5) l'analyse des compétences des travailleurs basée sur la compétence, les règles et les connaissances (Rasmussen, Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models, 1983). CWA n'est pas pris en charge par un outil.

TAKD (Task Analysis for Knowledge Descriptions) (Diaper, 2001) est une méthode qui est d'abord utilisée pour générer des descriptions de tâches, puis pour réexprimer les descriptions en termes de connaissances. Les descriptions de connaissances résultantes sont constituées de paires action / objet qui, lorsqu'elles sont combinées, représentent le contenu de connaissances des tâches. Cette méthode est liée à TKS. C'est une représentation sommaire des différents types de connaissances qui sont utilisés dans le comportement des tâches (Johnson, Johnson, Waddington, & Shouls, 1988), et il est soutenu par LUTAKD (Diaper, 2001).

CTT (Concurrent Task Tree) (Paternò, 2004) (Paternò, Mancini, & Meniconi, 1997) est un formalisme qui décrit un modèle de tâches de manière arborescente. Son avantage est sa représentation visuelle relativement aisée à comprendre. Les tâches sont organisées de façon hiérarchique et sont différenciées selon quatre critères : abstraites, interactives, cognitives et systèmes. Un opérateur de relation temporelle (ISO, 1989) est associé d'une tâche à une autre sur un même niveau d'arborescence permettant d'exprimer le déroulement de l'activité (tâches séquentielles, optionnelles, récursives, parallèles, …) contrairement aux autres modèles où les

opérateurs n'étaient définis que pour des relations mère filles. Cette représentation des opérateurs temporels présente l'avantage de pouvoir parcourir l'arbre selon le sens classique de lecture. Toutefois, les priorités implicites entre opérateurs peuvent créer une importante distance d'évaluation. CTT permet aussi de décrire des tâches coopératives : le modèle de tâches étant décomposé en plusieurs arbres de tâches, un pour chaque rôle, et un autre pour la coopération qui permet de décrire les aspects dynamiques entre des tâches de rôles (et donc modèles) différents. Une tâche dans CTT peut être décomposée en sous-tâches, jusqu'à atteindre un niveau de tâche élémentaire « basique ». La notation CTT est supporté par un outil appelé CTTe (Mori, Paternò, & Santoro, 2002), ainsi par un autre outil web, avec lequel les utilisateurs peuvent l'utiliser sur smartphone, tablette et pc, et de manière collaborative (Manca, Paternò, & Santoro, 2016).

AMBOSS (Giese, Mistrzyk, Pfau, & Von Detten, 2008) est un environnement de modélisation de 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. Un outil logiciel semble être développé, mais pas disponible.

La notation COMM (COllaborative and MultiModal) (Jourde, Laurillau, & Nigay, 2010) est une notation à base d'arbre de tâches. Elle permet la description de l'interaction de système multi-utilisateurs (Description de l'aspect coopératif et collaboratif) et multimodal. La notation se présente comme une extension de CTT (Paternò, Mancini, & Meniconi, 1997) dont elle étend les opérateurs temporels, et l'introduction de la description des rôles interactifs et des tâches multimodales. La particularité de la notation COMM est la possibilité de décrire des activité individuelles ou collaborative dans le même modèle de tâches. La notation COMM est outillée par l'éditeur e-COMM (Jourde, Laurillau, & Nigay, 2010), disponible sous la forme d'une application web.

SAMANTA (Situation Awareness Modeling and ANalysis for Transition Amelioration) (Villaren, Coppin, & Leal, 2012) est une méthodologie de conception d'interfaces homme-machine dans des systèmes dynamique prenant en compte les éléments de situation constituant l'activité des utilisateurs. Aussi, cette méthodologie repose sur l'usage conjoint de modèles couplant modèle de tâches et de description de contexte sous la forme de réseaux d'éléments de situation. La modélisation de tâches basée sur la notation CTT (Fabio, Breedvelt-Schouten, & de Koning, 1999). Le modèle d'éléments de situation est un graphe cartographiant des connaissances liées aux situations de réalisation de chaque tâche du modèle de tâche. SAMANTA est la seule notation dans laquelle il y a une tentative de proposer la production d'un modèle spécifique de connaissance au moyen d'un modèle appelé contexte. Il identifie ainsi clairement les connaissances nécessaires à l'exécution des tâches mais ne les sépare pas en plusieurs catégories.

3.2 État de l'existant de la notation et l'outil Hamsters

Ce chapitre présente l'état de la notation et l'outil Hamsters au moment du début de la thèse. Sachant que j'ai contribué à certaines extensions de la notation et de l'outil avant le début de la thèse, pendant mes trois stages du L3 au M2 effectué au sein de l'équipe ICS (Interactive Critical System), les extensions auxquelles j'ai participé seront mentionnées.

L'élaboration de la notation et l'outil HAMSTERS a été entamée en 2009 par trois étudiants en master2 IHM (André, Azzouzi, & Hoarau, 2009), dans le cadre de leur Chef d'œuvre. Elle a été poursuivie, par la suite, par Célia MARTINIE et Eric BARBONI de l'IRIT, qui ont commencé à développer l'outil et apporter des extensions à l'outil, et depuis Avril 2012 je participe à l'extension et au développement de la notation et l'outil. Cet outil a pour objectif l'édition et la simulation de modèles de tâches et la production de scénarios d'utilisation associés à ces modèles.

HAMSTERS (Human-centered Assessment and Modeling to Support Task Engineering for Resilient Systems) utilise certains mécanismes de notations de modélisation de tâches existantes. Il 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. Plus particulièrement, il utilise les opérateurs temporels LOTOS (ISO, 1989), utilisés aussi par la notation CTT (Paternò, Mancini, & Meniconi, 1997). La notation HAMSTERS offre une représentation graphique et possède un outil logiciel associé HAMSTERS permettant d'éditer, de classer et de simuler des modèles de tâches.

3.2.1 La notation Hamsters

3.2.1.1 Types de tâches

Les types de tâches permettent la description des activités de l'utilisateur en décrivant la nature de la tâche. Plus le type de tâche est raffiné, plus l'analyse des interactions entre l'utilisateur et le système, ou de l'activité humaine devient plus enrichi. La première sous-section présente les tâches génériques, puis la deuxième et troisième sous-sections présentent les raffinements des tâches interactives et des tâches utilisateurs.

Présentation générale

La notion de tâche est générique, elle peut être un but, un sous but ou bien une action utilisateur. Ainsi une tâche peut être un but et se décompose en un sous-ensemble de tâches. Le Tableau 1 présente les types de tâches génériques fournis par la notation Hamsters et leur représentation.

Tableau 1. Types de tâches générique de la notation Hamsters

Types de tâche	Représentation Hamsters
Tâche abstraite	
Tâche abstraite utilisateur	
Tâche abstraite système	
Tâche abstraite interactive	
Tâche utilisateur	
Tâches système	

Une **tâche abstraite** permet de décrire un but ou une sous-tâche rassemblant des sous-tâches de types différents.

Une **tâche abstraite utilisateur** permet de décrire un but rassemblant des sous tâches de types utilisateur.

Une **tâche abstraite système** permet de décrire un but rassemblant des sous tâches de types système.

Une **tâche abstraite interactive** permet de décrire un but rassemblant des sous tâches ou action de types interactive seules, ou un ensemble d'action interactive et utilisateur.

Une **tâche utilisateur** permet de décrire une sous tâche rassemblant des actions utilisateur.

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

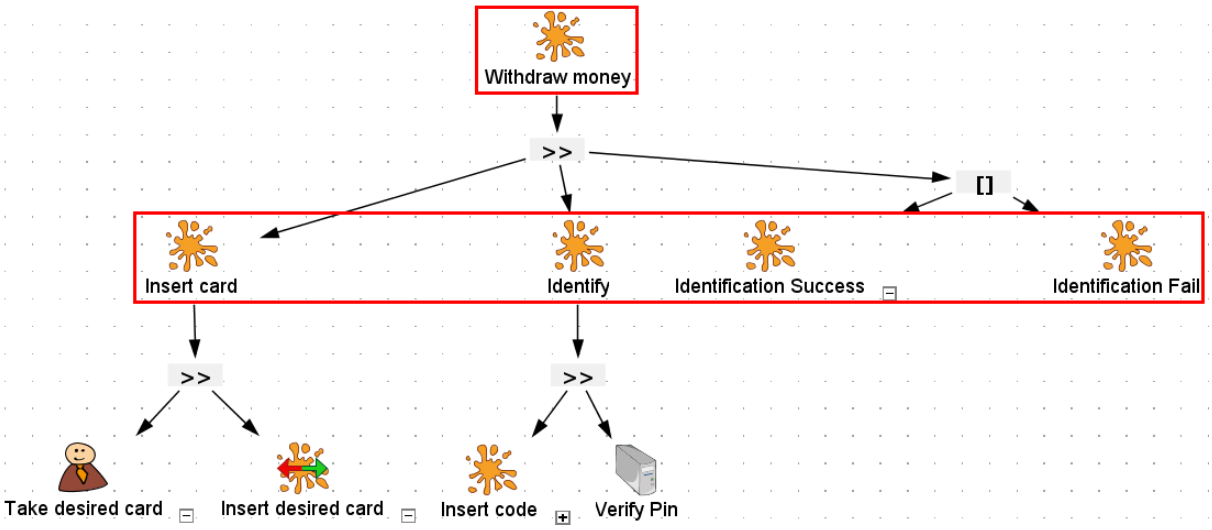


Figure 6. Exemple illustratif du distributeur de billets : modèle de tâches abstrait

La Figure 6 présente le modèle de tâches abstrait et très peu détaillé de l'utilisation d'un distributeur automatique de billet (DAB). La tâche abstraite « **Withdraw money** » (*Retirer de l'argent*) est la séquence des sous tâches « **Insert card** » (*insérer la carte*), « **Identify** » (*Identification*) et le choix entre une sous tâche où *l'identification est réussi* « **Identification Success** » et une sous tâche où *l'identification n'est pas réussi* « **Identification Fail** ».

Raffinement des tâches utilisateur

Le raffinement des tâches interactives permet de décrire la manière dont le flux d'information transite entre l'utilisateur et le système. Le Tableau 2 présente les types de tâches interactive fournis par la notation Hamsters et leur représentation.

Tableau 2. Types de tâches interactives de la notation Hamsters

Types de tâche	Représentation Hamsters
Tâche interactive d'entrée	 Tâche interactive d'entrée
Tâche interactive de sortie	 Tâche interactive de sortie
Tâche interactive d'entrée/sortie	 Tâche interactive d'entrée/sortie

Une **tâche interactive d'entrée** permet de décrire que l'utilisateur agit sur le système en lui fournissant un flux d'information en entrée, exemple : l'utilisateur envoie une information au système en appuyant sur une touche du clavier.

Une **tâche interactive de sortie** permet de décrire que l'utilisateur reçoit un flux d'information sortant du système, exemple : le système affiche une information à l'utilisateur sur un écran.

Une **tâche interactive d'entrée/sortie** permet de décrire que l'utilisateur envoie et reçoit simultanément des flux d'information avec le système, exemple : l'utilisateur scroll une page web à l'aide de la molette de la souris, et en même temps la page web défile.

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 7 présente une version un peu plus détaillée du modèle de tâches de l'exemple du retrait d'argent de la Figure 6. Dans cette version, le sous arbre de la tâche « **Insert desired card** » (*insérer la carte désirée*) est détaillé en une séquence de deux tâches interactives :

- La tâche interactive d'entrée « **Insert card** » (*insérer la carte*) permet de décrire le passage d'un objet physique de l'utilisateur vers le système du DAB.
- La tâche interactive de sortie « **Ask for PIN code** » (*demande du code*) permet de décrire que le DAB affiche à l'utilisateur sur l'écran qu'il est en attente d'une information de la part de l'utilisateur.

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 interactives en entrée sur le système et de concevoir le système différemment afin d'optimiser ces actions, en automatisant certaines tâches. Cet exemple a fait l'objet d'une publication et est détaillé dans (Martinie C. , Palanque, Barboni, & Ragosta, 2011).

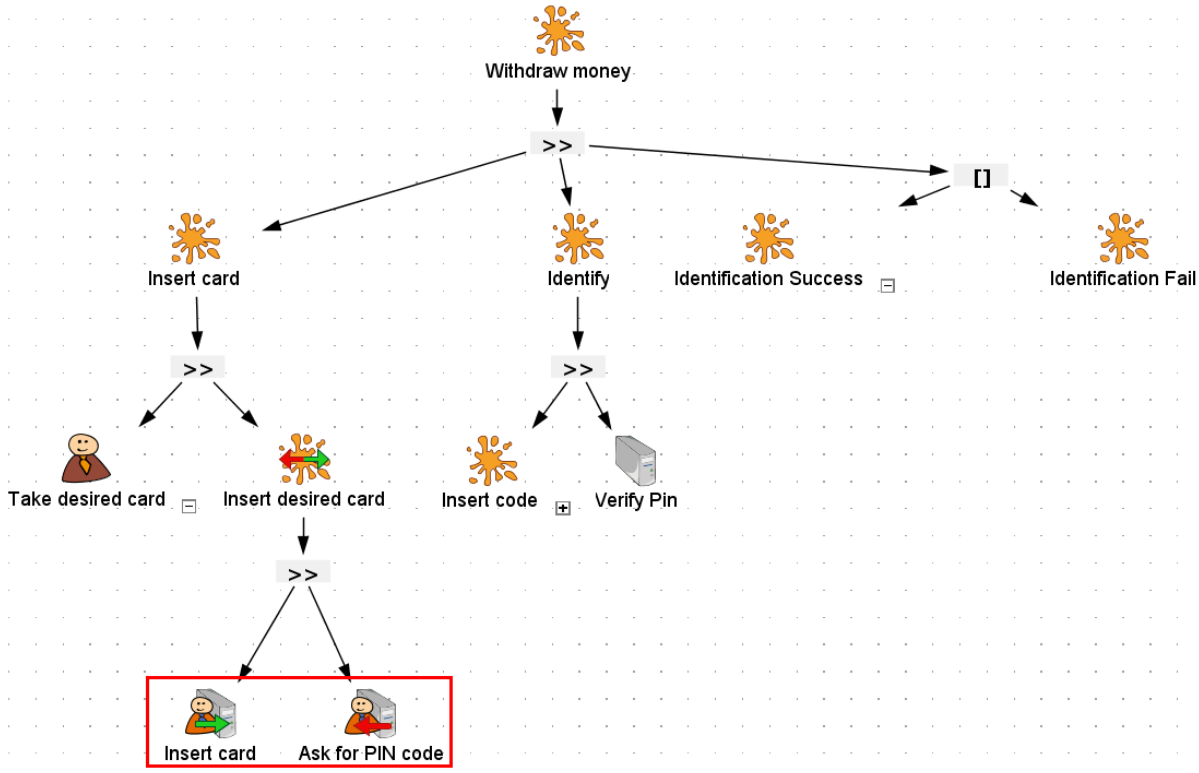


Figure 7. Exemple illustratif du distributeur de billets : raffinement de la tâche "Insert desired card" (Insérer la carte désirée) avec les sous-tâches interactives

Raffinement des tâches interactives

Le raffinement des tâches utilisateur permet de prendre en compte les déférents aspects d'une activé réalisée par un utilisateur. Les tâches utilisateurs peuvent être de nature différentes selon la manière dont l'utilisateur réalise sa tâche.

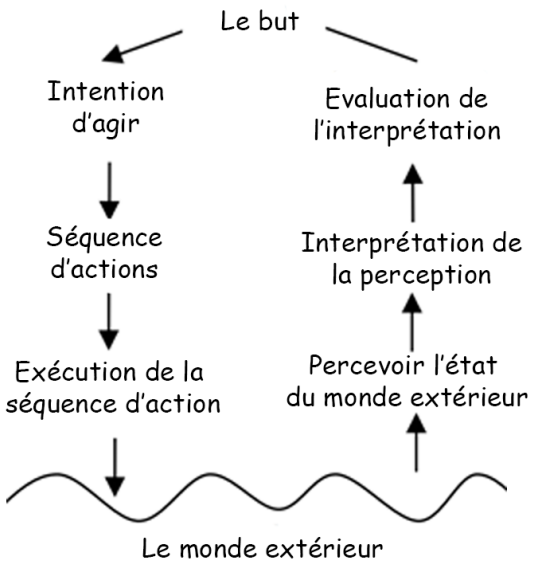


Figure 8. Les sept étapes de la théorie de l'action (Norman D. , 2002)

La Figure 8 décrit les sept étape de la théorie de l'action proposée par (Norman & Draper, 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 humaines de types différents. L'utilisateur établit un but puis une intention d'agir, puis il détermine la séquence d'action à réaliser et il l'exécute. Ensuite, il perçoit l'état actuel du système, puis il interprète sa perception et évalue son interprétation 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, The Psychology of Human-Computer Interaction, 1983) proposent un modèle du processus humain qui représente les différents types d'activités humaines selon les unités de traitement suivantes : processus moteur, processus perceptif et processus cognitif. La notation Hamsters permet la description de ces trois types d'activités humaines à l'aide des tâches représentées dans le Tableau 3.

Tableau 3. Types de tâches utilisateur de la notation Hamsters

Types de tâche	Représentation Hamsters
Tâche motrice	
Tâche perceptive	
Tâche cognitive	

Une tâche motrice permet de décrire un mouvement physique de la part de l'utilisateur. Exemple : appuyer sur la pédale de frein.

Une tâche perceptive permet de décrire une action sensorielle (toucher, voir, sentir, gouter, entendre, …) de récupération d'information extérieures. Exemple : regarder sur le rétroviseur.

Une tâche cognitive permet de décrire un traitement de l'information. Exemple : analyser s'il est possible de doubler une voiture.

La Figure 9 présente une nouvelle version du modèle de tâches de l'exemple illustratif du retrait d'argent au DAB. Dans cette nouvelle version, le sous arbre de la tâche « **Take desired card** » (*prendre la carte désirée*) est détaillé en une séquence de trois tâches utilisateurs :

La tâche cognitive « **Identify desired card** » (*identifier la carte désirée*) permet de décrire que l'utilisateur doit effectuer une tâche cognitive pour choisir la carte qu'il désire.

La tâche perceptive « **perceive the desired card** » (*percevoir la carte désirée*) permet de décrire que l'utilisateur doit percevoir la carte qu'il veut utiliser avant de la prendre.

La tâche motrice « **Grab desired card from wallet** » (*prendre la carte désirée du portefeuille*) permet de décrire que l'utilisateur doit effectuer une action physique en utilisant sa main et ses doigts, afin de retirer la carte de son portefeuille.

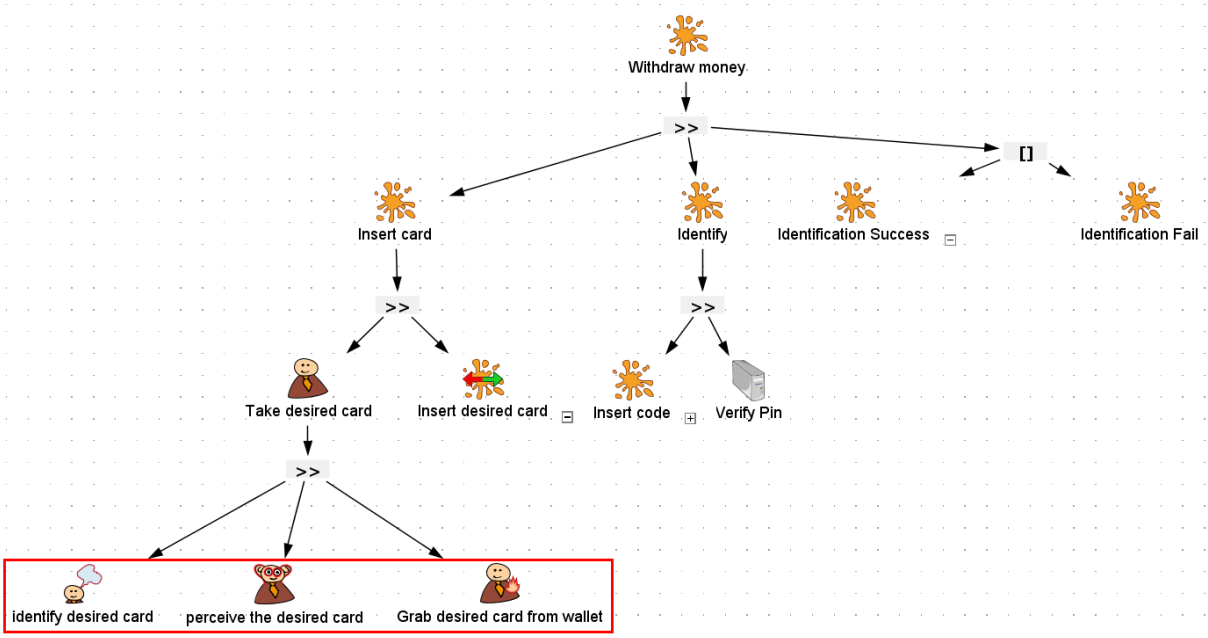


Figure 9. Exemple illustratif du distributeur de billets : raffinement de la tâche "Take desired card" (Prendre la carte désirée) avec les sous-tâches utilisateurs

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 ou plusieurs activités au système.

Hamsters fournit les moyens de raffiner une tâche cognitive en deux sous-tâches, voir le Tableau 4.

Tableau 4. Types de tâches cognitives de la notation Hamsters

Types de tâche	Représentation Hamsters
Tâche d'analyse	
Tâche de décision	

Une **tâche d'analyse cognitive** permet de décrire que l'utilisateur analyse une situation ou une information.

Une **tâche de décision cognitive** permet de décrire que l'utilisateur prend une décision en rapport avec une situation.

La Figure 10 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).

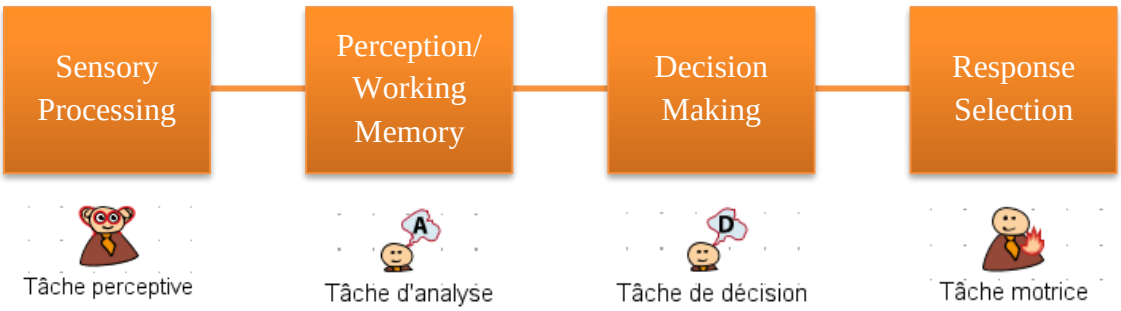


Figure 10. Répartition des sous-types de tâches utilisateur selon le modèle humain du traitement de l'information de Parasuraman

3.2.1.2 Structuration des modèles de tâches

Cette sous-section présente les différents mécanismes proposé dans la notation HAMSTERS pour structurer des modèles de tâches de systèmes complexes.

Hiérarchie

La notation de modélisation de tâches HAMSTERS se base sur la notation HTA, en offrant une représentation hiérarchique. Le but principal se décompose en sous buts, et les sous buts à leur tour se décomposent en d'autres sous buts, ou actions élémentaires. On peut définir cette notation comme un arbre avec une racine qui est le but, les nœuds qui sont les sous buts et les feuilles comme action élémentaire. La notation HAMSTERS ce lit de haut en bas, puis de gauche à droite.

Ordonnancement temporel

Les relations temporelles qui décrivent l'ordonnancement des sous buts d'un but, 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 permettant de faciliter l'ordonnancement des tâches. Le détails de ces opérateurs temporels est présenté dans l'Annexe 2 : Ordonnancement temporel.

3.2.1.3 Structuration de tâches complexes et nombreuses

Dans le cas de systèmes interactifs complexes, la modélisation des tâches utilisateur nécessite des moyens supplémentaires.

La notation HAMSTERS fournit les éléments de notation Sub-model (Copy Task) et Sub-routine. Ces termes font référence aux éléments de programmation structurée employés par un grand nombre de langages informatiques. Ils fournissent un support à modularité et à la réutilisabilité des modèles de tâche.

Sub-model :

Le mécanisme de *Sub-model ou copy-task* 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 que les tâches existantes avec une icône « COPY » en superposition sur la tâche (voir exemple tâche abstraite Figure 11), selon chaque type de tâche.

Figure 11. Exemple de copy-task d'une tâche abstraite

Sub-routine :

Le mécanisme de Sub-routine 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 Sub-routine:

- 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 Tableau 5décrit les différents types de Sub-routine selon leur capacité à recevoir et fournir des paramètres d'entrée/sortie.

Tableau 5. Types de Sub-routine dans la notation HAMSTERS

Type de Sub-routine	Représentation avec la notation HAMSTERS
Sub-routine sans aucun flux de données entrant/sortant	
Sub-routine avec un ou plusieurs flux de données entrant(s)	
Sub-routine avec un ou plusieurs flux de données sortant(s)	
Sub-routine avec un ou plusieurs flux de données entrant(s) et sortant(s)	

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

3.2.1.4 Représentation des données et leur traitement

Le pouvoir expressif de HAMSTERS va au-delà de la plupart des autres notations de modélisation de tâches en particulier en fournissant des moyens détaillés pour décrire les données requises et manipulées pour accomplir des tâches. La Figure 12 résume les éléments de notation pour représenter les données. L'information ("Inf:" suivie d'une zone de texte) peut être requise pour l'exécution d'une tâche système. Les objets physiques nécessaires à l'exécution d'une tâche peuvent également être représentés ("Phy O") ainsi que le périphérique (entrée et / ou sortie) avec lequel la tâche est exécutée ("i / o D"). Les connaissances déclaratives et situationnelles peuvent également être explicitées par les éléments «SiK» et «StK».

Figure 12. Représentation des Données, objets et périphériques dans HAMSTERS

Objet (générique + lié à l'implémentation) : permet d'illustrer n'importe quel type d'objet nécessaire ou produit à l'accomplissement d'une tâche ou n'ayant pas encore de type défini.

Information : permet de décrire un élément de connaissance (donnée, image, voix, …) nécessaire à l'exécution d'une tâche ou susceptible d'être produite après l'accomplissement d'une tâche.

Connaissance (de type déclaratif et/ou procédural) : HAMSTERS permet de représenter les connaissances déclaratives et procédurales nécessaires qui se réfèrent respectivement à ce que l'acteur connait et à des habiletés que l'acteur sait exécuter. Cet objet peut être raffiné en deux sous-types : stratégique et situationnel.

- ***Connaissance déclarative***

Hamsters permet de décrire le savoir nécessaire à l'accomplissement d'une tâche.

Tableau 6. Représentation des types de connaissance avec la notation HAMSTERS

Type de connaissance	Représentation HAMSTERS
Connaissance déclarative	DK: Declarative knowledge
Connaissance déclarative stratégique	StK: Strategic knowledge
Connaissance déclarative situationnelle	SiK: Situational Knowledge

Le Tableau 6 représente les différentes présentations de types connaissance ; la connaissance déclarative est raffinée en deux types, stratégique et situationnel.

- ***Connaissance procédurale***

Un modèle de tâche HAMSTERS doit permettre de décrire une activité pour atteindre un but (ceci est une information de type « procédural »).

Il permet donc de décrire des connaissances procédurales (procedural knowledge) nécessaires à l'accomplissement d'une tâche :

- De type Stratégique (voir Figure 13)

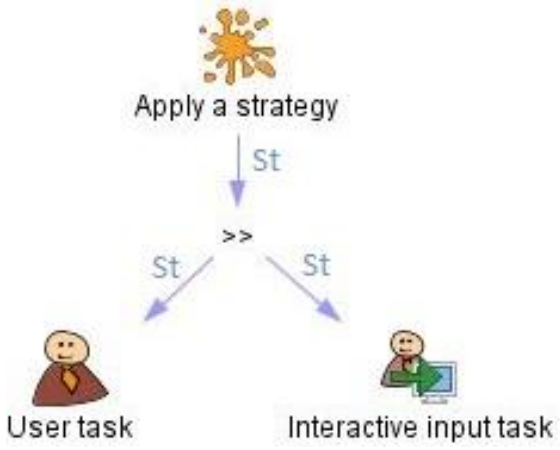


Figure 13.Représentation d'une connaissance procédurale de type stratégique avec la notation HAMSTERS

- De type Situationnel (voir Figure 14)

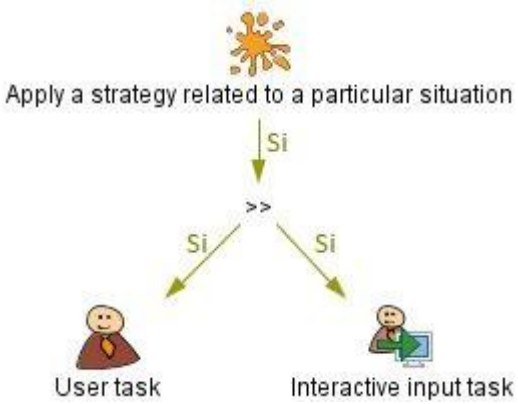


Figure 14.Représentation d'une connaissance procédurale de type situationnel avec la notation HAMSTERS

Application : permet de représenter une application liée ou générée à l'exécution d'une tâche.

Périphérique (device) : permet de représenter un périphérique utilisé pour exécuter une tâche interactive.

3.2.1.5 Description des tâches collaboratives

La notation de modélisation de tâches HAMSTERS, permet aussi de décrire des tâches collaboratives. Une tâche de groupe permet de décrire un ensemble d'acteurs agissant en interaction les uns avec les autres, à la poursuite d'un but commun ; le principe est que les acteurs du même groupe ne peuvent poursuivre leur tâche sans la participation active de chacun d'eux. Cette tâche collaborative peut être raffinée en plusieurs types de tâches, l'Annexe 1 : Raffinement des tâches collaboratives.

3.2.1.6 Synthèse des éléments de la notation Hamsters

3.2.2 L'outil 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 cette section.

3.2.2.1 Edition de modèle de tâches

L'outil HAMSTERS présenté dans la Tableau 14 offre une perspective d'édition de modèles de tâches, comprenant :

- Un navigateur de projets avec ses modèles de tâches et scénarios associés (numéro 1).
- Un cadre central permettant d'éditer et consulter plusieurs modèles de tâches à la fois (numéro 2).
- Une palette d'édition permettant de glisser/déposer les éléments de la notation (numéro 3).
- Une fenêtre d'édition des propriétés liées à un élément de la notation (numéro 4).
- Un navigateur de structure du modèle de tâche couramment édité (numéro 5).

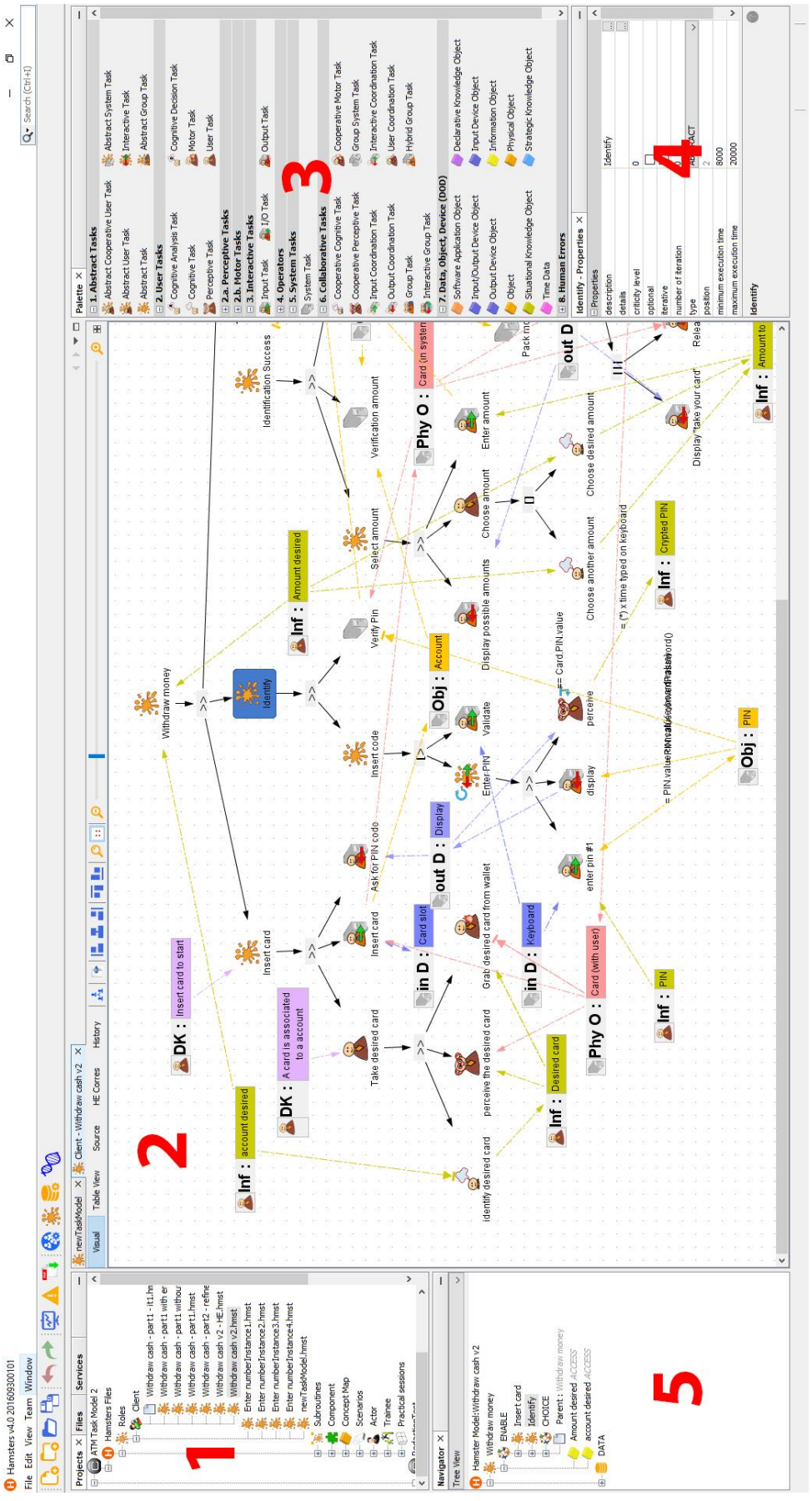


Figure 15. Vue globale de l'éditeur de tâches HAMSTERS

3.2.2.2 Simulation de modèle de tâches

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

La Figure 16 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, numéro 1) et la liste des tâches déjà effectuées dans leur ordre d'exécution avec le modèle auquel elles appartiennent (liste dans la partie inférieure, numéro 3). La zone centrale (numéro 2) permet au concepteur de saisir des données contextuelles sur l'exécution d'une tâche. Par exemple, 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 ou non. 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 projet, celle contenant la liste des modèles de tâches du projet, on peut définir un modèle de tâches principal, qui permet de montrer que c'est ce modèle qu'il faut exécuter en simulation. Pour le faire on fait un clic droit sur le modèle souhaité, et on sélectionne « Set as Main Role Model », l'icône du modèle de tâche passe de  à  .

Dans la partie édition 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 

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

.

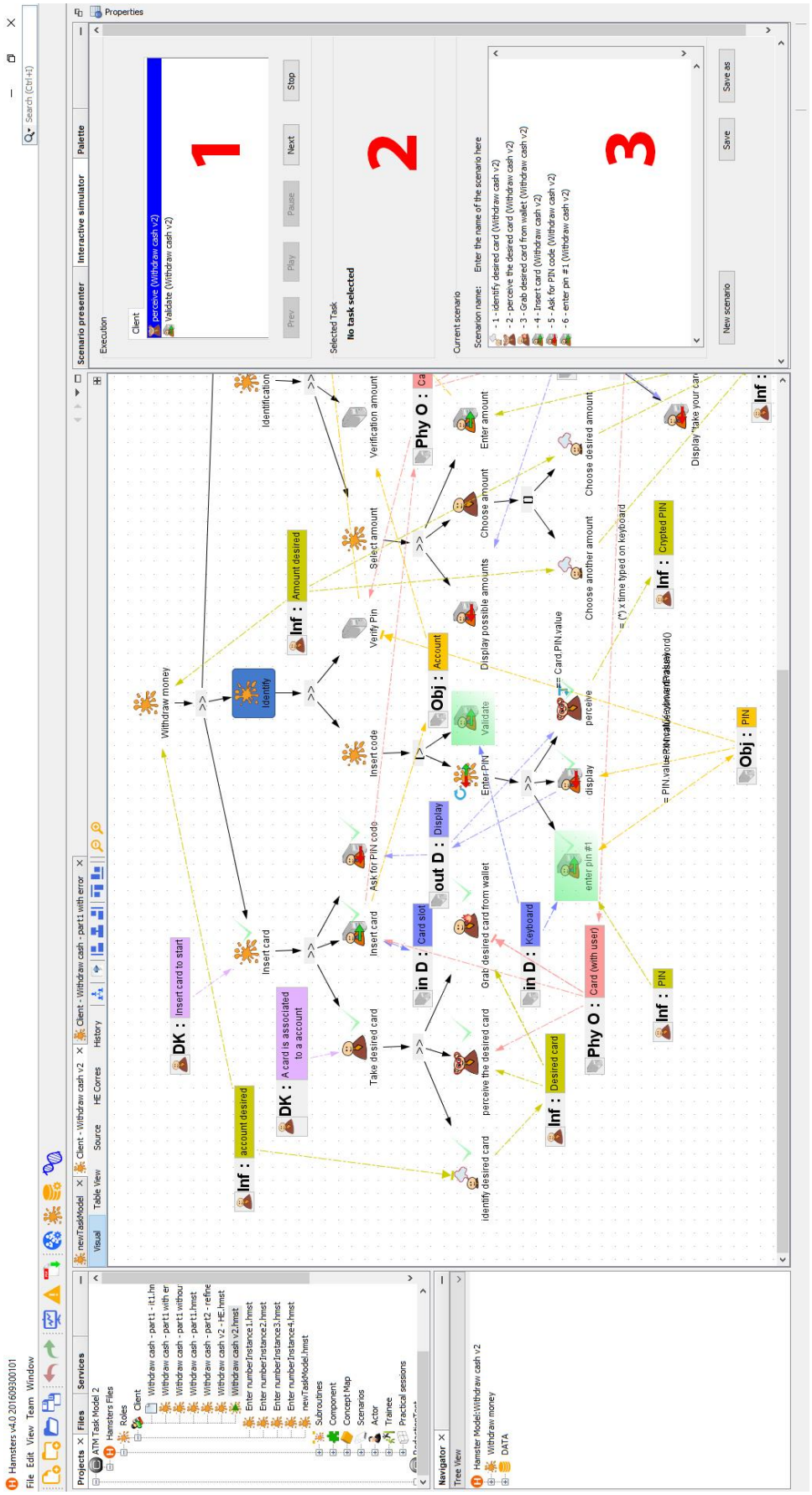


Figure 16. Copie d'écran d'une simulation de modèles de tâches avec l'outil logiciel HAMSTERS (exemple de tâches effectuées et actives)

3.3 Comparaison des techniques de modélisation des tâches et des méthodes associées selon les exigences identifiées

Le Tableau 7 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.

Tableau 7. Comparaison d'un échantillon représentatif de notations de modélisation de tâches utilisateurs

		HTA	GOMS	TKS	MAD	UAN	GTA	DIANE+	VTML	CWA	TAKD	CTT	AMBOSS	COMM	SAMANT	HAMSTE
Data	Description des connaissance			x						x					x	x
	Description des périphérique manipulés															x
	Description des informations										x	x			x	x
	Toutes les informations ci-dessus doivent être intégrées dans un seul modèle															x
Raffinement de tâche	Tâche motrice															x
	Tâche cognitive													x		x
	Tâche interactive											x		x	x	x
	Tâche perceptive															x
Description d'une tâche	Criticité de la tâche															x
	Temps max d'exécution de la tâche															x
	Temps min d'exécution de la tâche															x
Collaboration	Description des tâches collaboratives											x		x	x	x
	Description des rôles											x		x	x	x
Notation	Textuelle	x	x	x		x			x		x					
	Graphique	x		x	x		x	x	x		x	x	x	x	x	x
	Décomposition Hiérarchique	x	x	x	x	x	x	x			x	x	x	x	x	x
	Relation temporelles entre les tâches		x	x	x	x	x	x	x			x	x	x	x	x
Structuration	Description des subroutine															x
	Description des composant															
Erreur	Description des erreurs humaines															
	Représentation explicite des erreurs humaines															
	description de la récupération de l'erreur humaine	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
	représentation explicite de la récupération de l'erreur															
	Représentation des propriétés humaine requise à l'exécution d'une tâche															
Contexte	Description du contexte d'exécution de la tâche															
Outil	Outillé	Task-Architect	CAT-HCI	ADEPT	K-MADe	Quantum	EUTERPE	Isolde	SpecTRM		LUTAKD	CTTe	AMBOSS	eCOMM	SAMANTA	HAMSTERS

Le Tableau 7 nous permet de montrer que la majeure partie des notations permettent la décomposition hiérarchique des tâches, l'ordonnancement temporel des tâches, la représentation graphique des tâches et la description de tâche de reprise d'erreur. Cependant elle met aussi en évidence les points suivants :

- La possibilité de structurer les modèles, en utilisant des composants, lorsque le nombre de tâches est considérable, n'est fournie par aucune des notations existantes.
- Seul la notation HAMSTERS permets de décrire des tâches utilisateurs raffinées (Motrice, perceptive), et pour les tâches cognitives, COMM et HAMSTERS le permettent.
- Aucune notation ne permet de représenter explicitement les erreurs humaines et la reprise d'erreur.
- En revanche, toutes les notations de modélisation de tâches sont supportées par un outil, sauf CWA.

4 Conclusion

Ce chapitre a présenté de façon structurée et systématique les différentes contributions dans le domaine de la modélisation des tâches et buts utilisateurs. Nous avons en particulier présenté la notation HAMSTERS (dans son état au début de ce travail de thèse) ainsi que son outil éponyme. Au-delà du fait que cette notation a été développée dans mon équipe de recherche, nous avons, grâce à une comparaison fondée sur un ensemble de critères, justifié ce choix car cette notation est la plus adaptée à des extensions permettant la pris en compte des erreurs humaines. Cette prise en compte des erreurs humaines nécessite une étude des différents travaux portant sur les différents types d'erreurs humaines ce qui fait l'objet du chapitre suivant.

Chapitre 2. – Taxonomies des erreurs humaines

La technologie actuelle est caractérisée par la complexité, le changement rapide et la taille croissante des systèmes techniques. Cela a provoqué une inquiétude croissante à l'intervention humaine dans la sécurité du système. Les analyses des accidents majeurs au cours des dernières décennies ont conclu que des erreurs humaines sur des opérateurs, concepteurs ou gestionnaires ont joué des rôles majeurs (Rasmussen, 1990). Ces erreurs sont la source d'importantes pertes de production, et même de catastrophes majeures, telles que la collision sur la piste de l'aéroport de Ténériffe en 1977 (McCreary, Pollard, Stevenson, & Wilson, 1998), l'accident de la centrale nucléaire de Three Mile Island en 1979 (Ragheb, 2001), les catastrophes de Challenger et Tchernobyl en 1986 (Expert report to the IAEA on the Chernobyl accident 61, 1986), le naufrage du Herald of Free Enterprise en 1987 (Praetorius, Lundh, & Lützhöft, 2011), et l'explosion de la plate-forme pétrolière Piper Alpha en 1988 au large de l'Ecosse (Paté-Cornell, 1993). Pour cela, l'analyse du coût de l'erreur humaine est au centre des préoccupations des équipes en charge de la sécurité des systèmes industriels (Reason, 1990) et des concepteurs de systèmes critiques.

1 Définitions de l'erreur humaine

On considère comme « erreur humaine », tout comportement humain ne respectant pas le bon usage et pouvant conduire de façon involontaire à des préjudices divers. Comme l'affirme Leplat (Leplat, 1985), l'erreur a indéniablement un aspect négatif puisqu'elle témoigne, pour l'observateur, que l'opérateur n'a pas répondu aux exigences de la tâche qu'il avait à exécuter, qu'il n'a pas fait ce qui était attendu de lui.

La question au centre des débats concerne bien souvent la responsabilité humaine, celle de l'opérateur en l'occurrence. Pourtant, les nombreuses recherches sur l'erreur humaine, sur la fiabilité humaine, sur les accidents du travail, ont mis en évidence depuis nombre d'années que l'opérateur fait partie d'un système sociotechnique, c'est-à-dire qu'il agit en interdépendance, au sein d'un cadre organisationnel constitué d'exigences et de règles plus ou moins explicites, avec des machines et outils plus ou moins bien adaptés, et avec des collègues, y compris une ligne hiérarchique, dont les rôles respectifs sont plus ou moins bien définis.

Diverses définitions et nomenclatures de l'erreur existent. Reason (Reason, 1990), un auteur bien connu dans le domaine, définit l'erreur comme « tous les cas où une séquence planifiée d'activités mentales ou physiques ne parvient pas à ses fins désirées, et quand ces échecs ne peuvent être attribués à l'intervention du hasard ».

Villemeur (Villemeur, 1988) définit l'erreur humaine comme la cessation de l'aptitude de l'opérateur humain à accomplir une mission requise.

2 Classification des erreurs humaines

Il existe de nombreuses classifications des erreurs humaines (Rasmussen et Jensen-1974, Reason-1979, Rasmussen-1980, (Norman D. A., 1981), (Leplat, 1985), (Villemeur, 1988),

(Reason, 1990), Shappel et Wiegmann-2004), ~~mais celles qui reviennent le plus~~, sont celles de Reason (Reason, 1990), Norman (Norman D. A., 1981) et celle de Rasmussen (Rasmussen, 1983). Je commencerai par décrire celle de Rasmussen, puis les différentes classifications de Reason et je finirai par celle de Hollnagel (Hollnagel, 1993). Je ne parlerai pas de celle de Norman, qui est sans intérêt par rapport au sujet de recherche qui est de représenter les erreurs humaines dans les systèmes interactifs critiques, alors que Norman était essentiellement désireux de rendre compte des actions qui se produisent dans le déroulement normal de la vie quotidienne, généralement sans conséquence (Reason, 1990).

2.1 Classification en niveaux d'activité Automatisme-règle-connaissance de Rasmussen et Jensen (Rasmussen, 1983)

La classification « automatisme-règle-connaissance » a trouvé son origine dans une analyse de protocoles verbaux de techniciens dépannant des appareils électroniques. Cette distinction des niveaux d'activité est devenue une « norme de marché » dans la communauté de la fiabilité des systèmes critiques (Reason, 1990). Les trois niveaux d'activité correspondent à des niveaux décroissants de familiarité avec l'environnement de la tâche.

Le niveau basé sur les automatismes :

A ce niveau, l'activité humaine est contrôlée par des configurations mémorisées d'instructions préprogrammées, représentées comme des structures analogiques dans l'espace-temps (par exemple, le contrôle de la vitesse et de la direction d'un véhicule). A ce niveau, les erreurs prennent deux formes : le déclenchement d'un comportement cohérent, mais au mauvais endroit et/ou au mauvais moment, ou des omissions.

Le niveau basé sur les règles :

Ce niveau s'applique à la résolution de problèmes familiers, dont les procédures sont contrôlées par des règles mémorisées (règles de production) du type : si [état] alors [diagnostic] ou si [état] alors [action de remédiation]. Ici, les erreurs sont typiquement liées à de mauvaises classifications de situations qui conduisent à l'application de règles erronées ou rappel incorrect de procédures.

Le niveau basé sur les connaissances :

Ce niveau entre en jeu dans les situations nouvelles pour lesquelles les actions doivent être planifiées en temps réel, en s'appuyant sur des processus analytiques et des connaissances mémorisées, avec un contrôle conscient. A ce niveau, les erreurs sont dues à des limitations de ressources et à des connaissances incomplètes ou incorrectes.

2.2 Classification par type d'erreur, et étapes de conception de Reason (Reason, 1990)

Reason a situé le type d'erreur parmi les étapes qui vont de la conception à la mise en œuvre de la séquence d'actions. Il a classé ces étapes en trois grandes catégories (voir Tableau 8)

Tableau 8. Classification des types d'erreurs en fonction des étapes cognitives où ils apparaissent

Etape cognitive	Type fondamental d'erreur
Planification	Fautes
Stockage	Lapsus
Exécution	Ratés

La planification : met en jeu divers processus qui identifient le but à atteindre et les moyens pour y parvenir.

Le stockage : Comme les plans ne sont généralement pas immédiatement mis en œuvre, cette phase de durée variable prend place entre la formulation des actions et leur exécution.

L'exécution : recouvre les processus impliqués dans la mise en œuvre effective du plan mémorisé.

Reason inspiré des modèles de Rasmussen et Rousse a pu identifier des catégories de classification de modes de défaillance à chacun des trois niveaux d'activités de Rasmussen (voir Tableau 9)

Tableau 9. Principales catégories de classification des modes de défaillance aux niveaux d'activité

<table>
<tr><td colspan="2">Activité (ratés lapsus) basée sur les automatismes</td></tr>
<tr><td>Inattention</td><td>Attention excessive</td></tr>
<tr><td>• Ratés de double capture
• Omissions suivant des interruptions
• Intentionnalité réduite
• Confusions perceptives
• Erreurs d'interférence</td><td>• Omissions
• Répétitions
• inversions</td></tr>
<tr><td colspan="2">Activité (fautes) basée sur les règles</td></tr>
<tr><td>Mauvaise application de règles correctes</td><td>Application de règles erronées</td></tr>
<tr><td>• Premières exceptions
• Contresignes et non-signes
• Surcharge d'information
• Force de la règle
• Règles générales
• Redondance
• Rigidité</td><td>• Déficiences d'encodage
• Déficiences de l'action</td></tr>
<tr><td colspan="2">Activité (fautes) basée sur les connaissances déclaratives</td></tr>
<tr><td colspan="2">• Sélectivité
• Limitations de l'espace de travail
• Hors de la vue, hors de l'esprit
• Biais de confirmation
• Confiance excessive
• Révision biaisée
• Corrélation illusoire
• Effets de halo
• Difficulté avec la causalité
• Difficulté avec la complexité</td></tr>
</table>

Les ratés / les lapsus : ils sont basés sur les automatismes du sujet. C'est un échec de l'exécution, c'est-à-dire que l'action ne se déroule pas selon le plan prévu (ex: vouloir cliquer sur le bouton vert et cliquer sur le rouge...). Pour leur plus grande part, les ratés impliquent l'inattention, en omettant de faire une vérification nécessaire. Mais bon nombre d'entre eux sont dus à une attention excessive (vérification attentionnelle à un moment inapproprié de la séquence d'action automatisée).

Les fautes basées sur les règles : échec de la planification (le plan est inadéquat vis à vis des objectifs). On peut classer ces erreurs en deux catégories : a) les erreurs qui proviennent d'une mauvaise application de mauvaises règles, et b) celles qui sont dues à l'application de règles fausses. Une "bonne règle" est une règle qui s'est révélée utile dans une situation particulière, mais peut conduire à de mauvaises applications. Ce n'est que par la survenue de telles erreurs que ces règles "parentes" engendreront les règles "filles", plus spécifiques, nécessaires pour faire face à la variété des situations. Plus une règle a de "victoires" à son crédit (plus elle a conduit à un résultat satisfaisant), plus cette règle est "forte", plus elle a de chances de gagner dans des situations futures (et moins elle requiert un ajustement de qualité à la situation pour être déclenché). Les règles les plus générales (=les plus "hautes") sont presque toujours les plus fortes.

Les fautes basées sur les connaissances déclaratives : échec de la planification. A ce niveau, les erreurs prennent naissance dans deux aspects de la cognition humaine : la rationalité limitée et le fait que les connaissances pertinentes à l'espace-problème sont presque toujours incomplètes et souvent incorrectes. Ces défaillances surviennent lorsque le sujet doit avoir un raisonnement « en temps réel ». En raisonnant à ce niveau, on impose une lourde charge à l'espace de travail, qui est limité.

Les ratés précèdent généralement la détection d'un problème, alors que les fautes surviennent pendant les tentatives subséquentes de découverte d'une solution. La prise de conscience de l'existence d'un problème est une condition de définition des fautes basées sur les règles et sur les connaissances déclaratives.

Dans les deux premiers types d'erreurs, l'activité est caractérisée par un contrôle proactif (émanant de structures de connaissances stockées), alors que dans le troisième, le contrôle est essentiellement rétroactif (l'activité est dirigée par l'erreur). Aux niveaux basés sur les automatismes et basés sur les règles, les erreurs prennent la forme de routines fortes, mais fausses. Au niveau basé sur les connaissances déclaratives, la forme des fautes est beaucoup plus imprévisible

Je vais citer un exemple d'erreur pour chaque catégorie, plus de détails dans le livre de Reason (Reason, 1990) page 108.

Ratés de double capture : « j'avais décidé de réduire ma consommation de sucre et je voulais manger mes pétales de maïs sans sucre. Cependant, le matin suivant, j'ai saupoudré de sucre mes céréales comme je l'avais toujours fait »

Omissions suivant des interruptions : « En faisant le thé, je me suis rendu compte que la boîte était vide, j'ai pris un paquet de thé sur l'étagère et j'ai rempli la boite. Mais en suite j'ai omis de mettre du thé dans la théière et j'ai versé de l'eau bouillante dans une bouilloire vide »

Intentionnalité réduite : « je voulais fermer la fenêtre car il faisait froid, au lieu de cela j'ai fermé la porte du placard » -> Ratés

« J'ouvrais le réfrigérateur et je restais là, en regardant son contenu, incapable de me souvenir de ce que je voulais » ->Lapsus

2.3 Classification par phénotype et génotype d'erreur de Hollnagel

Les causes des erreurs et leur observation sont des concepts différents, qui devraient être séparé lors de l'analyse des erreurs des utilisateurs. Pour ce faire, Hollnagel (Hollnagel, 1991) a proposé une terminologie basée sur 2 concepts principaux : le phénotype et génotype. Le phénotype d'une erreur est défini comme étant l'action erronée que l'on peut observer. Le génotype de l'erreur est défini comme étant les caractéristiques de l'opérateur qui peut contribuer à l'apparition d'une action erronée.

3 Conclusion

Ce chapitre a présenté une étude des principales approches de description des erreurs humaines identifiées dans différents travaux du domaine de la psychologie. Le contenu de cette description est complété par une Annexe dédiée. En effet, certaines de ces contributions sont redondantes ou sont incluses les unes dans les autres et nous n'avons représenté dans le chapitre que les principales sources.

Les classifications présentées dans ce chapitre restent académiques et indépendantes de tout travail de conception ou d'analyse de systèmes. Le but ultime de ce travail de thèse étant de permettre la conception de systèmes tolérant aux erreurs humaines, il est nécessaire de pouvoir identifier de façon systématique ces erreurs sur un système donné. Pour cette raison le chapitre suivant présente les techniques et méthodes d'identification des erreurs humaine qui exploitent toutes les classifications présentées dans ce chapitre.

Chapitre 3. Technique et méthodes d'identification d'erreurs humaines

Ce chapitre présente un état de l'existant des différent technique et méthodes d'identification d'erreurs humaines. Nous commencerons par présenter les différents critères de comparaison de ces techniques, puis classifier ces techniques, afin d'en déduire les manques à rajouter, et enfin nous finirons par présenter la technique choisis à partir de la classification, afin de l'améliorer.

1 Classification des techniques et méthodes d'identification des erreurs humaines

Pour toutes les techniques présentées dans le Tableau 10, le processus d'identification des erreurs humaines potentielles repose fortement sur la description de tâches utilisateurs. Cette description de tâches doit être précise, complète et représentative des activités de l'utilisateur, afin d'être en mesure d'identifier toutes les erreurs possibles. En effet, le langage de description de tâches, ainsi que le moyen pour produire cette description affectent la qualité de l'analyse. Cependant, la plupart d'entre elles utilisent la technique HTA (Hierarchical Task Analysis) qui permet la décomposition des objectifs de l'utilisateur en tâches et sous-tâches, et pour décrire les relations séquentielles entre ces tâches (dans une représentation textuelle distincte appelée « plan »). Comme HTA ne fournit pas de support pour décrire précisément, les types d'actions de l'utilisateur, les différents types d'opérateurs d'ordonnancement temporels autres que la séquence d'action (comme la concurrence d'actions, l'ordre indépendant d'actions...), ainsi que les informations et les connaissances nécessaires pour effectuer une action, les erreurs liées à ces éléments ne peuvent être identifiés. En outre, comme la plupart de ces techniques ne possèdent pas d'outil associé, il est complexe de vérifier la couverture et de stocker les erreurs identifiées de façon systématique. Par exemple, comme HTA ne fournit pas de support pour décrire les connaissances nécessaires pour accomplir une tâche, aucune de ces méthodes ne fournissent un soutien explicite pour l'identification de toutes les erreurs possibles basées sur la connaissance.

1.1 Hazard and operability study (HAZOP)

La technique d'analyse de système d'étude HAZOP (Hazard and Operability) (Lawley, 1973) a été développée à la fin des années 1960. HAZOP a été développé en tant que technique permettant d'enquêter sur la sécurité ou la fonctionnalité d'une installation ou d'une opération. Il a été largement utilisé dans les industries de traitement de l'énergie nucléaire et des procédés chimiques. HAZOP est une approche technique bien établie qui a été développée pour être utilisée dans les audits de conception et l'évaluation des risques d'ingénierie. L'approche de type HAZOP a été mise au point, le simple fait de tirer des enseignements d'incidents passés n'étant plus acceptable dans les grandes usines chimiques.

Appliquée à l'origine aux diagrammes d'ingénierie, la technique HAZOP implique que l'analyste applique des mots-clés, tels que Non effectué, Supérieur ou ultérieur à, à chaque étape

d'un processus afin d'identifier les problèmes pouvant survenir. En règle générale, les analyses HAZOP sont effectuées sur la conception finale d'un système. Lors d'une analyse de type HAZOP, une équipe HAZOP est généralement composée d'opérateurs, de concepteurs, de spécialistes des facteurs humains et d'ingénieurs. Le responsable de HAZOP (qui devrait avoir une grande expérience des analyses de type HAZOP) guide l'équipe dans une enquête sur la conception du système à l'aide des mots-clés de « déviation » de HAZOP. L'équipe HAZOP considère les mots-clés pour chaque étape d'un processus afin d'identifier ce qui peut mal tourner. Les mots-clés sont proposés et le responsable demande ensuite à l'équipe de considérer le problème de la manière suivante :

- Quelle section de l'usine est envisagée ?
- Quel est l'écart et que signifie-t-il ?
- Comment cela peut-il arriver et quelle est la cause de la déviation ?
- Si cela ne peut pas arriver, passez à la déviation suivante.
- Si cela peut arriver, y a-t-il des conséquences importantes ?
- S'il n'y en a pas, passez au mot suivant.
- S'il y a des conséquences, quelles sont les fonctionnalités incluses dans la centrale pour faire face à ces conséquences ?
- Si l'équipe HAZOP estime que les conséquences du projet proposé ne sont pas suffisamment prises en compte, des solutions et des actions sont alors envisagées.

La Figure 17 représente le processus d'identification d'erreur humaine HAZOP à l'aide de modèles de tâches de types HTA. Ce processus prend donc en entrée un modèle de tâches HTA, puis la première tâche élémentaire, puis une analyse des erreurs est faite à l'aide de la taxonomie HAZOP, et à partir de cette taxonomie, on analyse si l'erreur est potentielle ou non, si oui on l'a décrit, sinon en passe à la suivante, puis à la tâche suivante.

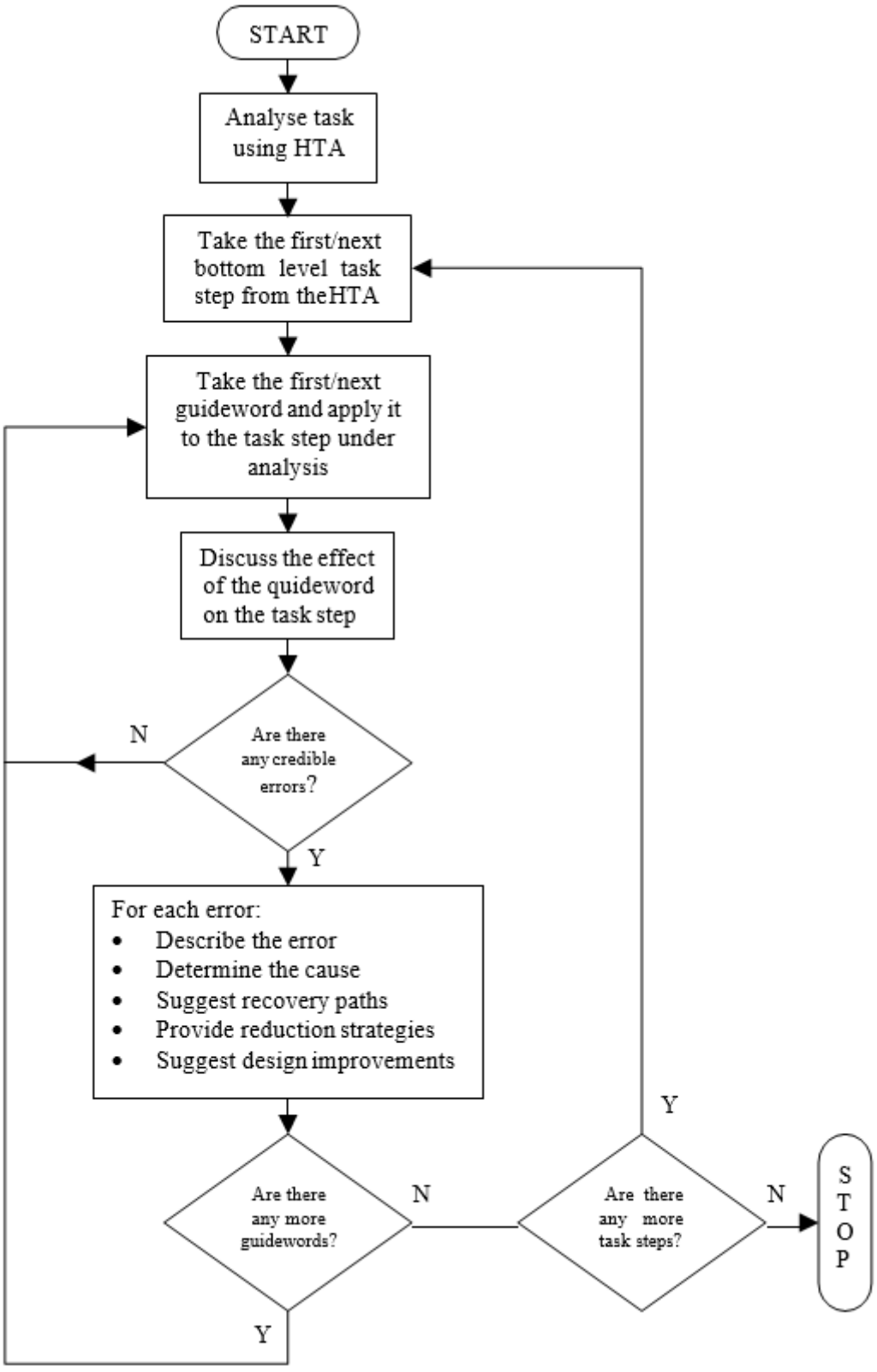


Figure 17. Processus d'identification d'erreur humaine HAZOP à l'aide de HTA

1.2 Systematic human error reduction and prediction approach (SHERPA)

SHERPA a été développé par (Embrey, 1986) en tant que technique de prévision des erreurs humaines, qui a également permis d'analyser les tâches et de présenter de manière structurée les solutions potentielles aux erreurs. La technique est basée sur une taxonomie de l'erreur humaine spécifiant le mécanisme psychologique impliqué dans l'erreur.

SHERPA a été initialement conçu pour aider les personnes travaillant dans les industries de traitement (par exemple, la production d'énergie conventionnelle et nucléaire, le traitement pétrochimique, l'extraction de pétrole et de gaz et la distribution d'énergie). Le domaine d'application s'est élargi à partir des années 2000 pour inclure les distributeurs de billets, les distributeurs automatiques et les distributeurs de cassettes audio dans la voiture.

La Figure 18 représente le processus d'identification des erreur humaine SHERPA à partir d'un modèle de tâches HTA.

Le processus commence par prendre en entré un modèle de tâches

Puis il prend une tâche élémentaire et la classifie en fonction du type de la tâche.

Une question sera posée, afin d'analyser s'il y a une erreur potentielle.

Si une erreur est identifiée, elle sera décrite et classer,

On boucle jusqu'à ce qu'on n'identifie plus d'erreur, puis en passe à la tâche élémentaire suivante.

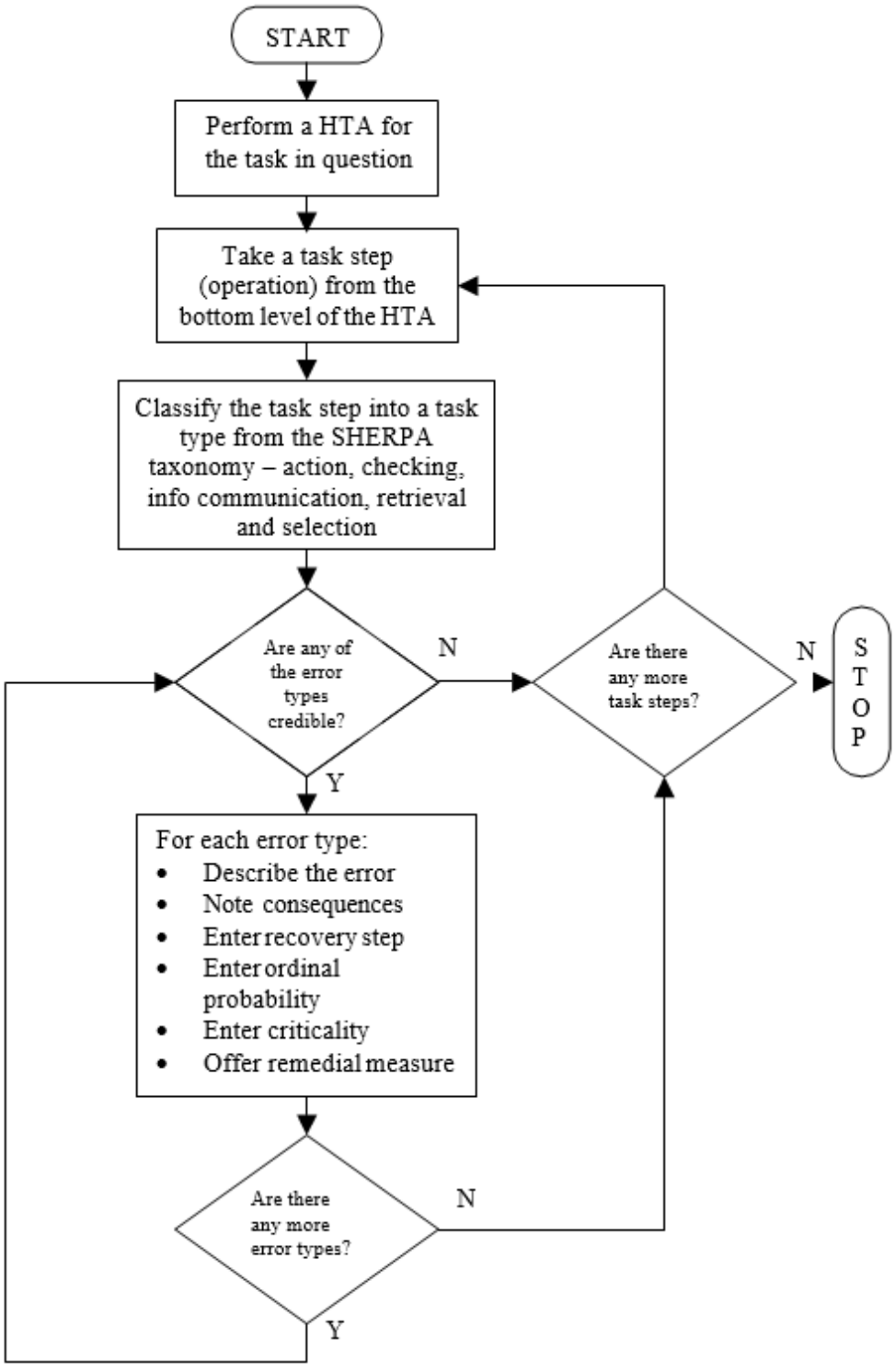


Figure 18. Processus SHERPA

1.3 Cognitive reliability and error analysis method (CREAM)

La méthode d'évaluation de la fiabilité et des erreurs cognitives (CREAM) (Hollnagel, 1998) est une méthode d'identification d'erreur humaine et d'analyse de fiabilité humaine. CREAM peut être utilisé à la fois de manière prédictive pour prédire l'erreur humaine potentielle et de manière rétrospective pour analyser et quantifier l'erreur. La technique CREAM consiste en

une méthode, un schéma de classification et un modèle. CREAM permet à l'analyste d'atteindre les objectifs suivants :

- Identifiez les parties du travail, les tâches ou les actions qui nécessitent ou dépendent de la cognition humaine et qui peuvent donc être affectées par les variations de la fiabilité cognitive.
- Déterminer les conditions dans lesquelles la fiabilité de la connaissance peut être réduite et où, partant, les actions peuvent constituer une source de risque.
- Fournir une évaluation des conséquences de la performance humaine sur la sécurité du système.
- Développer et spécifier des modifications qui améliorent ces conditions, servent donc à augmenter la fiabilité de la cognition et à réduire le risque.

1.4 Human error assessment and reduction technique (HEART)

HEART ou la technique d'évaluation et de réduction des erreurs humaines (Williams J. C., 1988) a été conçue principalement comme une technique d'identification d'erreur rapide, simple à utiliser et facile à comprendre. HEART est une technique hautement procédurale qui tente de quantifier l'erreur humaine. L'aspect le plus important de la technique HEART est le fait qu'elle ne vise que les erreurs qui auront un effet important sur le système en question, afin de réduire l'utilisation des ressources lors de l'application de la technique. La méthode utilise ses propres valeurs de fiabilité ainsi que des « facteurs d'effet » pour un certain nombre de conditions générant des erreurs. La méthodologie HEART a principalement été utilisée dans les évaluations de centrales nucléaires. La technique a été utilisée au Royaume-Uni pour l'évaluation des risques Sizewell B ainsi que pour les évaluations des risques pour les centrales britanniques Magnox et les réacteurs à refroidissement par gaz avancés.

1.5 Human error identification in system tool (HEIST)

L'outil d'identification des erreurs humaines dans les systèmes (HEIST) (Kirwan, 1994) est une technique basée sur une série de tableaux contenant des questions ou des «types d'erreurs» entourant les modes d'erreur externe (EEM), les facteurs de mise en forme de la performance (PSF) et les facteurs psychologiques. Lors de l'utilisation de HEIST, l'analyste identifie les erreurs en appliquant un ensemble de questions à toutes les tâches impliquées dans un scénario. Les questions associent le type d'erreur (EEM) aux PSF pertinents. Tous les EEM sont alors liés aux PEM (mécanismes d'erreur psychologiques). Une fois que cela est fait, le potentiel de récupération, les conséquences et les mécanismes de réduction des erreurs sont consignés dans un format d'analyse des erreurs sous forme de tableau. Cette approche questions-réponses concernant l'identification des erreurs se présente sous la forme d'un tableau contenant un code pour chaque question d'identification d'erreur, la question de l'identificateur d'erreur, le mode d'erreur externe (EEM), la cause identifiée (cause du système ou mécanisme d'erreur psychologique) et les directives de réduction des erreurs proposées.

La Figure 19 représente le processus d'identification des erreurs humaines à partir de modèles de tâches HTA, le processus se déroule de la même façon que celui de HAZOP, sauf que la taxonomie utilisée n'est pas la même, et la description de l'erreur.

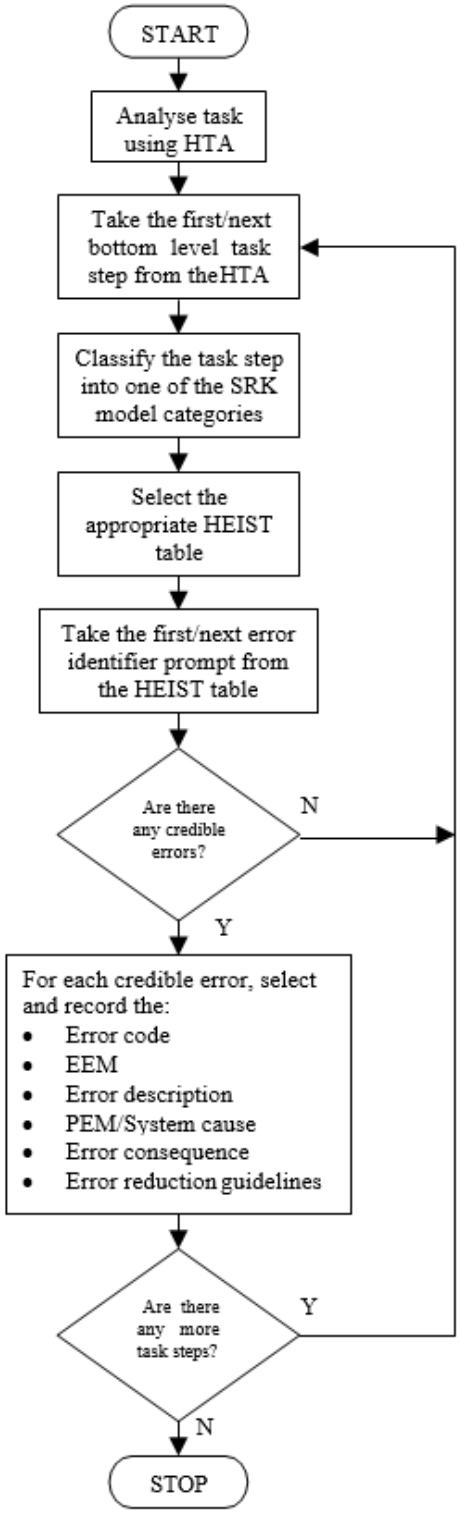


Figure 19. Processus d'identification d'erreur humaine HEIST

1.6 Human error template (HET)

HET (Human Error Template) (Stanton, et al., 2006) est une approche du style checklist pour la prédiction des erreurs qui se présente sous la forme d'un tableau d'erreur contenant douze modes d'erreur. Figure 20 montre le processus de la méthode HET. Le HET est appliqué à chaque étape d'une action élémentaire d'un modèle de tâche hiérarchique (HTA). La technique exige que l'analyste indique quels modes d'erreur HET sont crédibles (ou le cas échéant) pour chaque action élémentaire d'un modèle de tâches, en fonction de leur jugement. La taxonomie des erreurs HET se compose de 12 modes d'erreur de base sélectionnés sur la base d'une étude de l'incidence réelle des erreurs pilotes et des modes d'erreur existants utilisés dans les méthodes HEI contemporaines. Les douze modes d'erreur HET sont :

- Échec de l'exécution
- Exécution de la tâche incomplète
- Tâche exécutée dans la mauvaise direction
- Mauvaise tâche exécutée
- Tâche répétée
- Tâche exécutée sur le mauvais élément d'interface
- Tâche exécutée trop tôt
- Tâche exécutée trop tard
- Tâche exécutée trop lentement
- Tâche exécutée trop rapidement
- Informations erronées
- Autre

En prenant en entrée un modèle de tâches HTA pastille « 1 », on extrait toutes les tâches élémentaires, pour en prendre la première/suivante non traitée pastille « 2 ». Cette étape est importante dans le processus d'identification et de représentation des erreurs humaines, en effet on a besoin de parcourir les tâches élémentaires de façon systématique.

En suite en prend le scénario et les détails d'exécution de la tâche dans un tableau (voir pastille « 3 »). Ce qui nous permettra d'effectuer une analyse sur chaque tâche afin d'identifier les différentes erreurs potentielles.

Puis à l'aide de la liste des modes d'erreur citées ci-dessus, nous prenons le premier mode/ mode suivant (voir pastille « 4 »)

On analyse le mode d'erreur par rapport à la tâche élémentaire, puis on décide si l'erreur est potentielle ou non (voir pastille « 5 »)

Si l'erreur est crédible, on décrit l'erreur dans le tableau, en précisant la probabilité et la criticité de l'erreur (voir pastille « 6 »), puis on passe au mode suivant.

Si l'erreur n'est pas crédible, on passe directement au mode suivant (voir pastille « 7 »).

S'il n'y a plus de mode d'erreur, on passe à la tâche élémentaire suivante.

Et s'il n'y a plus d'autre tâche élémentaire (voir pastille « 8 »), on a fini l'identification.

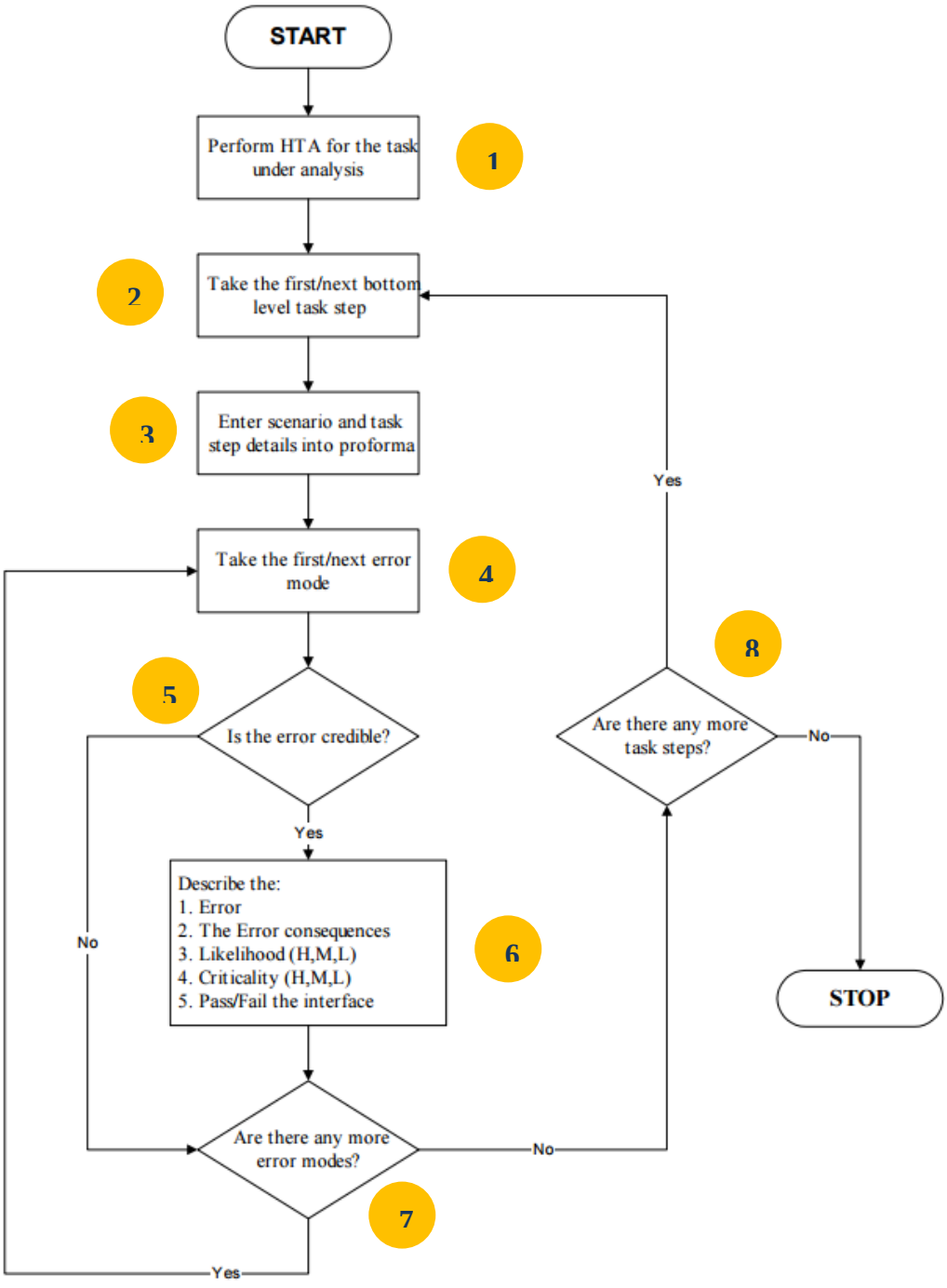


Figure 20. Prossessus d'identification d'erreurs humaines HET de Stanton

1.7 System for predictive error analysis and reduction (SPEAR)

Le système d'analyse prédictive des erreurs (SPEAR) (The Center for Chemical Process Safety, 1994) a été mis au point par le Center for Chemical Process Safety (CCPF) pour être utilisé dans le programme HRA de l'industrie américaine du traitement des produits chimiques.

SPEAR est une approche systématique d'identification des erreurs qui est très similaire à d'autres techniques d'identification d'erreurs systématique, telles que SHERPA. La principale différence entre SPEAR et SHERPA réside dans le fait que la technique SPEAR utilise des facteurs de performance (PSF) afin d'identifier tout facteur environnemental ou situationnel susceptible d'accroître le risque d'erreur. La technique SPEAR elle-même agit sur les tâches de niveau inférieur (opérations) d'un HTA de la tâche analysée. . À l'aide d'un jugement subjectif, l'analyste utilise la taxonomie des erreurs humaines SPEAR pour classer chaque étape de la tâche dans l'un des cinq types de comportement suivants :

- Action
- Récupération
- Vérifier
- Sélection
- Transmission

1.8 Task Analysis For Error Identification (TAFEI)

L'analyse des tâches pour l'identification des erreurs (TAFEI) (Pocock, Harrison, Wright, & Johnson, 2001) est une méthode qui permet aux utilisateurs de prévoir les erreurs d'utilisation du périphérique en modélisant les interactions entre l'utilisateur et le périphérique analysé. Il suppose que les utilisateurs utilisent les appareils de manière ciblée, de telle sorte que l'interaction puisse être décrite comme un « effort de coopération », et c'est par ce processus que des problèmes se posent.

En outre, la technique suppose que les actions sont limitées par l'état du produit à un moment donné de l'interaction et que le périphérique offre des informations à l'utilisateur sur ses fonctionnalités. Ainsi, l'interaction entre les utilisateurs et les appareils progresse dans une séquence d'états. À chaque état, l'utilisateur sélectionne l'action la plus pertinente par rapport à son objectif, en fonction de l'état du système.

La base de cette approche est basée sur la théorie générale des systèmes. Cette théorie est potentiellement utile pour traiter l'interaction entre les sous-composants dans les systèmes (c'est-à-dire l'humain et le périphérique). Cela suppose également un ordre hiérarchique des composants du système, c'est-à-dire que toutes les structures et fonctions sont ordonnées en fonction de leur relation avec d'autres structures et fonctions, et que tout objet ou événement particulier est composé d'objets et d'événements de moindre importance. Les informations concernant l'état de la machine sont reçues par la partie humaine du système par le biais de processus sensoriels et perceptuels, et converties en activité physique sous la forme d'entrées dans la machine. L'entrée modifie l'état interne de la machine et un retour est fourni à l'homme sous forme de sortie. La frontière entre les humains et les machines est particulièrement intéressante, car c'est là que les erreurs deviennent apparentes.

Sur le plan procédural, la TAFEI comprend trois étapes principales. Tout d'abord, une analyse hiérarchique des tâches (HTA) est réalisée pour modéliser le côté humain de l'interaction. Comme il apparaîtra clairement, TAFEI se concentre sur une série de tâches visant à atteindre un objectif spécifique. Ensuite, les diagrammes espace-état (SSD) sont construits pour

représenter le comportement de l'artefact. Les plans de HTA sont mappés sur le SSD pour former le diagramme TAFEI. Enfin, une matrice de transition est conçue pour afficher les transitions d'état lors de l'utilisation du périphérique. TAFEI a pour but d'aider à la conception d'artefacts en illustrant les cas où une transition d'état est possible mais non souhaitable (c'est-à-dire illégale). Rendre impossible toute transition illégale devrait faciliter la coopération dans l'utilisation de l'appareil.

2 Critères de comparaison des techniques et méthodes existantes

De nombreuses techniques ont été proposées pour identifier les erreurs humaines peuvent se produire dans un contexte particulier, et quelles pourraient être les conséquences dans ce contexte donné. Plusieurs techniques d'évaluation de la fiabilité humaines telles que CREAM, HEART, et THERP sont fondées sur l'analyse des tâches. Ils apportent un soutien pour évaluer la possibilité d'occurrence d'erreurs humaines en structurant l'analyse autour d'une description de tâches. Au-delà de ces points communs, THERP fournit les moyens d'évaluer la probabilité d'occurrence d'erreurs humaines. Le Tableau 10 présente une vue d'ensemble sur les techniques existantes pour identifier les erreurs humaines potentielles. Pour chaque technique, les informations suivantes sont mises en évidence :

- Type de technique : pour indiquer à quel domaine scientifique cette technique est liée. Les types peuvent être HEI (Human Error Identification), DC (Dependable Computing), SA (Safety Analysis) ...
- Technique de description de tâches associée : pour indiquer comment les tâches utilisateur sont décrites. La plupart d'entre elles exploitent la notation HTA (Hierarchical Task Analysis) ;
- Prise en charge d'outils pour l'analyse et la modélisation de la tâche : pour indiquer si oui ou non un outil logiciel est disponible pour fournir un appui pour l'application de la technique ;
- Classification d'erreur associée : pour indiquer quelle classification d'erreur humaine est utilisée pour identifier les erreurs potentielles. « G » et « S » indique si la classification générique provient d'une analyse des défaillances du système, ou si elle est spécifique aux erreurs humaines ;
- Capacité à traiter les combinaisons d'erreur : pour indiquer si oui ou non les techniques fournissent un moyen explicite pour identifier les combinaisons possibles d'erreurs. Ici seulement 2 valeurs sont possibles : « Non » (indique que la méthode ne fournit pas les moyens de traiter les combinaisons d'erreurs) et « Oui » (indique que la méthode fournit les moyens de traiter les combinaisons d'erreurs, mais pas de forcement de façon explicite).

Tableau 10. Classification des technique et méthodes d'identification des erreurs humaines

Nom de la technique	Type de technique	Technique de description	Outil logiciel	Classification d'erreur associée (Générique/Spécifique)		Combinaisons d'erreur
Hazard and operability study (HAZOP) (Lawley, 1973)	Safety analysis	HTA	Aucun	Not done, Less, More, As well as, Other than, Repeated, Sooner, Later, Misordered, Part of	G	Oui
Systematic human error reduction and prediction approach (SHERPA) (Embrey, 1986)	HEI, HRA	HTA	Aucun	Action errors, Checking errors, Communication errors, Info retrieval errors, Selection errors	S	Non
Potential human error cause analysis (PHECA) (Kirwan, 1992)	HEI, HRA	HTA	Aucun	HAZOP classification	G	Oui
Cognitive reliability and error analysis method (CREAM) (Hollnagel, 1998)	HEI, HRA	HTA	Aucun	Timing,Duration, Sequence, Object, Force, Direction, Distance, Speed	S	Oui
Human error assessment and reduction technique (HEART) (Williams J. C., 1988)	HEI, HRA	HTA	Aucun	None (concrete description of the human error)	S	Oui
Human error identification in system tool (HEIST) (Kirwan, 1994)	HEI, HRA	HTA	Aucun	Skill, Rule, Knowledge model	S	Oui
Human error template (HET) (Stanton, et al., 2006)	HEI, Human Factors	HTA	Aucun	Fail to execute : Task execution incomplete, Task executed in the wrong direction, Wrong task executed, Task repeated, Task executed on the wrong interface element Task executed : Too early/too late/too much/too little, Misread information, other	S	Non
System for predictive error analysis and reduction (SPEAR) (The Center for Chemical Process Safety, 1994)	HEI, HRA, SA	HTA	Aucun	Action, Retrieval, Check, Selection, Transmission	G	Non
Task Analysis For Error Identification (TAFEI) (Pocock, Harrison, Wright, & Johnson, 2001)	HEI, Human Factors	HTA	HTA	Generic categories	S	Non

3 Conclusion

Ce chapitre a présenté un ensemble de critères pour comparer les techniques et méthodes d'identification des erreurs humaines, ainsi qu'une comparaison de ces différentes techniques et méthodes selon ces critères. Nous avons en particulier détaillé la méthode HET qui va servir de cadre aux travaux présentés dans les chapitres suivants.

L'identification des erreurs humaine n'est pas une fin en soi. En effet, le résultat d'une phase d'identification des erreurs humaines à vocation à être exploité dans des phases aval comme la modélisation ou l'analyse de tolérance aux erreurs humaines. Il existe dans la littérature de nombreuse méthodes d'analyse de tolérance aux erreurs humaines (Human Reliability Analysis methods) comme par exemple CREAM (Cognitive reliability and error analysis method) (Hollnagel, 1998) ou THERP (Technique for human error rate prediction) (Swain & Guttmann, 1964). Dans la mesure où ce travail de thèse se focalise sur la partie identification et représentation des erreurs humaines nous ne détaillerons pas ces méthodes ni leurs fondements. Le lecteur intéressé peut trouver une présentation et une comparaison de ces méthodes dans (Swain A. D., 1989).

PARTIE II – Contributions

Chapitre 4. Extensions de la notation de modélisation de tâches HAMSTERS pour la prise en compte des erreurs humaines

Il existe un nombre important de types d'erreurs humaine (voir Annexe). Afin de pouvoir les analyser et les représenter dans des modèles de tâches, il est important de pouvoir les identifier de manière systématique et fournir une représentation graphique.

Ce chapitre présente :

1. Les taxonomies utilisées pour l'identification et la représentation des erreurs humaines dans les modèles de tâches
2. Les extensions apportées à la notation et l'outil HAMSTERS pour l'identification et la représentation des erreurs humaines dans les modèles de tâches

Plusieurs taxonomies des erreurs humaines ont été proposé afin de regrouper ces erreurs. Notre travail n'était pas de proposer une nouvelle taxonomie, mais plutôt de les représenter dans une notation de modélisation de tâches.

Dans ce chapitre nous allons présenter les taxonomies retenus d'après le Chapitre 2 , afin de les décrire dans la notation de modélisation de tâches HAMSTERS. Nous présenterons aussi les extensions apportées à l'outils HAMSTERS qui permettent la prise en compte des erreurs humaines. On finira le chapitre par un exemple d'une étude approfondie de l'utilisation du distributeur automatique de billets (DAB).

Tout au long de ce chapitre nous allons illustrer avec un exemple simple et connu par tous, qui est la tâche « Retirer de l'argent » à partir d'un DAB.

1 Taxonomies d'erreurs humaines utilisées dans HAMSTERS

Dans cette section, nous allons présenter les différentes classifications d'erreurs humaines utilisées pour étendre la notation et l'outil HAMSTERS. Ces taxonomies sont décrite plus en détails dans le Chapitre 2 Nous commencerons par présenter et différencier le concept de phénotype et le concept de génotype, puis nous présenterons le lien entre le concept de génotype, les 3 niveaux d'activité SRK (Rasmussen, Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models, 1983) et la classification GEMS (Reason, 1990).

1.1 Une structuration à base de phénotype et génotype

Plusieurs contributions dans le domaine des facteurs humains portent sur l'étude des processus humains internes qui peuvent conduire à des actions qui peuvent être perçues comme erronée d'un point de vue externe (voir chapitre 2). Les causes des erreurs et leur observation sont des

concepts différents, qui devraient être séparés lors de l'analyse des erreurs des utilisateurs. Pour ce faire, Hollnagel (Hollnagel, 1993) a proposé une terminologie basée sur 2 concepts principaux : le phénotype et le génotype. Le phénotype d'une erreur est défini comme étant l'action erronée que l'on peut observer. Le génotype de l'erreur est défini comme une cause pouvant contribuer à l'apparition d'une action erronée.

Pour illustrer le phénotype et le génotype, prenons comme exemple le distributeur automatique de billets (DAB). Lors d'une tâche de retrait d'argent, une des erreurs qui peut se manifestée est le fait de se tromper en tapant le code pin, et après validation du code, on s'aperçoit de notre erreur, avec un message sur l'écran nous informant qu'on s'est trompé, c'est le phénotype de l'erreur. Les causes de cette erreur sont nombreuses. Il pourrait s'agir d'un appui intentionnel sur une autre touche, d'une omission suite à une interruption durant la saisie du code pin (l'utilisateur reprenant la saisie avec une touche déjà appuyée), ou bien la saisie du code pin associé à une autre carte. Ces causes-là sont des génotypes, et peuvent être classées parmi 3 types : Raté (Slip) (Norman D. A., 1981), Lapsus (Lapse) (Reason, 1990) ou bien Faute (Mistake) (Reason, 1990).

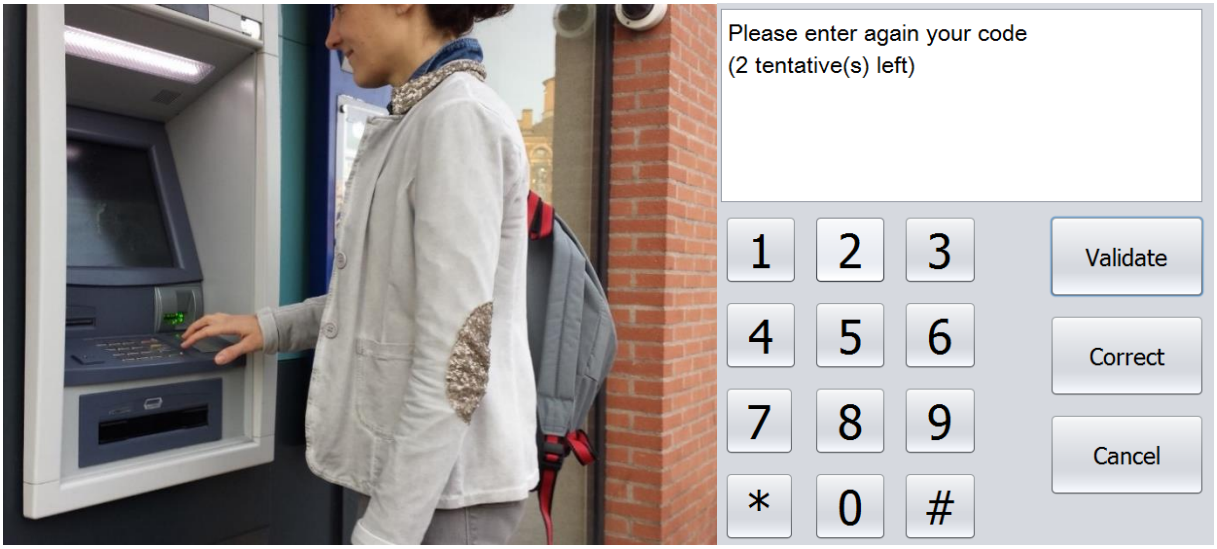


Figure 21. Distributeur automatique de billet

1.2 Une décomposition du génotype en fonction du niveau de traitement de l'information (SRK)

Dans les années 80, Rasmussen a développé un modèle qui définit trois types de comportement présent dans le traitement de l'information d'un opérateur, les trois catégories décrivent essentiellement les niveaux de contrôle du comportement humain (Rasmussen, Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models, 1983) :

- Comportement basé sur les automatismes :
- Comportement basé sur les règles :
- Comportement basé sur les connaissances :

Dans chaque comportement, les types d'erreurs humaines (génotype) diffèrent, et c'est sur cette classification que Reason a proposé un raffinement de types d'erreurs par rapport au niveau d'activité (Reason, 1990).

1.3 Raffinement des génotypes en raté, lapsus et faute et mise en relation avec SRK

Dans les années 1970, Norman, Rasmussen et Reason ont proposé des théories pour analyser l'erreur humaine. Norman, a proposé une classification des différents types de ratés (Slips) (Norman D. A., 1981). Rasmussen a proposé un modèle de performance humaine qui distingue trois niveaux : Automatismes, Règles et Connaissances (modèle Skills-Rules-Knowledges SRK) (Rasmussen, 1983). Ce modèle prend en charge le raisonnement sur les erreurs humaines possibles et a été utilisé pour classer les types d'erreur. Reason (Reason, 1990) tire parti des contributions de Norman et Rasmussen, et distingue trois grandes catégories d'erreurs :

- L'erreur basée sur les automatismes.
- L'erreur basée sur les règles.
- L'erreur basée sur les connaissances.

Aussi, Reason a proposé un modèle de performances humaines appelé GEMS (Reason, 1987) (Generic Error Modelling System), qui est également basé sur le modèle SRK et dédié à la représentation des mécanismes d'erreur humaine. GEMS est un cadre conceptuel qui intègre une description détaillée des causes potentielles pour chacune des types d'erreur ci-dessus. Ces causes sont liées à divers modèles de performance humaine. Par exemple, une erreur de confusion perceptive dans GEMS est liée au processeur perceptif du Modèle de processeur humain (Card, Moran, & Newell, 1986).

2 Extension de la notation HAMSTERS pour la représentation des erreurs humaines

Dans chapitre 1 nous avons présenté l'état de la notation et l'outils HAMSTERS avant le début de la thèse. Ici nous allons présenter les extensions apportées à la notation HAMSTERS pour permettre d'identifier et de décrire les erreurs humaines dans les modèles de tâches.

2.1 Représentation des phénotypes et génotypes

Plusieurs éléments de notation ont été ajoutés à HAMSTERS pour permettre la représentation explicite des génotypes et phénotypes. Le Tableau 14 résume ces éléments qui peuvent être utilisés pour décrire une conséquence observable d'une erreur (phénotype) et ses causes potentielles associées (génotypes).

Dans le Tableau 14, la première colonne indique les types de génotypes d'erreurs suivant la classification GEMS (Reason, 1987). La deuxième colonne établit la connexion avec la classification SRK (Rasmussen, Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models, 1983) comme vu précédemment. La troisième colonne présente les nouveaux éléments de notation dans HAMSTERS, pour décrire les génotypes d'erreurs. Quatre nouveaux éléments sont ajoutés : Ratés, Lapsus, Erreurs basées sur les règles et des erreurs basées sur les connaissances (Reason, 1990). En ce qui concerne les phénotypes un seul élément de notation est proposé. En effet, le phénotype a seulement besoin d'être représenté explicitement, l'étiquette en dessous (voir Figure 22) fournit une description textuelle, et la relation avec les causes est faite en connectant les génotypes au phénotype. Ces connexions seront présentées dans les sections suivantes.

Tableau 11. Représentation du phénotype et génotypes avec la notation HAMSTERS

Type d'erreur (GEMS)	Niveau d'activité	Représentation du génotype dans HASMTERS	Représentation du phénotype dans HAMSTERS
Slip	Skill-based	Slip	
Lapse		Lapse	
Mistake	Rule-based	RBM	
	Knowledge-based	KBM	

2.2 Représentation des liens entre les génotypes et les tâches

Pour effectuer une tâche, l'utilisateur doit exécuter un ensemble d'actions ou de sous-tâches, et lors de l'exécution de ces dernières, une ou plusieurs erreurs (génotypes) peuvent se manifester. Ainsi, chaque sous-tâche dans le modèles de tâches peut être associé à un ou plusieurs génotypes d'erreurs. La connexion entre une sous-tâche et un génotype est représenté par un trait rouge fin entre la tâche et l'erreur. On peut voir ce lien dans Figure 22, entre la tâche cognitive « identify desired card » et le génotype de type Rule Based Mistake (RBM) « Misapplication of good rules ».

Figure 22. Lien entre un génotype et une tâche

2.3 Représentation de multiples causes (disjonction) et compositions de causes (conjonction)

Une erreur (phénotype) peut être à l'origine d'une ou plusieurs causes différentes (génotype(s)), ou bien d'une combinaison de plusieurs causes (génotypes). Avec l'exemple du DAB, si on appuie accidentellement sur une autre touche, ou si on répète le même chiffre après une interruption, le génotype associé est l'insertion d'un mauvais code PIN, qui sera à l'origine d'une erreur de (« mauvais code pin inséré »). Par contre avec l'exemple du crash du Vol 092 British Midland, le phénotype de l'erreur est le crash de l'avion. La cause de l'erreur est une combinaison de deux génotypes, qui sont : l'arrêt du mauvais réacteur, et l'omission après interruption du pilote par la tour de control pendant qu'il suivait la procédure de vérification des bons choix de gestion de panne, s'il aurait suivi toute sa procédure, il se serait rendu compte qu'il avait éteint le mauvais moteur, et il aurait eu le temps de le rallumer et d'éteindre le moteur défectueux.

Avec la notation HAMSTERS, pour représenter les causes possibles indépendantes ou combinés on utilise respectivement le OU (représenter par un rond plein rouge ●), et ET (représenter par un rond rouge avec un croix à l'intérieur ⊕). Ces icones permettent de lier un phénotype à une tâche, ainsi que les causes (génotypes) du phénotype (voir

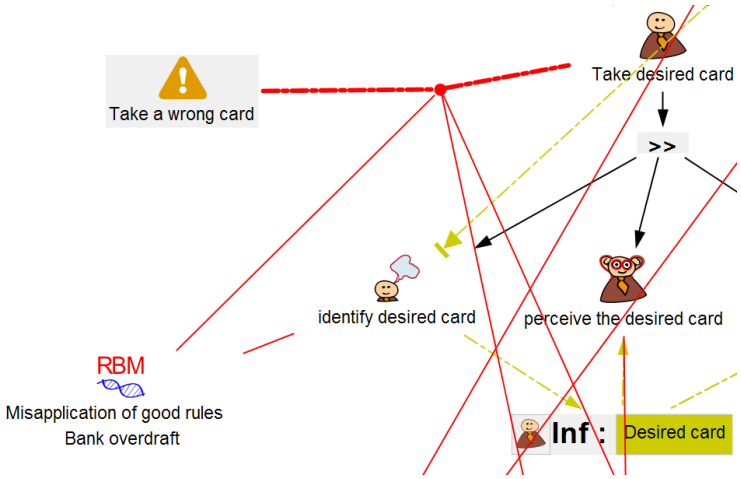


Figure 23. Connecteur OU entre phénotype, génotype et tâche

3 Extension de l'outil HAMSTERS pour l'édition des modèles prenant en compte les erreurs humaines

Dans la section précédente, on a vu les extensions apportées à la notation HAMSTERS pour représenter les erreurs humaines. Comme vue dans la section 3.2 du chapitre 1, la notation HAMSTERS est supporté par l'outil HAMSTERS. Une nouvelle version de l'outil a été créée afin d'intégrer des extensions qui permettent d'éditer des phénotypes et génotypes, leurs propriétés, ainsi que les connexions entres eux et les tâches associées.

3.1 Édition des phénotype et génotype

Comme vu dans le chapitre 1, l'outil HAMSTERS dispose d'une palette d'outils (voir Figure 15), contenant les différents éléments nécessaires à la modélisation de tâches. À cette palette on a ajouté une sous-catégorie « Human Errors » (voir Figure 24), elle contient tous les éléments liés à la représentation des erreurs humaines :

- Le phénotype : Représenté dans la palette par une l'icône ⚠ « Human Error », il est lié à une tâche et des génotypes par des connecteurs ET/OU (voir Chapitre 4.2.3)
- Le génotype : On peut représenter 4 types de génotype, qui sont représentés par une icône bleue, qui représente un gène en dessous du type du gène en texte (voir Tableau 14), les 4 types de génotypes sont :
  - Knowledge Based Mistake : Faute basée sur les connaissances
  - Rule Based Mistake : Faute basée sur les règles
  - Lapse Error : Lapsus basé sur l'automatisme
  - Slip Error : Raté basé sur l'automatisme

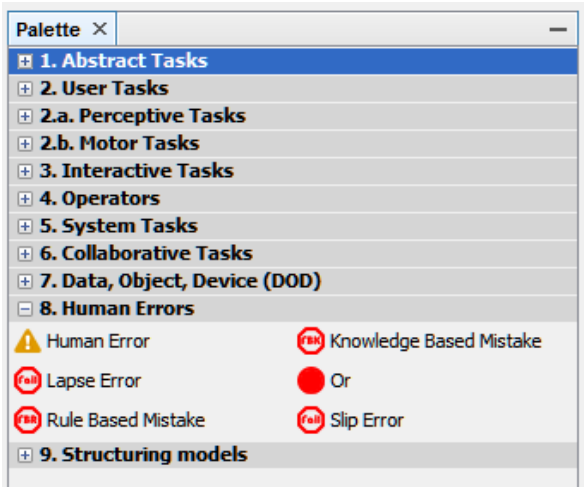


Figure 24. Éléments de la sous-catégorie "Erreur Humaine" dans la palette d'édition HAMSTERS

Pour utiliser ces nouveaux éléments, on procède de la même manière que pour les autres types d'éléments, on sélectionne l'élément souhaité, et on fait un glisser/déposer depuis la palette vers la scène d'édition. L'élément s'ajoute au modèle de tâches avec des attributs par défaut, chaque élément peut disposer d'attribut commun ou différent, c'est ce qu'on va voir dans la section suivante.

3.2 Édition des attributs

Chaque élément de l'outil HAMSTERS possède au moins un attribut, y compris les nouveaux éléments pour la représentation des erreurs humaines.

Le phénotype : a seulement besoin d'une description textuel, qui décrit l'action erronée dans son état observable.

Les génotypes : eux aussi ont besoin d'une description textuel, afin de décrire explicitement la cause de l'erreur, mais pas que, on peut attribuer un sous types de génotype, sois les sous types ajoutés par défaut dans l'outils, soit des sous types personnalisés par l'utilisateur (on verra l'édition des sous types de génotypes dans le Chapitre 4.3.4)

Pour modifier la description d'un phénotype ou d'un génotype, deux méthodes sont possibles, la première est de faire un double clic sur la description en bas de l'icône du phénotype ou génotype, ou bien sélectionner l'élément souhaité et aller dans la fenêtre de propriétés en bas à droite (voir Figure 15), et changer le texte dans le champ associé à la « Description » (voir Figure 25)

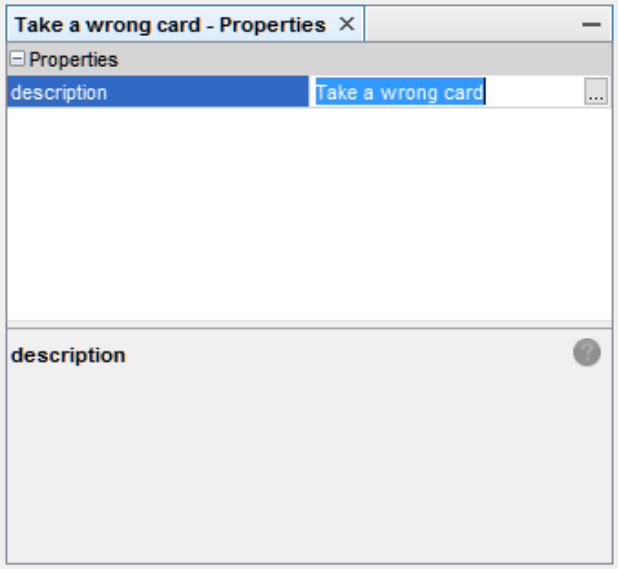


Figure 25. Fenêtre de propriété d'un élément HAMSTERS de type "Phénotype"

Pour le génotype, en plus de la description, possède un attribut pour raffiner le type du génotype, par défaut il est à indéfini « Undefined ». Afin de changer le sous-type du génotype, pareil que la description, deux méthodes sont possibles, la première qui est de double cliquer sur le texte du sous-type situé entre l'icône du génotype et la description (voir Figure 26 étape 1), le sous-type devient modifiable (voir Figure 26 étape 2), et en cliquant sur la flèche à droite du sous-type, une liste déroulante s'ouvre (voir Figure 26 étape 3), après on peut sélectionner le sous-type souhaité.

La deuxième méthode de changement de l'attribut du sous-type, pareil que pour la description, on utilise la fenêtre de propriété située en bas à droite. En cliquant dans la scène sur l'icône du génotype qu'on veut modifier, la fenêtre de propriété affiche les attributs du génotype sélectionné, en cliquant sur le champ associé au sous-type, une liste déroulante avec les différents sous-types possible s'affiche (voir Figure 27), et on sélectionne le sous-type désiré.

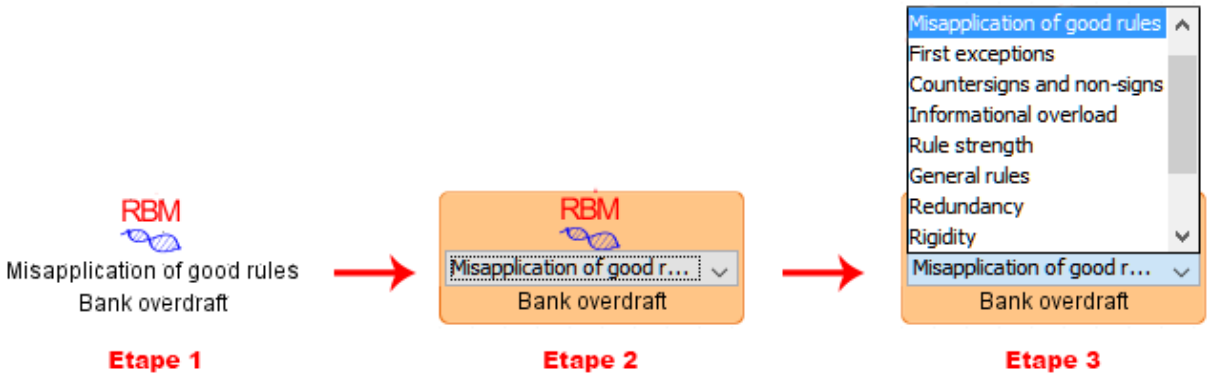


Figure 26. Séquence d'action pour changer un sous-type de génotype

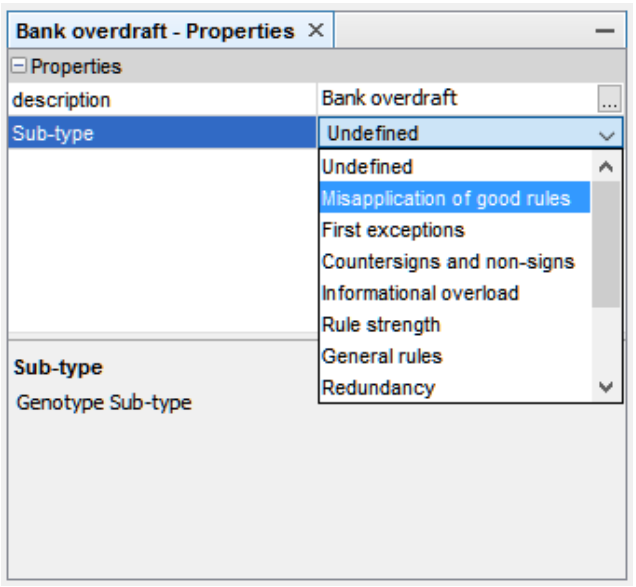


Figure 27. Fenêtre de propriété d'un élément HAMSTERS de type "Génotype"

3.3 Édition des connexions

Dans la notation HAMSTERS, chaque génotype est associé à la tâche qui peut entrainer à une cause d'erreur, cette connexion est représentée par une ligne rouge qui lie l'icône de la tâche à l'icône du génotype (voir C1 de la Figure 28) (dans la Figure 28, les textes C1,C2,C3 et C4 ne font pas partie du modèles de tâches, ils ont été ajouté seulement pour les différencier dans cette section), le génotype doit aussi être connecté à un opérateur d'erreur (ET/OU) aussi avec une ligne rouge (voir C2 de la Figure 28), ce même operateur doit relier à son tour le phénotype (l'erreur visible causée par le(s) génotype(s)) à la tâche associée, ces deux liens sont plus épais que ceux du génotype (voir C3 et C4 de la Figure 28).

Pour ajouter une connexion avec l'outil HAMSTERS, on utilise la même méthode que pour éditer les autres liens, en maintenant la touche CTRL (avec un PC) et tirer une ligne entre les deux éléments à connecter.

Et pour supprimer ces liens, un simple clic droit sur la ligne à supprimer, une popup menu s'affiche avec un item « Delete » (voir Figure 29), en cliquant dessus, la connexion s'efface.

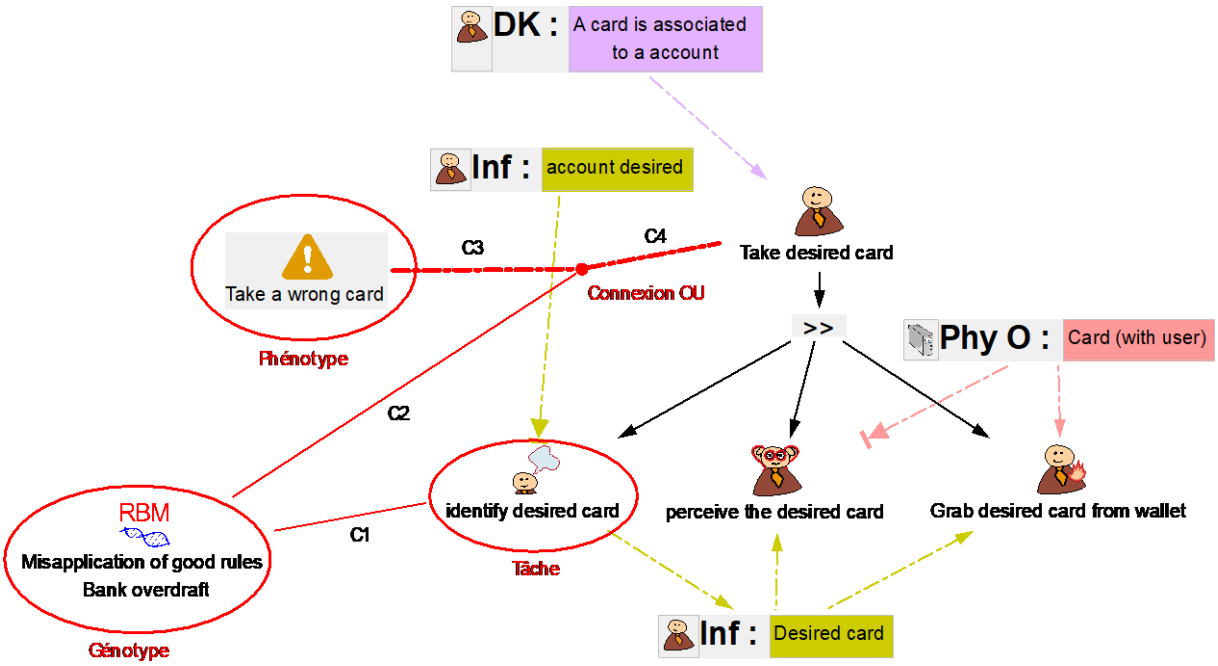


Figure 28. Représentation d'une branche du modèle de tâche du DAB, avec la représentation des connexions entre phénotype, génotype et tâche

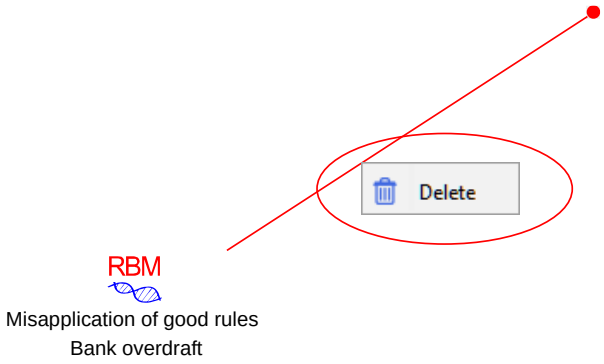


Figure 29. Popup menu des connexions de génotypes

3.4 Edition des sous types de génotype

Comme vue dans le Chapitre 4.1.3, la notation HAMSTERS permet la représentation de types de génotypes classifier selon la taxonomie de Reason (Reason, 1990) en distinguant les 3 niveaux d'activité de Rasmussen (Rasmussen, Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models, 1983), dans lesquelles l'erreur se produit. On retrouve les erreurs basées sur l'automatisme, et représentées dans l'outil HAMSTERS par « raté » ou « lapsus », les erreurs basées sur les connaissances, ainsi que les erreurs basées sur les règles.

Pour chaque type de génotype, on a repris les sous-types de Reason, et rajouté une possibilité de mettre le sous-type en indéfini (voir Figure 30). Lors d'un rajout d'un génotype dans le

modèle de tâches par un glisser/déposer, le sous-type du génotype est initialisé à indéfini, pour le modifier voir le Chapitre 4.3.2.

La fenêtre de gestion des sous-types de génotype, est accessible à partir de l'outil HAMSTERS, en cliquant sur l'icône représentant un gène de couleur bleu, et situer en haut dans la barre d'outils (voir Figure 31). Cette fenêtre de gestion permet la visualisation et l'édition des sous-types de génotype et aussi la gestion des correspondances entre les génotypes et les types de tâches, et accessible en cliquant sur « Correspondence Table », on le verra plus en détails dans le Chapitre 5. La partie de gestion des sous-types, permet de rajouter (voir Figure 32), modifier ou supprimer (voir Figure 33) des sous-types personnalisés. Les sous-types du modèle GEMS (Reason, 1987) ne peuvent pas être modifier ou supprimer (voir Figure 34).

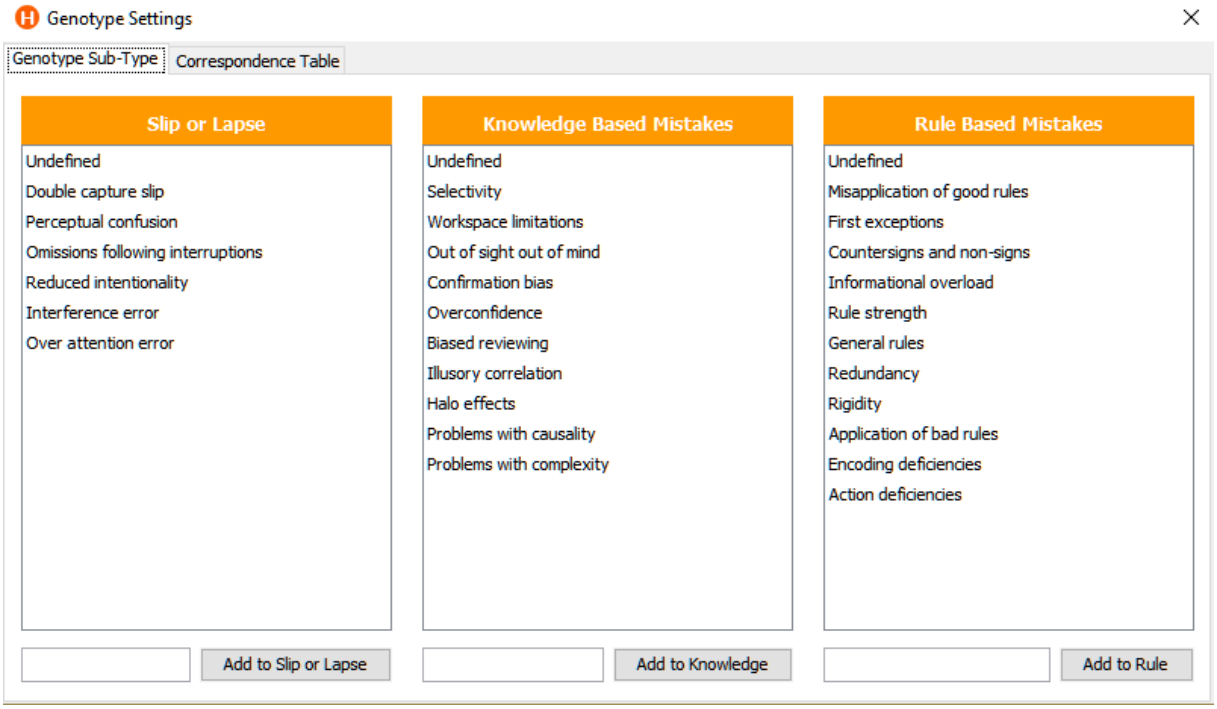


Figure 30. Fenêtre de gestions des sous-types de génotype avec l'outil HAMSTERS

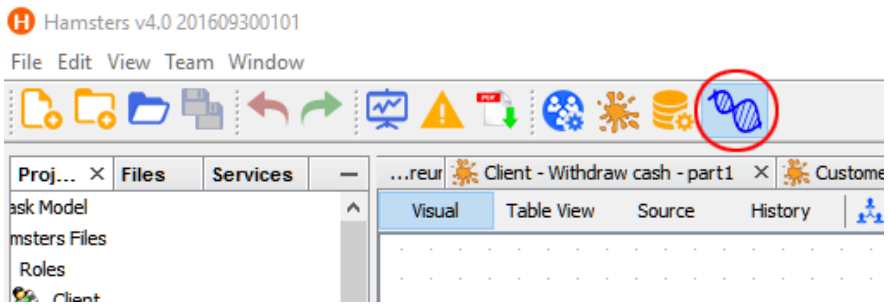


Figure 31. Barre d'outil HAMSTERS, montrant le bouton d'ouverture de la fenêtre de gestion des sous-types de génotype

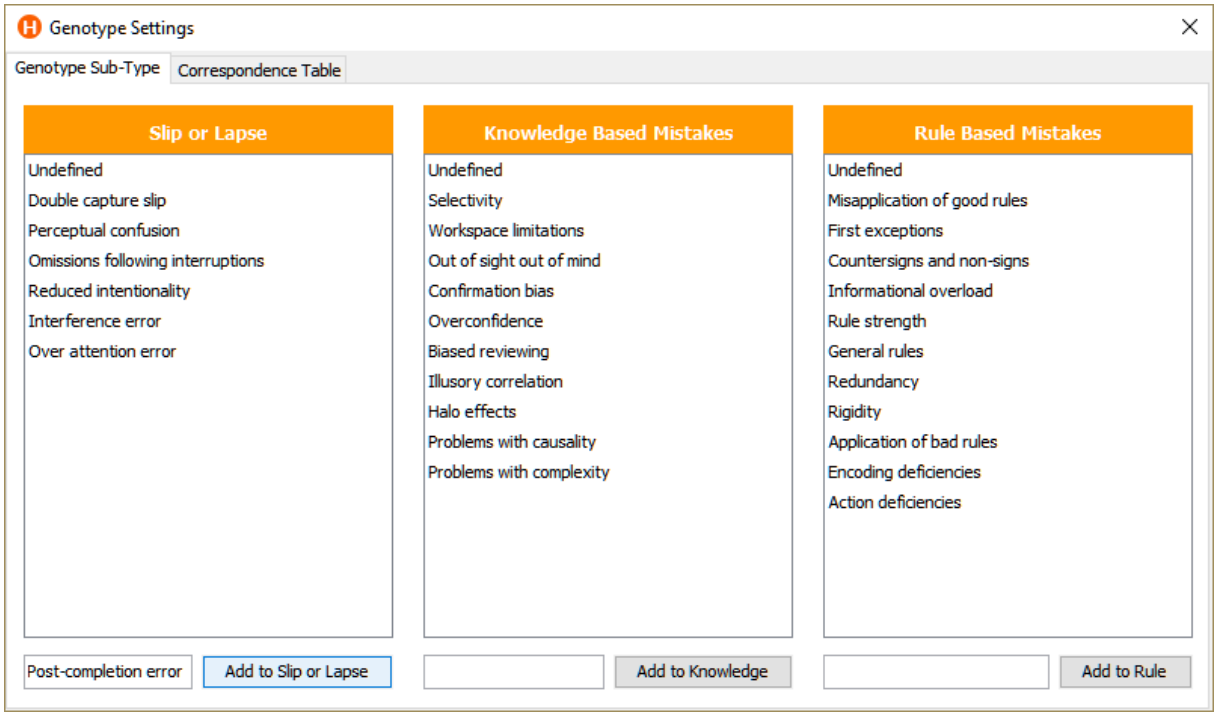


Figure 32. Ajout d'un nouveau type de Slip/Lapse dans la fenêtre de gestion des génotypes

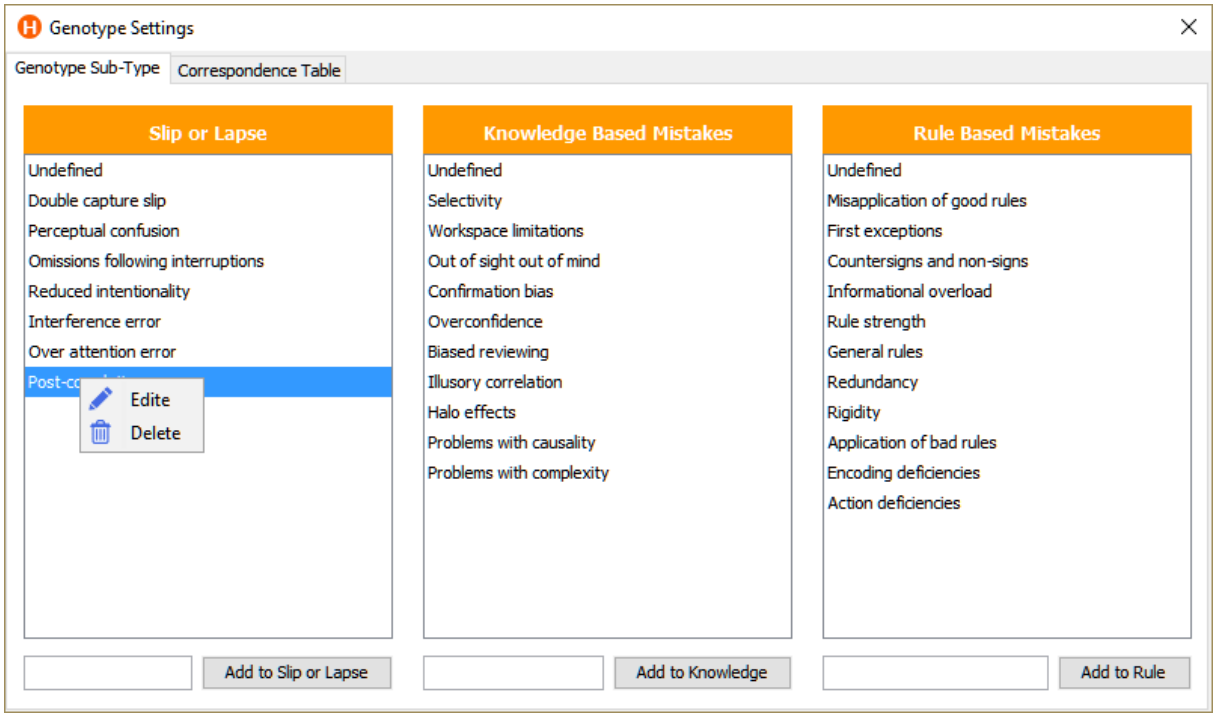


Figure 33. Popup menu d'un sous-types personnalisé de génotype dans la fenêtre de gestion des génotypes

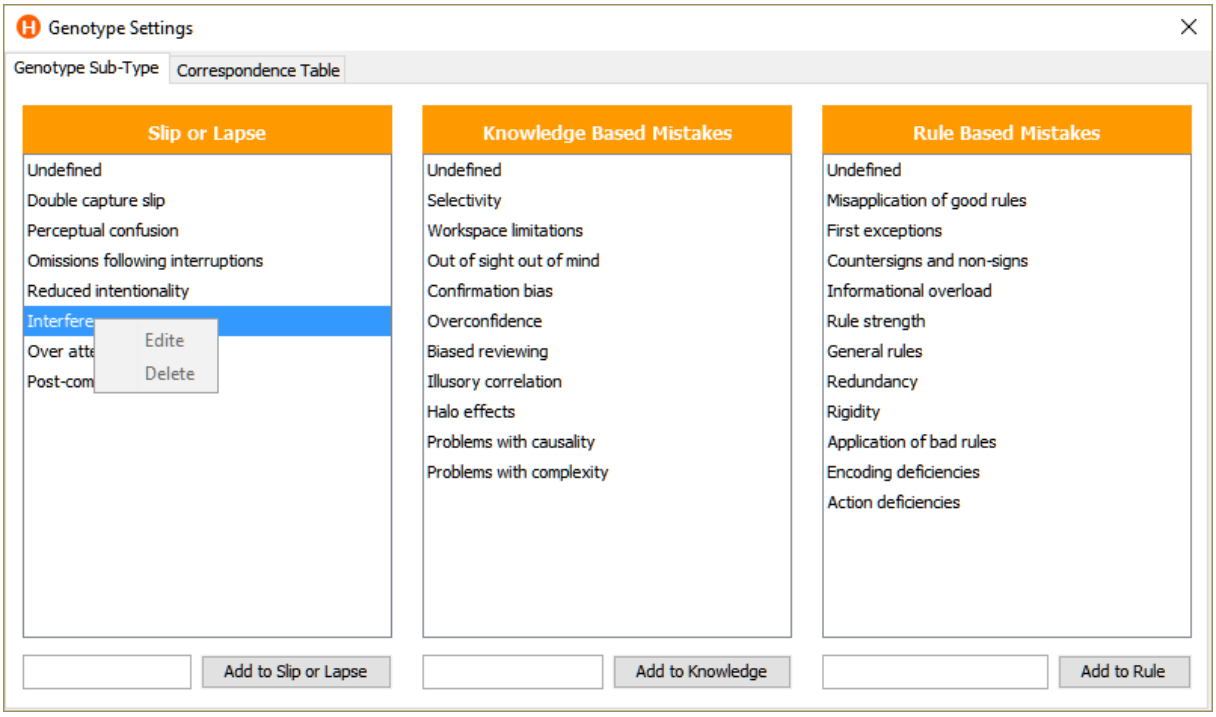


Figure 34. Popup menu d'un sous-types de génotype dans la fenêtre de gestion des génotypes

3.5 Fonctionnalités d'aide à la visualisation des modèles qui intègrent des erreurs humaines

Les modèles de tâches enrichis d'erreurs humaines se voient rajoutés encore plus de connexions qui rendent le visuel du modèle de tâches presque illisible (voir les liens rouges dans la Figure 35). Pour remédier à ce problème, on a implémenté un système de filtres basé sur les calques, et qui permet d'afficher ou de masquer les éléments qu'on souhaite voir sur le modèle de tâche, pour cela, dans la partie navigation dans l'outil HAMSTERS (vu dans la section 3.2 du chapitre 1) on sélectionne le type d'éléments qu'on veut afficher ou masquer (Phénotype ou Génotype), puis on va dans la partie propriété, et on coche ou on décoche les éléments souhaités.

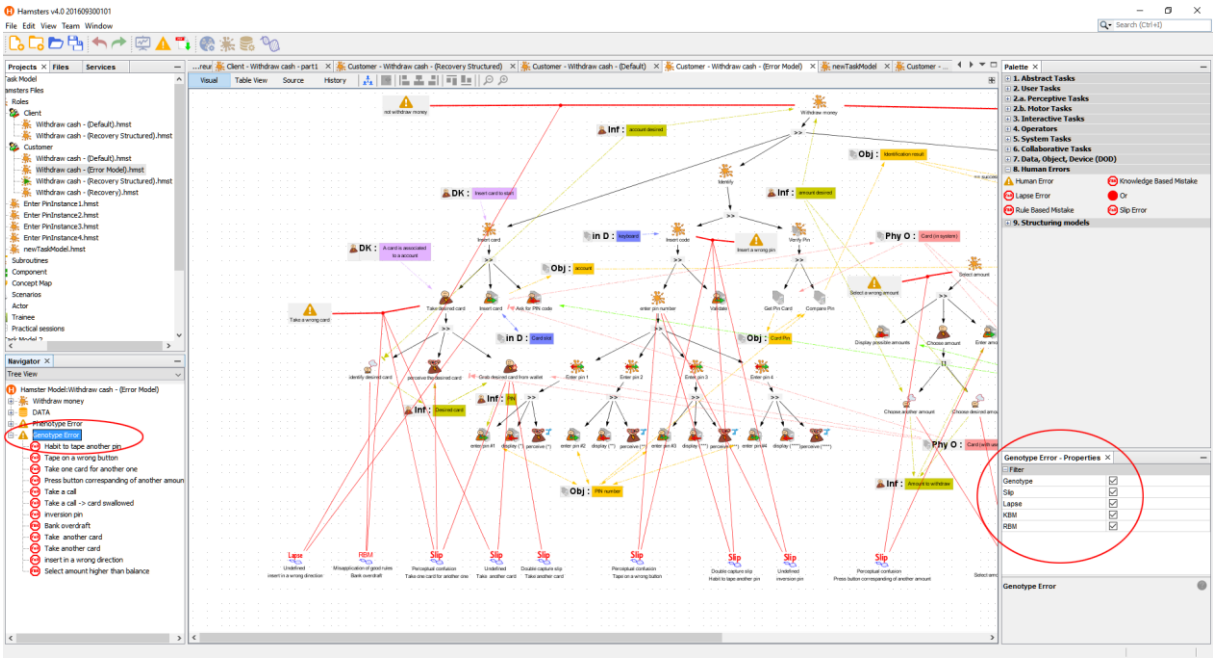


Figure 35. Utilisation des filtres pour masquer et afficher les phénotypes et génotypes dans les modèles de tâches avec l'outil HAMSTERS

4 Exemple avec l'étude approfondie de l'utilisation du DAB

Pour illustrer toutes ces notions de représentation d'erreurs dans les modèles de tâches, on va utiliser comme exemple le retrait d'argent à partir d'un distributeur automatique de billet (DAB). On a choisi le DAB, car c'est un exemple assez simple à comprendre et que la plupart l'ont déjà utilisé, et aussi pour son évolution historique qui au fur des années a apporté plus de fonctionnalités afin de permettre aux utilisateurs de reprendre leurs erreurs. Parmi ces fonctionnalités, on retrouve le bouton cancel qui permet d'annuler la tâche en cours, le bouton corriger qui permet de corriger son code pin, la possibilité de retaper son code pin, etc…

Nous allons commencer par présenter l'historique du DAB avec les différents problèmes de conception, ainsi de son état actuel. Puis nous allons décrire le processus global qui inclut la partie identification et la partie analyse de la criticité, ainsi que leurs intégrations dans l'outil HAMSTERS. Et on finira par une étude de cas du DAB.

4.1 Présentation du DAB

Le distributeur automatique de billet (DAB) tel que l'on connait à ce jour, n'est pas une invention d'une seule personne, mais est le résultat de séries d'inventeurs. Contrairement à ce que l'on peut croire, le concept de GAB est apparu avant le DAB. L'un des premiers inventeurs du concept du GAB est Luther George Simijian qui a commencé à déposer des brevets dès 1939, et en 1960 il a réussi à convaincre la New York City Bank d'installer ses GAB au sein de leurs agences, ces GAB permettaient seulement de faire des dépôts, au bout de 6 mois la banque retire ses automates à cause de la faible demande des clients. Après dans les débuts des années

1960, John Shepherd-Barron travaillant pour le compte de « De La Rue Instruments », il conçoit le premier DAB utilisable 24h/24 et 7jour/7. Le premier DAB a été installé à l'agence Barclays à Londres le 27 juin 1967[3], il utilisait une carte magnétique à usage unique, que l'utilisateur récupérer auprès de son agence, avec un code à 4 chiffres. Et pour retirer son argent, il devait introduire sa carte dans l'automate, taper son code personnel, puis il récupérer son argent, mais la carte reste stockée dans l'appareil.

Ces premiers DAB tel que le premier DAB installé en France (voir Figure 36) contenaient que 9 touches, les chiffres de 0 à 9, dès que le client introduisait 4 chiffres, le système vérifiait la validation du code, si le code est bon, les billets sont distribués un à un, et si le code était incorrect, rien ne disait que le code était faux, et la carte resté dans le distributeur. L'utilisateur n'avais le choix que de repartir demander une autre carte à son agence.

Avec ce DAB, le système marche bien, mais seulement si l'utilisateur est parfait et qu'il ne fait pas d'erreur. Mais l'utilisateur peut faire dans différent cas des erreurs :

- Si l'utilisateur oublie son code, il ne pourra pas finir sa tâche
- Si l'utilisateur se trompe en tapant son code, par exemple au lieu de taper sur « 1 » il tape sur « 2 », même s'il s'en rend compte avant de finir de taper son code, il ne pourra rien faire, et il ne pourra pas accomplir sa tâche
- Si l'utilisateur au cours de l'exécution de sa tâche décide d'annuler sa tâche, et la reporter ultérieurement, il ne pourra pas, une fois la carte introduite, pas possible de la récupérer.
- Dernière supposition, si l'utilisateur ne compte pas le nombre de billets reçu, et vu que le DAB distribue les billets un par un, le client risque de partir avant de récupérer la totalité de ses billets.

Ces problèmes de conception, ont été corriger avec le temps, jusqu'à arriver au guichet automatique de banque qu'on connait de nos jours.

[3] http://news.bbc.co.uk/2/hli/business/6230194.stm

Figure 36. Premier distributeur automatique de billet de France en 1968 par la société marseillaise de crédit[4]

Les distributeurs automatiques de billets comme on les connais aujourd'hui (voir Figure 37), nous permettent en plus de la fonction de retirer de l'argent, ils nous permettent d'effectuer des virements, de consulter le solde, de déposer des billets, de commander un chéquier, d'imprimer un relevé d'identité bancaire (RIB), etc …

Ces automates sont appelés GAB (Guichet Automatique Bancaire), ils viennent remplacer des DAB, mais dans certains cas, le GAB ne peut que proposer la fonction du DAB, qui est de retirer de l'argent, tout dépend du type de la carte bancaire qu'on utilise, par exemple, avec une carte de retrait, on peut que retirer de l'argent, et seulement dans un automate de la banque émettrice de la carte, ou bien l'utilisation d'une carte bancaire ne peut permettre que le retrait d'argent à partir d'automate d'autres banque.

Les fonctionnalités et services proposés par les GAB diffèrent d'une banque à une autre, et le seul service commun proposé par toute les banques, c'est le retrait d'argent. La séquence d'actions permettant de retirer de l'argent peut différer aussi d'une banque à une autre, et même d'un automate à un autre, on peut sélectionner le montant avant de s'identifier, comme l'on peut s'identifier puis sélectionner le montant, mais les principaux sous fonctions sont les mêmes, on insère une carte bancaire, on s'identifie, on sélectionne un montant, et on reprend la carte et les billets.

[4] http://www.ina.fr/video/CAF97059697

Figure 37. Exemple du Guichet Automatique de Banque (GAB) actuel

4.2 Présentation de la tâche « Retirer de l'argent »

La tâche « retirer de l'argent » est une opération par laquelle un client retire de son compte, soit à un automate (DAB, GAB), soit au guichet de sa banque ou de son établissement de paiement une certaine somme en espèces, et avec la possibilité de ré exécuter la même tâche. Le retrait d'argent à partir d'un DAB/GAB est une interaction entre l'utilisateur et le système (Homme/Machine), contrairement au retrait à partir d'un guichet, où l'interaction est entre le client et l'agent (Homme/Homme).

4.2.1 Actions à exécuter pour réaliser la tâche « Retirer de l'argent »

Pour comprendre mieux la tâche de retrait d'argent, nous allons dérouler un scénario d'un utilisateur (voir Scénario 1), c'est-à-dire qu'il ne fera pas d'erreur d'utilisation, et nous supposant que l'utilisateur possède un compte bancaire, et une ou plusieurs cartes bleues, et qu'il se trouve devant un automate.

1. ***Pour commencer*** *son opération****, il sait qu'il doit introduire sa carte bancaire associée*** *au* ***compte désiré*** *;*
   Il retire ***sa carte*** *qui est rangé dans son portefeuille parmi d'autres cartes, et l'introduit dans la* ***fente à carte****, après un bref instant, le DAB demande au client de s'identifier en lui affichant un message à* ***l'écran*** *;*
2. *Il commence à taper* ***son code PIN*** *à l'aide du* ***clavier****, à chaque appui sur une touche associée à un chiffre,* ***l'écran*** *affiche* ***une/des étoile(s)*** *au nombre de fois appuyées sur une touche, puis fini par appuyer sur la touche valider ;*
   Le système vérifie que le code PIN entré par le client est identique à celui enregistré sur sa carte bancaire ;
3. *L'identification est réussie, le DAB propose au client les montant possible à retirer, si le* ***montant souhaité*** *est proposé, le client le sélectionne, sinon il choisit* ***un autre montant*** *;*
4. *Le DAB prépare* ***l'argent****, en débitant* ***le compte sélectionné****, un message s'affiche pour demander au client de prendre sa carte bancaire, le client la prend et la remet dans son portefeuille, un autre message s'affiche pour inviter le client à récupérer son argent, que se dernière prend à partir de* ***la fente à billets****.*

Scénario 1. Un client retire de l'argent à partir d'un DAB sans commettre d'erreur

À partir du Scénario 1 on peut identifier 4 sous but par rapport au but principal qui est de retirer de l'argent, en reprenant respectivement les paragraphes numérotés, on peut les résumer avec ce qui suit :

1. ***Introduction de la carte (voir tâche 1 de la*** Figure 38***):***
   L'utilisateur choisi sa carte et l'introduit dans l'automate.
2. ***Identification du client (voir tâche 2 de la*** Figure 38***):***
   L'utilisateur introduit le code pin associé à sa carte.
3. ***Sélection du montant (voir tâche 3 de la*** Figure 38***) :***
   L'utilisateur sélectionne le montant souhaité.
4. ***Récupération de la carte et l'argent (voir tâche 4 de la*** Figure 38***):***
   L'utilisateur récupère sa carte et son argent.

La Figure 38 représente le modèle de tâches abstrait du DAB, où l'on peut identifier les 4 sous-buts principaux vu en dessus. Le but principal est représenté par une tâche abstraite (voir section 3.2 du chapitre 1) complètement en haut de la figure, et qui est « retirer de l'argent » (Withdraw money), ce but est décomposé en plusieurs branches. La première branche qui est « insérer la carte » (Insert card) (voir tâche 1 de la Figure 38), permet de décrire les actions à effectuer pour insérer une carte dans le DAB. La branche suivante qui est « S'identifier » (Identify) (voir tâche 2 de la Figure 38), s'effectue en séquence après la première tâche, et permet de décrire les actions nécessaire pour identifier la carte, après l'exécution de cette tâches deux possibilités s'offrent à l'utilisateur, soit l'identification échoue et l'utilisateur passe dans la branche « Identification Fail », soit elle réussit et il passe dans la branche « Identification Success », dans cette dernière l'utilisateur va sélectionner un montant « Select amount » (voir tâche 3 de

la Figure 38), puis le système va vérifier si le solde est supérieur ou égal au montant sélectionné, si c'est le cas, l'utilisateur passe dans la branche « Verification account Success », où le système distribue le montant sélectionné « Pack money », puis l'utilisateur finalise sa tâche « Finalize withdraw » (voir tâche 4 de la Figure 38), sinon il passe dans la branche « Verification account fail ».

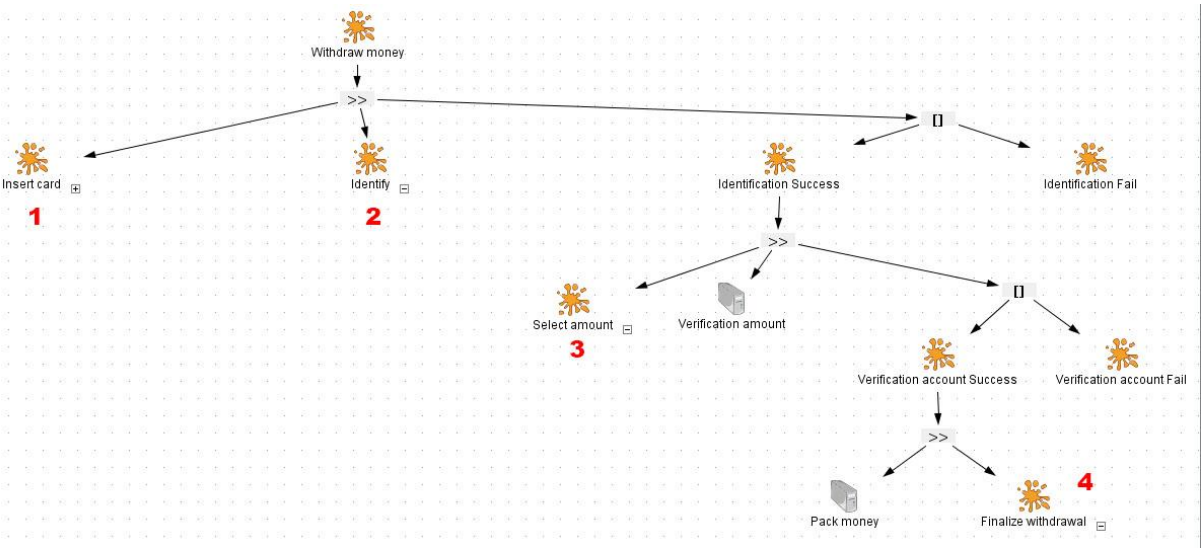


Figure 38. Modèle de tâches abstrait du DAB sans la représentation des erreurs humaines et des objets

Les numéros de 1 à 4 dans le modèle de tâches correspondent aux sous tâches qui permette de réaliser les 4 partie du Scénario 1. Nous allons maintenant voir plus en détails les différentes branches du modèle :

1) Introduction de la carte :

La Figure 39 représente la sous tâche « introduire la carte » (**Insert Card**) du modèle de tâches « Retirer de l'argent ». Pour réaliser cette sous tâche, l'utilisateur dois savoir qu'il faut insérer la carte pour commencer sa tâche, cette connaissance est représentée par une connaissance déclarative (DK) en couleur violet dans le modèle de tâche avec comme description « **DK : Insert card to start** ».

Sous la sous tâche « **Insert card** », on a un opérateur de séquence et en dessous 3 sous tâches qui doivent être exécuté en séquence :

Prendre la carte désirée (**Take desired card**), pour exécuter cette tâche, l'utilisateur doit savoir qu'une carte bancaire est associée à un compte bancaire (voir connaissance déclarative « **DK : A card is associated to a account** »). L'utilisateur exécutera en séquence 3 actions, il va identifier la carte désirée par rapport au compte bancaire désiré « **Identify desired card** » (qui est une tâche cognitive), cette tâche va produire une information « carte désirée » (**Desired card**) dans la tête de l'utilisateur, qui sera utilisé par la tâche « percevoir la carte désirée » (**perceive the desired card**) et par la tâche « Retirer la carte désirée du portefeuille » (**Grap desired card from wallet**). La tâches « percevoir la carte désirée » est une tâche de perception, et elle a besoin en entré un objet physique qui est la carte (**Phy O : Card (with user)**) en rose dans le modèle de tâches), le « with user » dans la description de l'objet permet de dire que la

carte est chez l'utilisateur. Après avoir perçu la carte, l'utilisateur va retirer la carte désirée du portefeuille (**Grap desired card from wallet**) pour l'introduire dans le système à l'aide du périphérique d'entré « fente à carte » (**in D : Card slot**) (voir la tâche interactive d'entrée « **Insert card** », cette tâche récupère la carte de chez l'utilisateur « **Phy O : Card (with user)** » pour la mettre dans le système et qui devient « **Phy O : Card (in system)** », cette tâche produit aussi un objet dans le système qui est le compte associé à la carte (**Obj : Account**). Après l'introduction de la carte, le système va demander à l'utilisateur d'introduire son code PIN (voir la tâche interactive de sortie « **Ask for PIN code** ») en lui affichant un message sur l'écran (qui est un périphérique de sortie « **out D : Display** »).

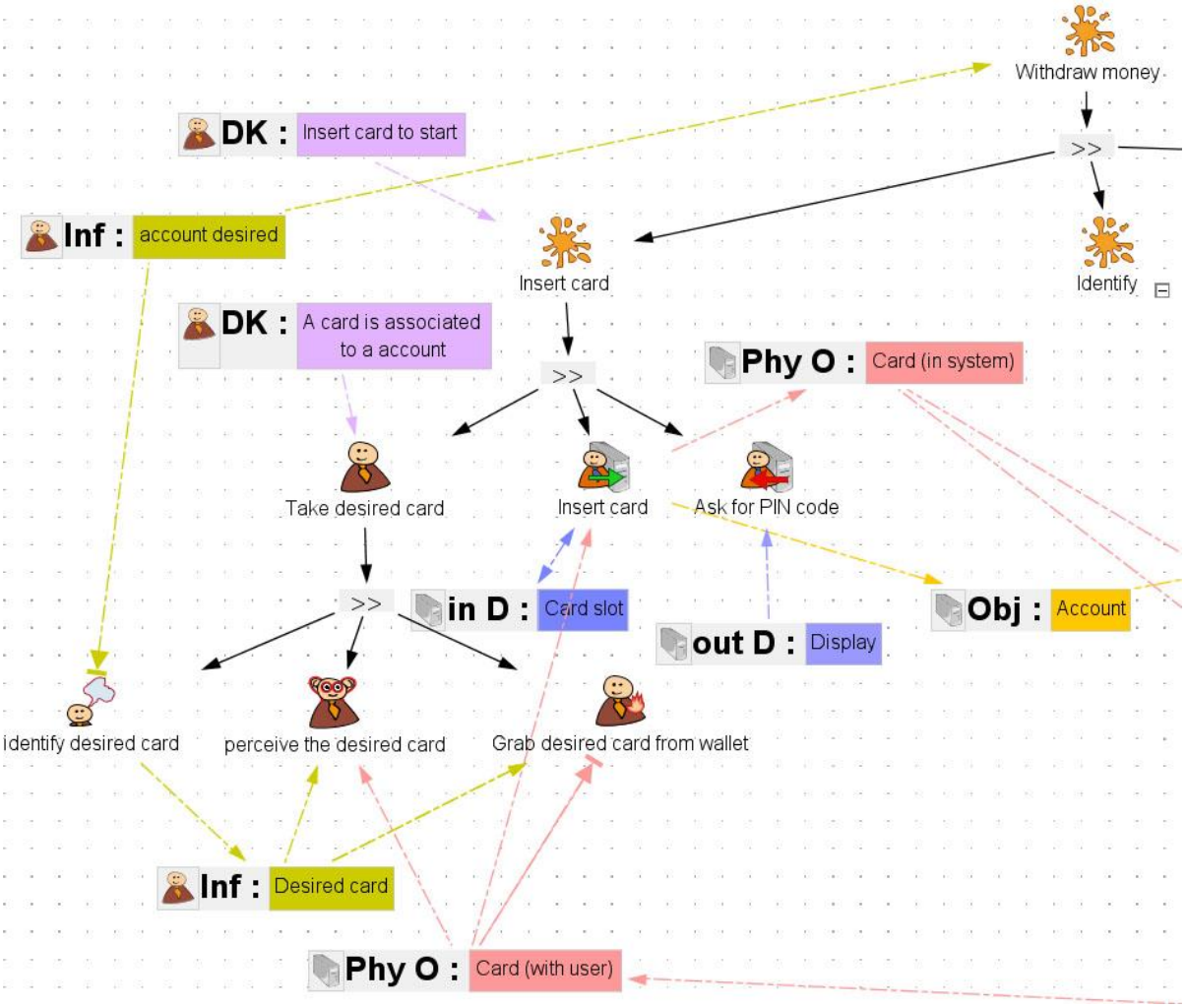


Figure 39. Sous tâche d'insertion de la carte (Insert Card) de la tâche "retirer de l'argent"

2) Identification du client :

Après avoir introduit la carte, l'utilisateur doit s'identifier, pour cela il passe dans la branche « **Identify** » (voir Figure 40), en dessous on a une branche, on a un opérateur de séquence avec deux tâches, une tâche abstraite « **Insert code** » puis une tâche système « **Verify Pin** ». Dans la branche « Insert code » l'utilisateur va introduire les chiffres qui constituent son code Pin, sous cette tâche, y a un opérateur de désactivation entre la tâche abstraite interactive « **Enter PIN** » et la tâche interactive d'entrée « **Validate** », la branche « Enter PIN » est itérative

(comme l'on peut le voir avec l'icône d'une flèche en rond à gauche de l'icône de la tâche abstraite interactive « Enter PIN »), cela veut dire que l'utilisateur va exécuter cette branche jusqu'à ce qu'il exécute la tâche « Validate ». La branche « Enter PIN » décrit la séquence d'action à effectuer par l'utilisateur pour entrer son code Pin, il commence par la tâche interactive d'entrée « Enter PIN » qui à l'aide du clavier du DAB « in D : Keyboard » et utilisant l'information du code pin qui est dans la tête de l'utilisateur « Inf : PIN » permet de modifier l'objet pin « Obj : PIN » qui est une instance dans le système, en concaténant à chaque fois la valeur associée à la touche tapée, puis le système affiche à l'utilisateur sur l'écran « out D : Display » le nombre de chiffre que contient l'objet « Obj : PIN », avec une représentation d'étoile (par exemple : si l'utilisateur a tapé { 4,3,2 }, l'objet « Obj : PIN » aura comme valeur « 432 » et le message afficher à l'écran sera « *** »), que l'utilisateur peut apercevoir avec la tâche de perception « Perceive » et qui n'est pas obligatoire (voir l'icône des deux flèches bleu à droite de la tâche « Perceive »), si l'utilisateur l'execute, elle va produire une information dans sa tête sur le nombre de chiffres entrés (voir « Crypted PIN »). Après avoir fini de taper le code Pin, l'utilisateur valide son code Pin avec la tâche « Validate », en appuyant sur la touche « valider » du clavier « in D : Keyboard ». Après validation le système va prendre l'instance « Obj : PIN » et la comparer avec le code Pin mémorisé dans la carte bancaire « Phy O : Card (in system) », et mettre le résultat de cette comparaison dans une instance du système « Obj : Identification result », si le résultat est égal à « true » l'utilisateur passe dans la branche « Identification success », si c'est « false » il passe à la branche « Identification fail » (voir Figure 38).

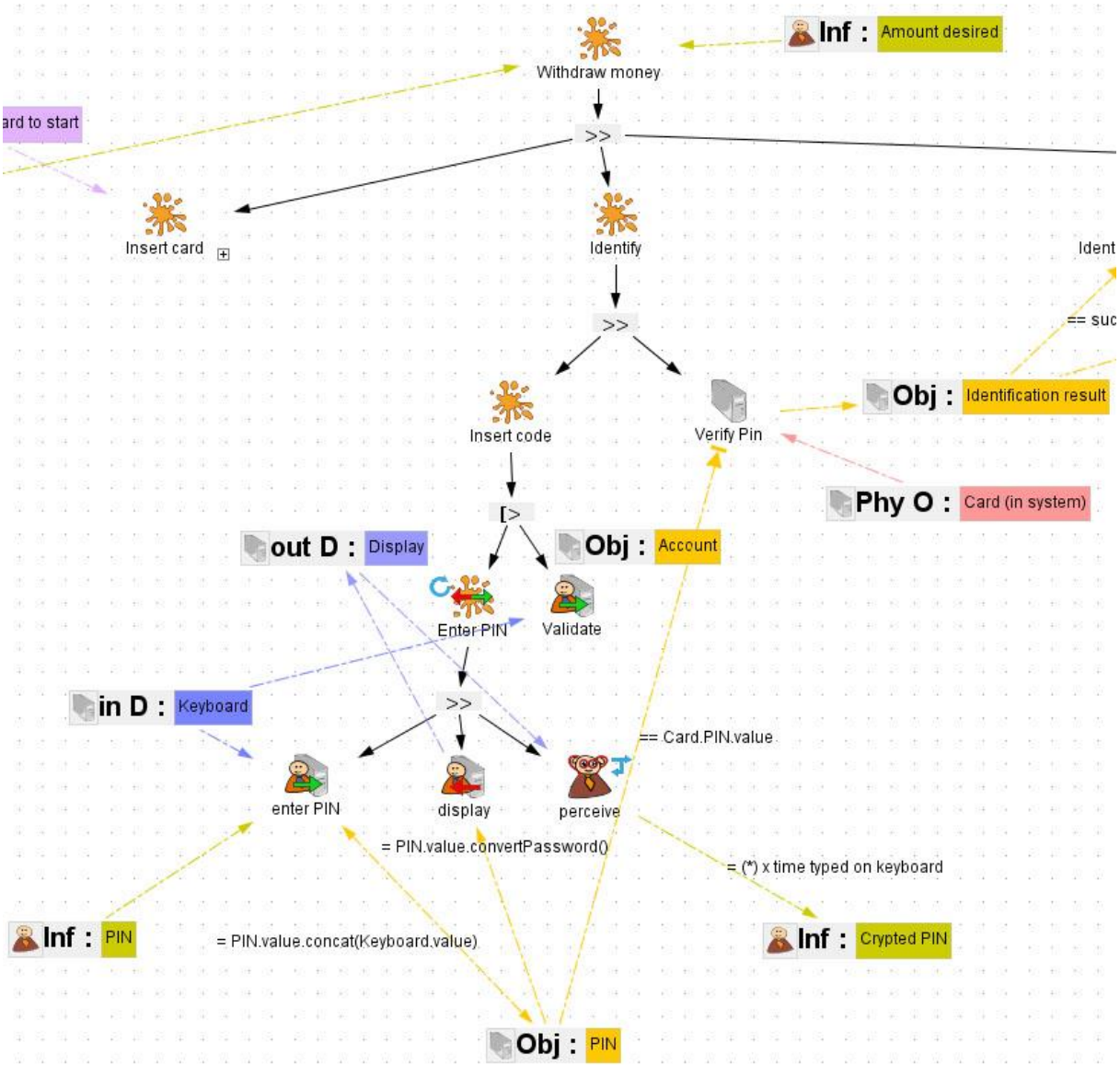


Figure 40. Sous tâche d'identification du client (Identify) de la tâche "retirer de l'argent"

3) Sélection du montant :

Après que l'utilisateur ait réussi l'identification de sa carte bancaire, il aura accès à la sélection du montant désiré à retirer, pour cela il va exécuter la branche « Select Amount » (voir la Figure 41). En séquence, le système lui affiche sur l'écran (« out D : Display ») les différents montant possible à retirer, puis il aura à faire un choix entre choisir le montant désiré initial « Inf : Amount desired » (voir la tâche cognitive « **Choose desired amount** »), ou de changer le montant par rapport aux montants proposés (voir la tâche « **Choose another amount** »), et en sortie de cette décision, l'utilisateur aura l'information du montant à retirer dans sa tête « Inf : Amount to withdraw », à partir de cette information, il va appuyer sur la touche associée au montant à retirer (voir la tâche interactive d'entrée « Enter amount », qui va instancier un objet « Obj : amount » dans le système, afin que le système vérifie si le solde du compte « Obj : Account » est suffisant, si c'est le cas, le système va instancier un objet de type booléen « Obj : Verification result » avec comme valeur « success », sinon avec la valeur « fail ».

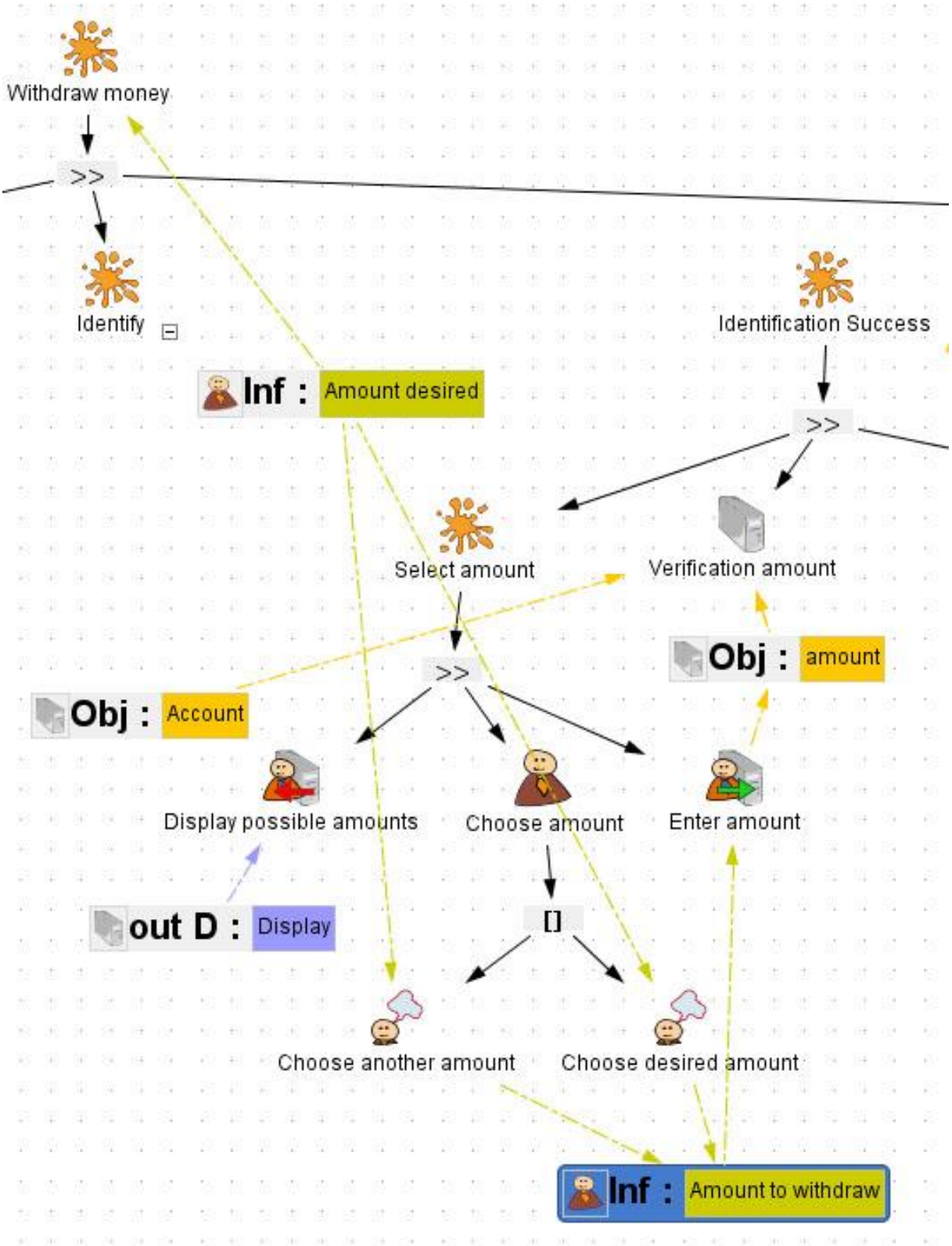


Figure 41. Sous tâche de sélection du montant (Select amount) de la tâche "retirer de l'argent"

4) Récupération de la carte et l'argent :

Après avoir sélectionner le montant, le système vérifie la disponibilité du solde, et si le solde est inférieur au montant sélectionner, on passe dans la branche échec de la vérification (voir la tâche abstraite « Verification account Fail » dans la Figure 42), sinon on passe dans la branche

solde vérifié (voir la tâche abstraite « Verification account Success »), entre ces deux branches, il y a un opérateur de choix, qui permet d'activer et de désactiver les branches par rapport à la valeur de l'objet « Obj : Verification result ». On présentera ici que la branche « Verification account Success », quand l'utilisateur arrive à cette sous tâche, le système va préparer les billets associé au montant « Obj : amount » à partir des billets disponible dans le DAB « Phy O : Disponible money », afin de les sortir à l'utilisateur « Phy O : packed money », à ce moment-là, l'utilisateur va finaliser sa tâche « Finalize withdraw » reprenant sa carte bancaire « Phy O : Card (with user) », qui était dans le système « Phy O : Card (in system) » (voir la tâche interactive « Take card »), pour la remettre dans son portefeuille « Put card in wallet ». Puis le système affichera un message à l'utilisateur en lui demandant de prendre son argent (voir tâche interactive de sortie « Display 'Take your money' ») en même temps que l'argent « Phy O : packed money » sorte de la fonte à billet «out D : Money slot » (voir la tâche interactive de sortie « Release money »). Et pour finir sa tâche et atteindre son but, l'utilisateur va récupérer son argent « Phy O : packed money » (voir la tâche motrice « Take money »).

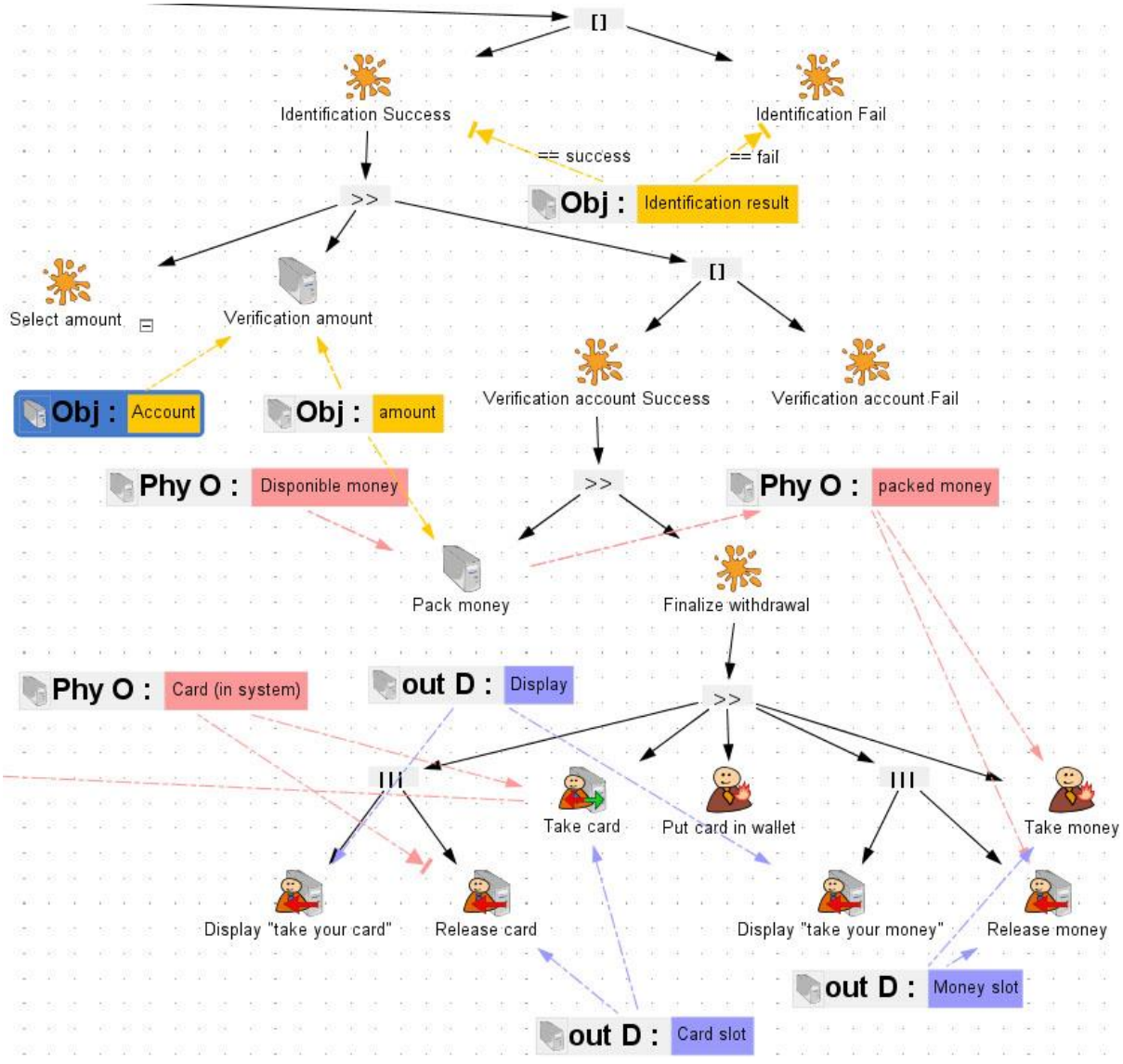


Figure 42. Sous tâche de récupération de la carte et l'argent (Finalize withdraw) de la tâche "retirer de l'argent"

La Figure 43 représente le modèle de tâches « Retirer de l'argent » dans sa globalité.

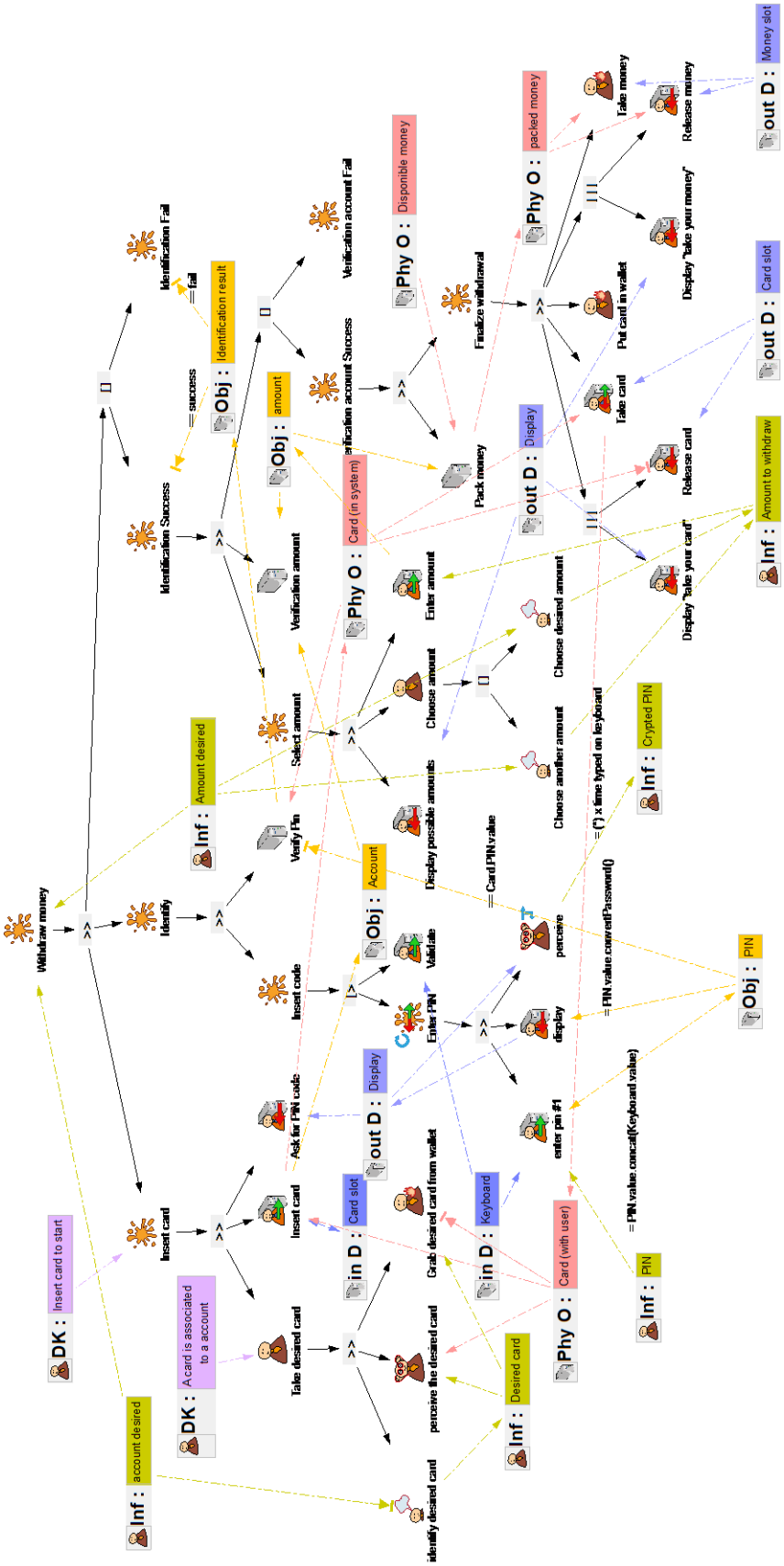


Figure 43. Modèle de tâches « Retirer de l'argent » avec la description d'objet, donnée et périphérique manipulés

4.2.2 La représentation des phénotypes/génotypes dans le modèle de tâches « Retirer de l'argent »

Après une description de tâche utilisateur, une phase d'analyse d'erreur humaine peut être appliquer au modèle de tâches, afin d'identifier les erreurs humaines associées à chaque tâche, et de les représenter sur le modèle de tâches, on l'appellera le modèle de tâches enrichit avec des erreurs humaines, cette identification se fait à l'aide d'un processus d'identification systématique qu'on détaillera dans le chapitre suivant.

La Figure 44 représente le modèle de taches enrichit d'erreurs humaine de la tâche « Retirer de l'argent », comme l'on peut voir, le modèle de tâche est le même que celui de la Figure 43, sauf que sur celui-ci, on a représenté les erreurs humaines, on a en tout 6 phénotypes et 12 génotypes, les différents phénotypes sont :

- Ne pas avoir retirer de l'argent
- Retirer une somme différente que celle souhaitée
- Se tromper sur le code PIN de la carte
- Insérer une carte non bancaire
- Ne pas avoir récupérer la carte
- Et ne pas avoir récupérer l'argent

Ces Phénotype sont causés par différent génotypes, prenons comme sous exemple le phénotype « Take a wrong card » (« prendre la mauvaise carte »), ce phénotype peut être causé par :

- Une confusion perceptuelle, deux cartes avec le même visuel peut causer cette confusion, ce génotype se manifeste lors de la séquence d'actions « percevoir la carte dans le portefeuille » et « retirer la carte du portefeuille ».
- Avoir l'habitude de retirer avec une autre carte, et par réflexe, l'utilisateur sort sa carte habituelle, alors qu'il voulait utiliser une nouvelle, ce génotype se manifeste lors de la tâche motrice qui est de « retirer la carte du portefeuille ».

Il est bien évident que le modèle de tâches enrichit d'erreurs humaines (voir Figure 44) est presque illisible, pour cela il est possible d'utiliser le filtre vu précédemment, et afficher seulement les éléments souhaité, par exemple ici, afin d'effectuer une analyse des tâches, les objets ne nous sont d'aucune utilité, et en utilisant le système de filtrage, on peut obtenir un modèle de tâches moins chargé en éléments et plus lisible (voir Figure 45)

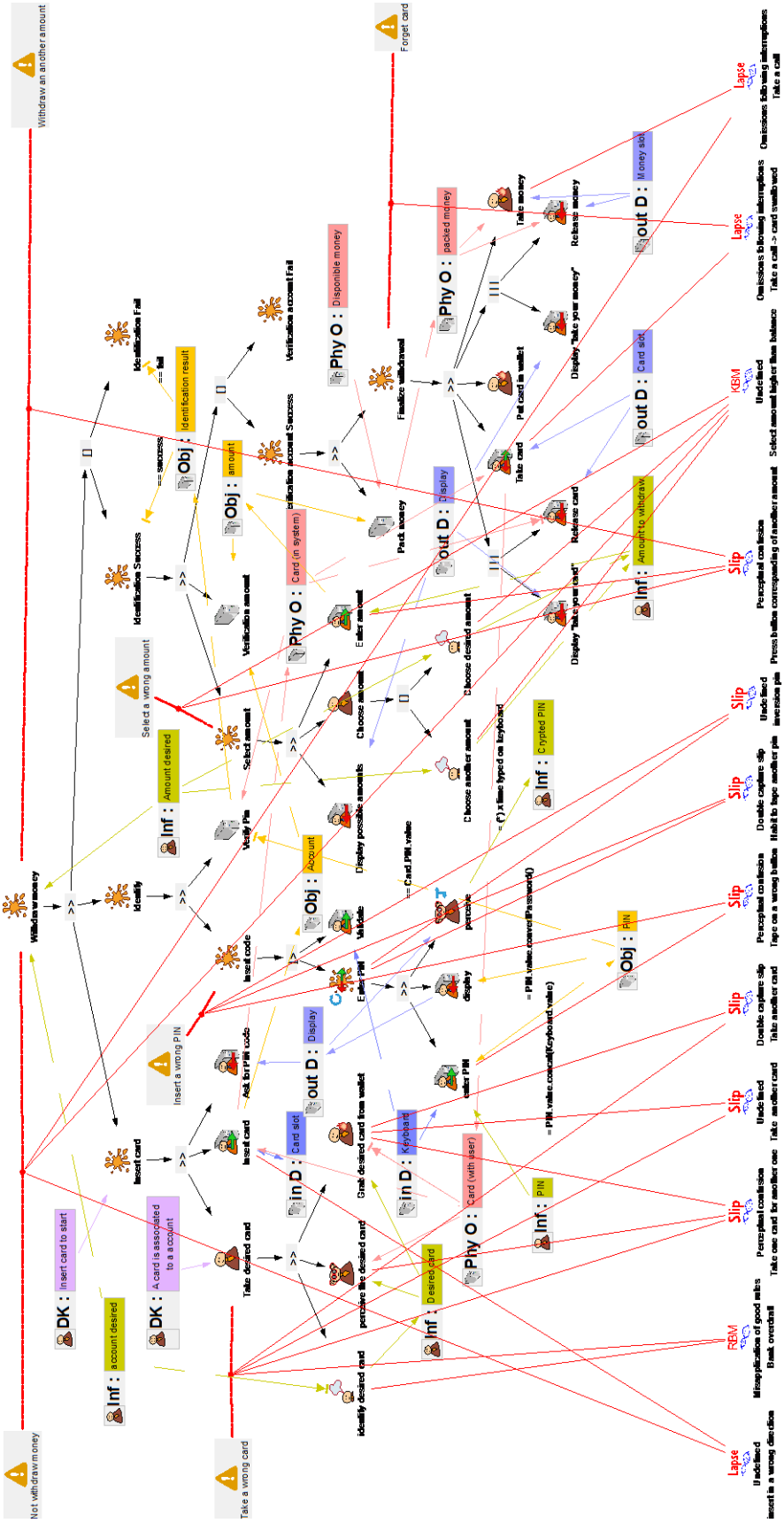


Figure 44. Modèle de tâches « Retirer de l'argent » avec la représentation des erreurs humaines

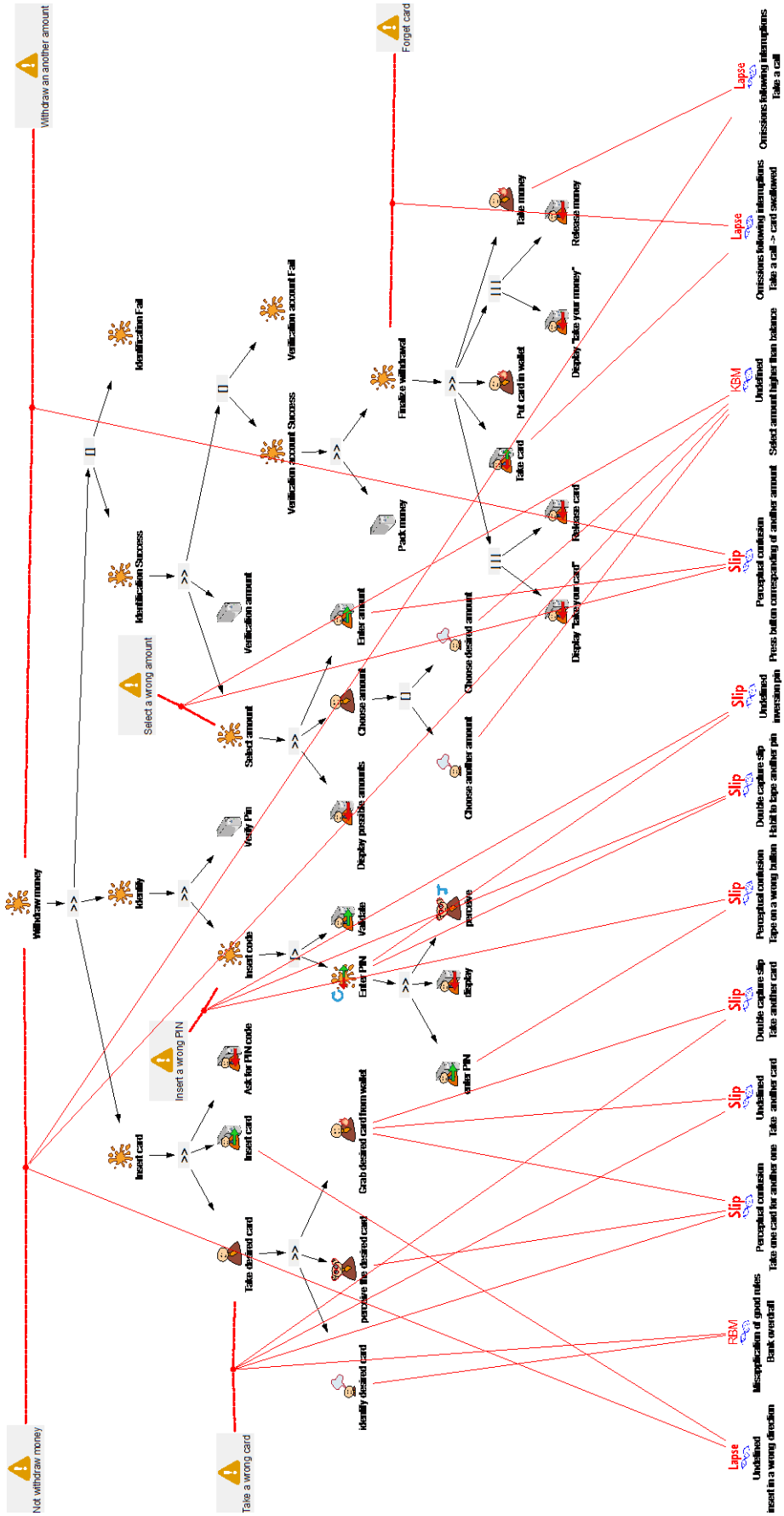


Figure 45.Modèle de tâches « Retirer de l'argent » avec la représentation des erreurs humaines et sans la description des données, objets et périphériques

5 Conclusion

Dans ce chapitre nous avons présenté les différentes extensions apportées à l'outil de modélisation et de simulation de tâche HAMSTERS, qui permettent la représentation des erreurs humaines dans des modèles de tâches. Nous avons choisi de représenter les phénotypes et génotypes séparément, et les génotypes sont sous catégorisé en quatre types, les ratés, les lapsus, les fautes basées sur les règles ainsi que les fautes basées sur les connaissances, ces catégories ont à leur tour des sous types définis par les travaux de Reason, mais nous avons aussi proposer la possibilité de rajouter d'autres sous-type d'erreur différents de ceux de Reason.

Maintenant que nous avons un outil qui permet de représenter explicitement les erreurs humaines dans les modèles de tâches, il nous manque la partie du processus qui permet d'identifier ces erreurs de façons systématiques, et qui est présenter dans le prochain chapitre.

Chapitre 5. – TASSE : Un processus d'identification systématique d'erreurs humaines (Tasks-based Assessment for Structural Specification of Errors)

Dans le précédent chapitre nous avons proposé une correspondance entre des sous-types d'erreurs humaines et des types de tâche. Cette correspondance permet d'effectuer une identification systématique des sous-types d'erreurs humaines à partir d'un modèle de tâches. Le Chapitre 3 présente plusieurs processus existants pour l'identification d'erreurs humaines. Certains de ces processus se basent sur une analyse systématique des modèles de tâches. Ces erreurs sont représentées indépendamment des modèles de tâches, aucun lien entre une tâche et ses erreurs associées ne n'est représentée, mais seulement une liste d'erreurs humaines possibles.

Il existe aussi d'autres méthodes pour effectuer une identification des erreurs humaines, mais de manière non systématique. Parmi ces méthodes, les tests utilisateurs et le retour d'expérience permettent de recenser les erreurs humaines constatées pendant l'utilisation d'un système interactif. Ces méthodes-là ne seront pas détaillées, car ce qui nous intéresse, c'est de proposer un processus qui permet d'effectuer une identification des erreurs humaines de manière systématique. Cependant, le processus proposer doit pouvoir être utilisé de manière complémentaire avec les approches non systématiques et doit ainsi permettre de rajouter de façon inopinée des erreurs humaines identifiées par un test utilisateur ou un retour d'expérience.

Dans ce chapitre nous allons présenter un processus qui permet d'identifier et de décrire de manière systématique les erreurs humaines en s'appuyant sur les modèles de tâches. Nous commencerons par représenter une vue globale du processus, ainsi que le tableau HEECA utilisé lors de l'application du processus pour recenser les erreurs possibles ainsi que leur criticité. Ensuite, nous détaillerons les sous processus qui composent le processus principal. Tout d'abord, nous détaillerons le sous processus d'identification systématique des erreurs humaines » (zone A, Figure 46), qui prend en entrée un ou des modèle(s) de tâches, et génère en sortie, un ou des modèle(s) de tâches contenant(s) des erreurs humaines. Nous détaillerons ensuite le sous processus de « prise en compte de déviation opérationnelle constatée » (zone B, Figure 46), permettant de rajouter de manière inopinée des erreurs humaines constatées lors de tests ou de retours d'expérience dans les modèles de tâches. Puis nous présenterons le sous processus de « traitement des erreurs humaines » (zone C, Figure 46).

Dans ce chapitre, nous présenterons aussi les extensions ajoutées à l'outil de modélisation de tâches HAMSTERS qui permettent de faire une identification systématique des erreurs humaines et de les représenter dans les modèles de tâches. Et nous finirons par illustrer le déroulement de l'application du processus sur l'exemple illustratif du distributeur automatique de billets.

1 Description abstraite du processus

Le processus présenté dans cette section s'appuient sur plusieurs travaux sur l'identification d'erreurs humaines. Il est principalement issu de deux travaux de recherches, l'un sur l'analyse des effets des erreurs humaines et des défaillances systèmes présenté dans (Martinie C. , et al., 2016), et l'autre sur l'identification et la représentation des erreurs humaines présenté dans (Fahssi, Martinie, & Palanque, Enhanced task modelling for systematic identification and explicit representation of human errors, 2015).

Ce processus permet d'effectuer une identification systématique des erreurs humaines, et de les représenter dans des modèles de tâches, ainsi que les erreurs constatées de manière inopinée. Il permet aussi d'effectuer une analyse de leurs effets et de leurs criticités.

1.1 Vue globale du processus TASSE

Ce processus est présenté dans la Figure 46, il intègre plusieurs sous processus. On commence par effectuer une analyse d'activité utilisateurs, puis effectuer une description de tâches de façon itératif afin de répondre au activités utilisateurs. A partir du modèle de tâches, on passe au processus d'analyse de criticité des erreurs humaines (voir Figure 46 pastille 1), ce qui nous permettra d'effectuer le traitement des erreurs (voir Figure 46 pastille 3). Le processus intègre aussi un sous processus de prise en compte de déviation opérationnelle constatée (voir Figure 46 pastille 2).

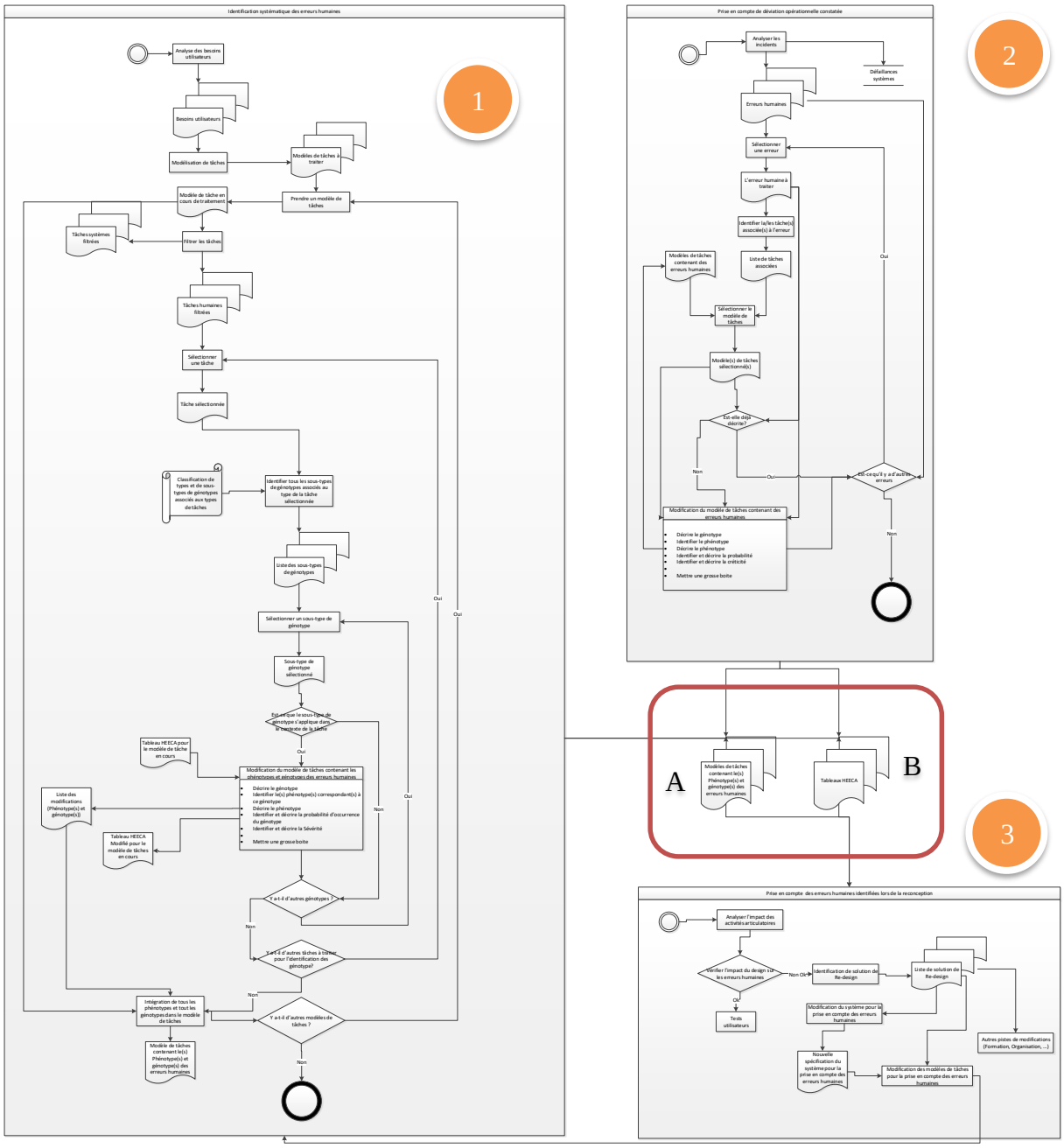


Figure 46. Processus d'identification, de représentation et de traitement des erreurs humaines

1.2 Présentation du tableau HEECA et de l'analyse de la criticité d'une erreur humaine

Comme nous l'avons vu dans la Figure 46, les deux processus 1 et 2 génère en sortie, un ou plusieurs modèle(s) de tâches et un ou plusieurs tableau(x) d'analyse de l'effet et de la criticité des erreurs humaines, appelé tableau HEECA (Humain Error Effect and Criticality Analysis).

Dans nos travaux présenter dans (Martinie C. , et al., 2016), nous proposons une technique inspirée et adaptée de l'AMDEC (U.S. Department of Defence, 1980) qui vise à analyser de

manière systématique les effets et la criticité des erreurs humaines. Le processus d'analyse HEECA (voir Figure 47) comprend plusieurs étapes :

- Définition des rôles, des nœuds de tâches et des nœuds d'action à analyser. Des descriptions complètes des tâches et actions ont été effectuées pendant la phase de modélisation des tâches.
- Identification de toutes les erreurs potentielles (en utilisant la classification d'erreur humaine de Reason (Reason, 1990)) et les écarts (en utilisant la méthode HAZOP) pour chaque élément et étudier leur effet sur l'élément analysé.
- Préparer des descriptions du génotype d'erreur qui illustrent les erreurs et les écarts potentiels et leurs conséquences.
- Évaluer chaque erreur ou écart potentiel en termes de conséquences potentielles les plus graves (sur le but correspondant et sur la mission) et attribuer une catégorie de gravité (les catégories et le numéro de gravité correspondant sont décrits dans le Tableau 12).
- Évaluer la probabilité d'occurrence de chaque erreur ou écart potentiel identifié et attribuer une catégorie de criticité (CC) en utilisant CC = NG × NP (où NP le numéro de probabilité, décrit dans le Tableau 13).

Tableau 12. Numéros de gravité appliqués aux différentes catégories de gravité et effets de défaillance.

Catégorie de gravité (CC)	**Numéros de gravité (NG)**	**Effet d'échec**
Catastrophique	4	Mort possible ou perte de système
Critique	3	Possible blessure majeure ou dommages du système
Majeur	2	Possible blessure mineure ou dégradation de l'efficacité de la mission
Négligeable	1	Nécessite la maintenance du système, mais ne présente aucun risque pour le personnel ou l'efficacité de la mission

Tableau 13. Niveaux de probabilité, limites et nombres

Niveau	**NP**
Probable	4
Occasionnel	3
Rare	2
Très rare	1

La Figure 47 représente un extrait du tableau HEECA d'analyse de l'effet et la criticité des erreurs humaines identifiées à partir du modèle de tâches « Retirer de l'argent à partir d'un distributeur automatique de billets ».

Prenons comme exemple la ligne encadrer en rouge :

- Le numéro de l'erreur identifiée est : 3
- Le rôle est : Un utilisateur du DAB
- La description du génotype : L'utilisateur perçoit mal la carte qu'il sort de son portefeuille, qui est différente de celle qu'il voulait utiliser
- Tâche parente : Prendre la carte désirée
- Action élémentaire : percevoir la carte désirée
- Déviation HAZOP : Other than
- Type de génotype : erreur basée sur les automatismes (Raté)
- Sous-type de génotype : confusion perceptuelle
- Erreur liée aux informations / connaissances procédurales ou déclaratives : procédurale
- Effet local : Une autre carte est utilisée
- Effet sur le but : tâche retardée
- Effet sur la mission : retardée
- Gravité : Majeur (2)
- Probabilité : rare (2)
- Criticité : G*P = 2*2 = 4

#	Role	Scenario	Task node	Action node	Deviation (HAZOP)	Human error reference classifications	Error related to procedural or declarative information/ knowledge	Local effect	Effect on goal	Effect on mission	Severity	Proba	Criticality
1	Controller	The operator selects the procedure to trigger, but before the start it, he gets a call that made him start the procedure too late wrt. the satellite mode.	Start procedure SWITCH ON SADA2	Mouse selection: [start procedure]	Late	Not Applicable	Not applicable	Activity delayed	Task delayed	Delayed	Critical(3)	Remote (2)	6
2	Controller	The operator selects the procedure that is next to the good one, and he doesn't notice because he doesn't look the selected procedure on the display.	Mouse selection: [select procedure]	Perceive selection	No or not	Selectivity	Procedural	Another procedure is selected	Task interrupted	Delayed	Critical(3)	Remote (2)	6
3	Controller	The operator selects the procedure that is below the good one, and he doesn't notice because the both procedures have the same name by a few letters.	Mouse selection: [select procedure]	Analyze that selection is correct	Other than	Perceptual confusion	Declarative	Another procedure is selected	Task interrupted	Delayed	Critical(3)	Remote (2)	6
4	Controller	Operator select the wrong procedure but when he intends to change, he receives a call. After the call, the operator does not change the selection and starts the procedure.	Mouse selection: [select procedure]	Analyze that cursor is on item to be selected	Other than	Omissions following interruptions	Procedural	Another procedure is selected	Task interrupted	Delayed	Critical(3)	Remote (2)	6
		"down" button without noticing, and the selected procedure is not the targeted one.	SWITCH ON SADA2	down				Another procedure is	Task	Delayed	Critical(3)		
20	Controller	Operator select the wrong procedure but when he intends to change, he receives a call. After the call, the operator does not change the selection and starts the procedure. The unintentionnally selected procedure causes the mission to fail.	Mouse selection: [select procedure]	Analyze that cursor is on item to be selected	Other than	Omissions following interruptions	Procedural	Another procedure is selected	Task interrupted	Mission fails.	Catastrophic (4)	Remote (2)	8
21	Controller	The operator selects a procedure in a list, and before starting the procedure, he touches the "down" button without noticing, and the selected procedure is not the targeted one. The unintentionnally selected procedure	Start procedure SWITCH ON SADA2	Push finger down	Other than	Slip	Procedural	Another procedure is selected	Task interrupted	Mission fails.	Catastrophic (4)	Remote (2)	8

Figure 47. Tableau HEECA d'analyse des effets, probabilité et criticité des erreurs humaines (Human Error Effect and Criticity Analysis)

Le tableau HEECA nous permet de lister et décrire les erreurs humaines identifiées de manière systématique ou inopinée, afin d'effectuer une analyse sur la criticité de l'erreur, et décidé de traiter l'erreur ou non, comme par exemple donner un seuil de criticité.

Ce tableau est implémenté dans l'outil de modélisation de tâches HAMSTERS, ce qui permet de le préremplir de manière systématique. C'est ce que nous allons voir dans la sous-section suivante, ce sont les deux processus qui nous permettent de remplir le tableau de façon systématique ou même inopiné. Puis à partir de ce tableau on passe sur un autre processus qui permet d'effectuer le traitement des erreurs humaines.

1.3 Vue détaillée du TASSE

Comme expliqué dans la section précédente, le processus TASSE intègre 3 sous processus : l'identification systématique des erreurs humaines, la prise en compte de déviation opérationnelle constatée et la partie de traitement des erreurs humaines.

1.3.1 Identification systématique des erreurs humaines

Le sous processus d'identification systématique des erreurs humaines (étiquette 1 de la Figure 46) est inspiré de la technique HET (Stanton, et al., 2006). Contrairement à HET qui utilise des modèles de tâches produits avec la méthode HTA (Annett, Hierarchical task analysis, 2003), nous utilisons la notation de modélisation de tâches HAMSTERS, pour des raisons d'expressivité de notation expliquées dans le chapitre 1. Le choix de s'appuyer sur le processus HET est aussi justifié dans le chapitre 3 par le fait que ce processus est plus détaillé que la plupart de processus d'identification des erreurs humaines.

Nous avons présenté dans la section 1.6**Erreur ! Source du renvoi introuvable.** du Chapitre 3 un processus d'identification d'erreurs humaines (HET) qui permet de détecter de façon systématique les erreurs humaines, mais il utilise une description de tâches peu expressive, et il n'offre pas d'outil d'aide à l'identification des erreurs et à l'association de ces erreurs à des types de tâches utilisateur.

Nous avons étendu ce processus afin qu'il puisse permettre d'identifier les génotypes d'erreurs et d'associer et de représenter explicitement chaque génotype d'erreur à la tâches associée.

La Figure 48 présente une version étendue du processus HET. Ce processus fournit un support pour identifier et représenter les génotypes et phénotypes des erreurs humaines possibles, en intégrant des descriptions d'erreur dans les modèles de tâches. Le processus étendu commence par une analyse et modélisation des tâches (qui correspond à une version étendue de la description des tâches avec HET). Dans notre cas, les modèles de tâches produits sont raffinés, afin de représenter les tâches perceptives, cognitives et motrices, ainsi que les informations et les connaissances nécessaires pour effectuer les tâches. Ces modèles tirent pleinement parti de la puissance expressive de la notation HAMSTERS (présentée dans le Chapitre 1.3.2). Toutes les modifications apportées par rapport au processus HET ont été explicitées en utilisant différentes nuances de gris.

Le processus se déroule de la façon suivante (les numéros d'étapes correspondent aux étiquettes placées sur chaque rectangle dans la Figure 48) :

1. Nous commençons par effectuer une analyse des besoins utilisateurs
2. Nous effectuons une modélisation de tâches, pour obtenir un ensemble de modèles de tâches qui permettent de décrire toute les tâches utilisateurs du système à analyser.
3. A partir de ces modèles de tâches, nous allons effectuer une identification systématique d'erreurs humaines pour chaque modèle.
4. Nous prenons un modèle à traiter
5. Nous filtrons d'un côté les tâches humaines et de l'autre les tâches systèmes qui ne seront pas traiter dans ce processus.
6. Nous prenons une tâche humaine
7. Nous effectuons une identification des sous-types de génotype associés au type de la tâche sélectionnée, et cela à l'aide de la classification de types et sous-types de génotypes associés aux types de tâches (voir Tableau 14, présenté dans le paragraphe suivant cette description du sous processus).
8. Après l'étape 7, nous obtenons une liste de sous-types de génotypes.
9. Nous prenons un sous-type de génotype, et nous analysons le contexte de la tâche pour décider si le sous-type de génotype peut s'appliquer à la tâche.
10. Si non, nous allons directement à l'étape 12
11. Si oui :
    - Nous décrivons le génotype dans le tableau HEECA (présenté dans la section 1.2) du modèle de tâche en cours de traitement
    - Nous identifions le(s) phénotype(s) correspondant(s) à ce génotype
    - Nous décrivons le génotype dans le tableau HEECA (présenté dans la section 1.2)
    - Nous identifions et décrivons la probabilité d'occurrence du génotype ainsi que la sévérité.
12. Nous vérifions s'il y a d'autre sous-types de génotype à traiter
13. Si oui, nous reprenons à l'étape 9
14. Nous vérifions s'il y a d'autre tâches à traiter pour l'identification des génotypes
15. Si oui, nous reprenons à l'étape 6
16. A partir du tableau HEECA (présenté dans la section 1.2), nous intégrons tous les génotypes et phénotypes dans le modèle de tâches pour obtenir un nouveau modèle de tâches contenant les erreurs humaines
17. Nous vérifions s'il y d'autre modèle de tâches à traiter
18. Si oui, nous reprenons à l'étape 4
19. Si non, le processus est terminé.

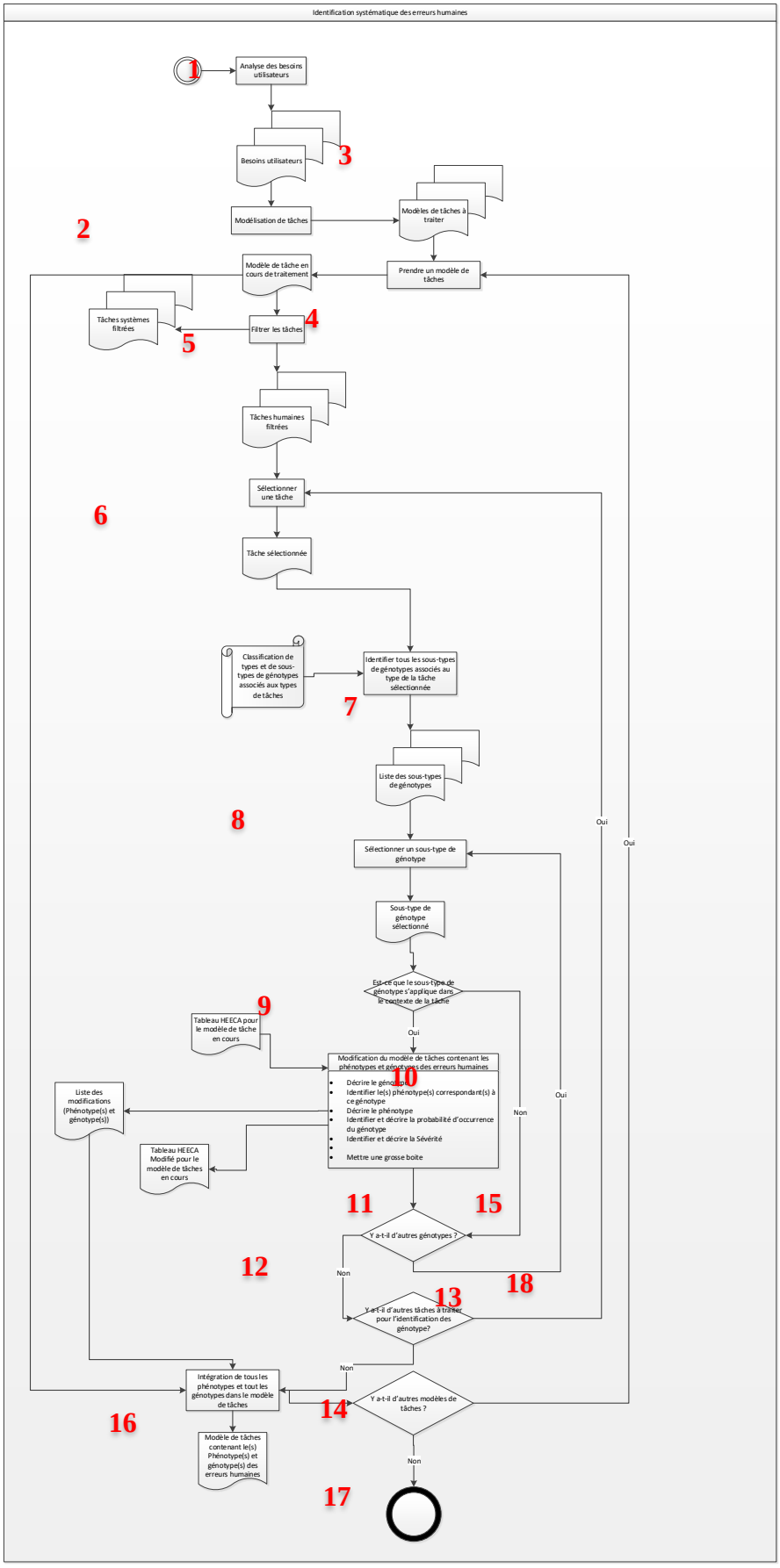


Figure 48. Processus d'identification systématique des erreurs humaines

Le Tableau 14, représente les correspondances entre le type de tâche utilisateur ou le type de données manipulées (voir Chapitre 1.3.2), et les types de génotypes associés (voir Chapitre 4.1).

La colonne de gauche représente les types d'éléments de la notation HAMSTERS, dans lequel on retrouve les types de tâches utilisateurs : tâche perceptive, tâche interactive d'entrée, tâche motrice et tâches cognitives. On y retrouve aussi les données manipulées par l'utilisateur, ce qui veut dire que si pour l'exécution d'une tâche, il faut manipuler une information ou une connaissance, les types de génotypes associés aux données, seront aussi associés à la tâche. La colonne de droite représente les sous-types de génotypes de la classification de Reason (Reason, 1990). On y retrouve tous les sous types de génotype possibles pour un type de génotype.

Par exemple, la première ligne du tableau indique que, pour une tâche perceptive, le seul génotype associé est « l'erreur basée sur les automatismes », et que les sous types de génotype possibles sont : la confusion perceptuelle et l'erreur d'interférence.

Tableau 14. Tableau de correspondance entre sous-type de génotype et type de tâche

<table>
<tr><th>Éléments de la notation HAMSTERS</th><th colspan="2">Sous type de génotypes</th></tr>
<tr><td>Tâche perceptive
  </td><td colspan="2">Perceptual confusion (Skill Based Error)
Interference error (Skill Based Error)</td></tr>
<tr><td>Tâche interactive d'entrée
 
Tâche motrice
  </td><td colspan="2">Interference error (Skill Based Error)
Double capture slip (Skill Based Error)
Omissions following interruptions (Skill Based Error)</td></tr>
<tr><td rowspan="3">Tâche cognitive
  </td><td>Skill based errors</td><td>– Double capture slip
– Omissions following interruptions
– Reduced intentionality
– Interference error
– Over-attention errors</td></tr>
<tr><td>Rule based mistakes</td><td>Misapplication of good rules
– First exceptions
– Countersigns and non-signs
– Informational overload
– Rule strength
– General rules
– Redundancy
– Rigidity
Application of bad rules
– Encoding deficiencies
– Action deficiencies</td></tr>
<tr><td>Knowledge based mistakes</td><td>– Selectivity
– Workspace limitations
– Out of sight out of mind
– Confirmation bias
– Overconfidence
– Biased reviewing
– Illusory correlation
– Halo effects
– Problems with causality
– Problems with complexity</td></tr>
<tr><td>Information
Inf : Information</td><td colspan="2">Double capture slip, Omissions following interruptions, Interference error, all of the Rule Based Mistakes and Knowledge Based Mistakes</td></tr>
<tr><td>Connaissance déclarative
DK : Declarative</td><td colspan="2">All of the Knowledge Based Mistakes</td></tr>
</table>

1.3.2 Prise en compte de déviation opérationnelle constatée

Dans la vie réelle, un incident ou accident dû à une erreur humaine peut arriver, sans qu'on ait pu prévoir cette erreur humaine lors de la conception. L'analyse de ces incidents et accidents a pour but de découvrir l'origine de l'erreur. Les résultats de ce type d'analyse peuvent être

intégré aux processus de l'analyse systématique des erreurs humaines.

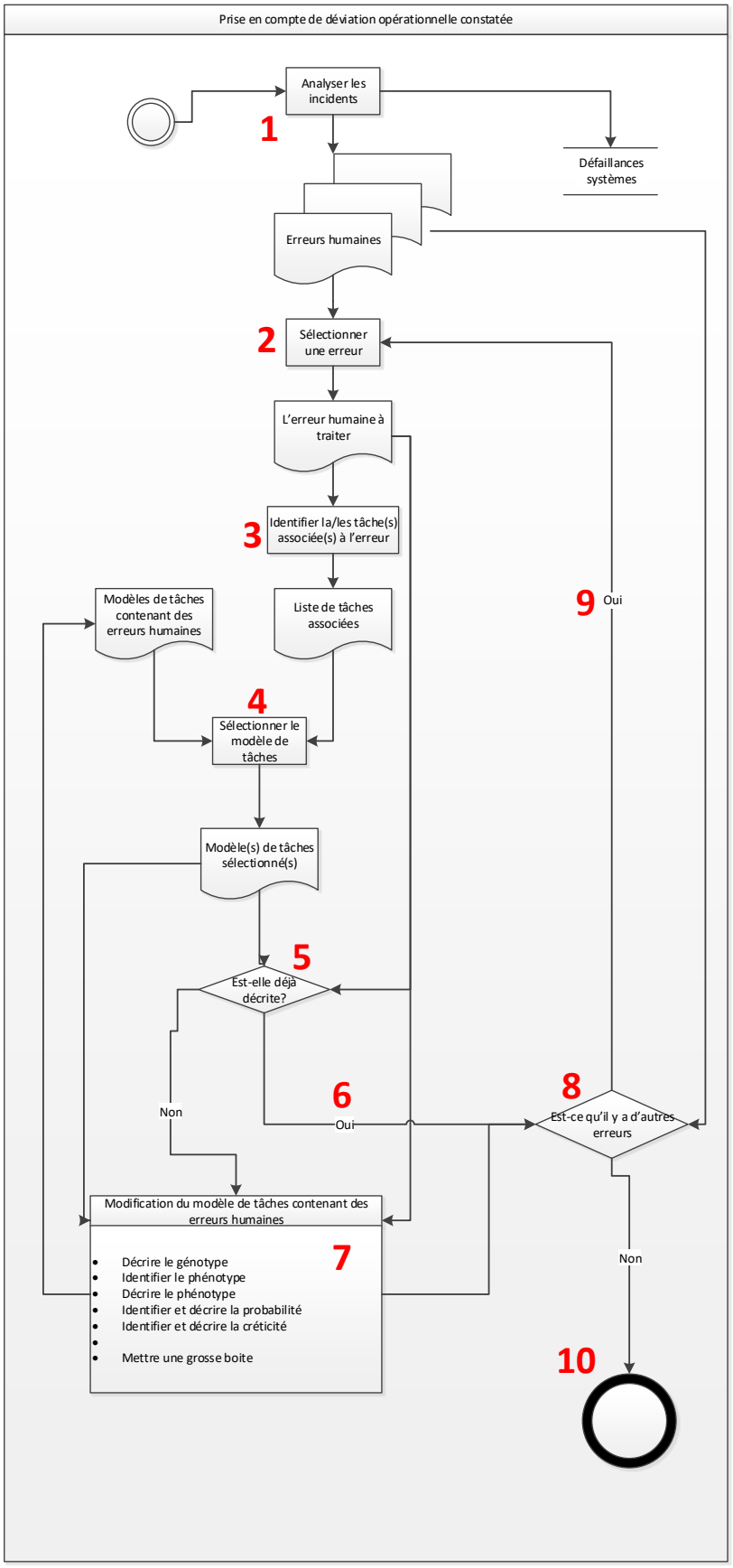


Figure 49 représente le sous processus permettant d'intégrer des erreurs humaines identifiées de façons inopinée (les numéros d'étapes correspondent aux étiquettes placées sur chaque rectangle dans la Figure 49) :

1. Le processus commence par une analyse des incidents et accidents, afin de définir les causes, et traiter seulement les causes humaines
2. Nous prenons une erreur humaine à traiter
3. Nous identifions les tâches associées à cette erreur
4. Nous sélectionnons les modèles de tâches contenant ces tâches là
5. Nous vérifions si l'erreur est déjà décrite
6. Si oui, nous passons à l'étape 8
7. Si non,
    a. Nous décrivons le génotype dans le tableau HEECA (présenté dans la section 1.2) du modèle de tâche en cours de traitement
    b. Nous identifions le(s) phénotype(s) correspondant(s) à ce génotype
    c. Nous décrivons le génotype dans le tableau HEECA (présenté dans la section 1.2)
    d. Nous identifions et décrivons la probabilité d'occurrence du génotype ainsi que la sévérité.
8. Nous vérifions s'il y a d'autres erreurs
9. Si oui, nous reprenons à l'étape 2
10. Si non, le processus est terminé.

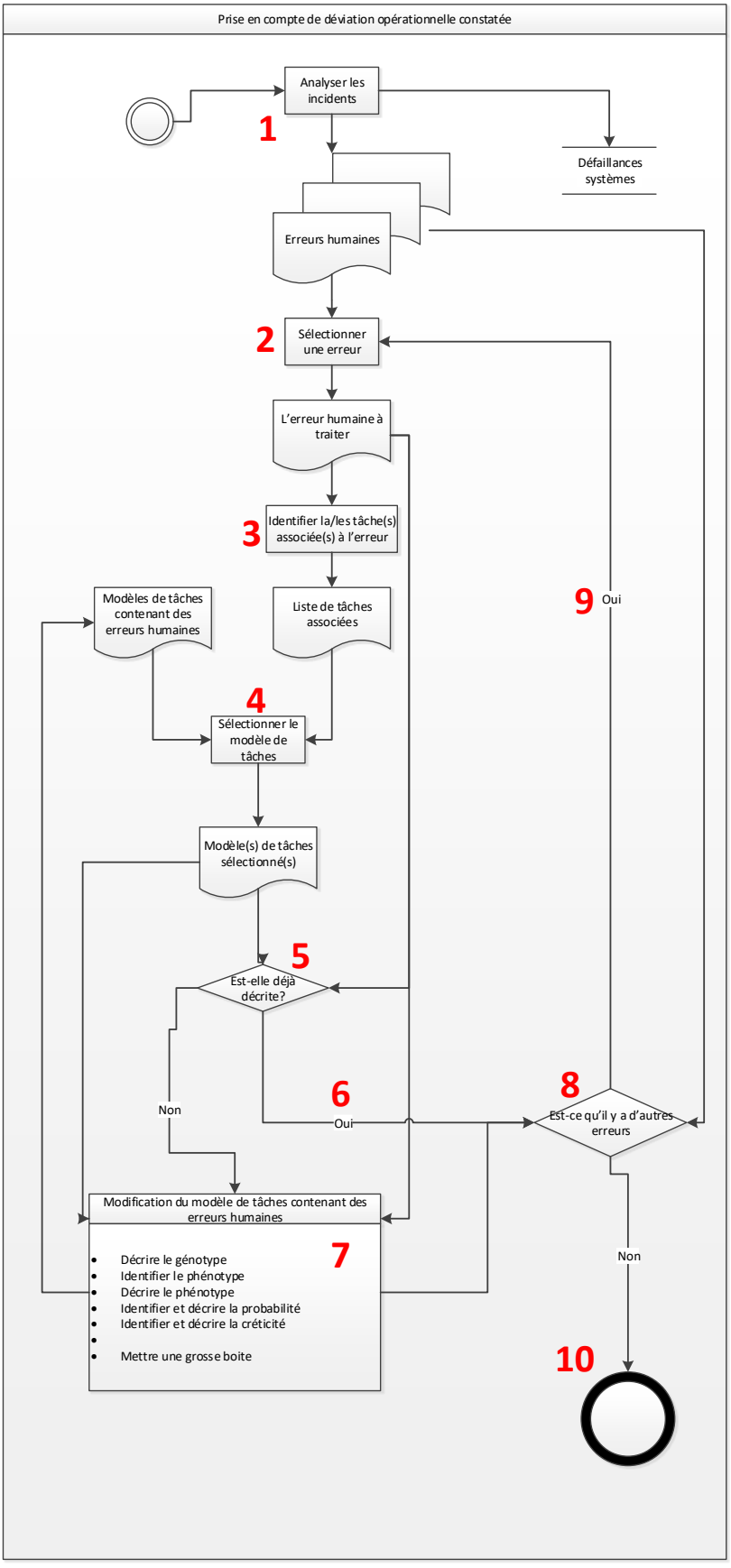


Figure 49. Processus de prise en compte de déviation opérationnelle constatée

1.3.3 Traitement des erreurs humaines

Après avoir appliqué les processus précédents, nous obtenons en sortie de ces processus des modèles de tâches contenant les phénotypes et génotypes des erreurs humaines, ainsi que le tableau HEECA (présenté dans la prochaine section).

La Figure 50 représente le processus qui permet de décrire la démarche que nous proposons afin d'éviter les erreurs humaines identifiées. Ce processus prend en entrée les modèles de tâches contenant les erreurs humaines et le tableau HEECA.

1. Nous commençons par effectuer une analyse de l'impact des activités articulatoires
2. On vérifie l'impact du design sur les erreurs humaines
3. Si c'est bon, nous effectuons des tests utilisateurs
4. Sinon :
    a. Nous identifions des solutions de re-design
    b. A partir de la liste de modifications, nous effectuons soit des modifications du système pour la prise en compte des erreurs humaines, soit la modification des modèles de tâches, ou autres pistes de modifications (tel que les formations, d'autres organisations, …). Sachant qu'une modification du système, implique une modification des modèles de tâches, et ce qui implique aussi une autre identification systématique des erreurs humaines.

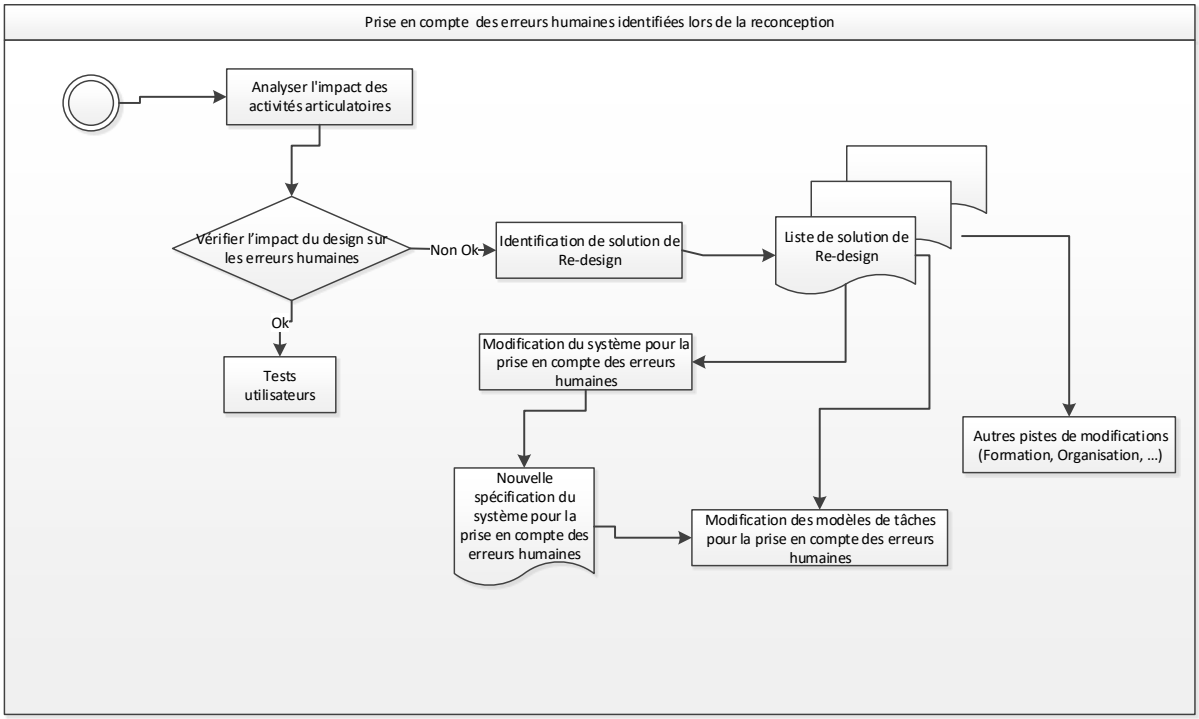


Figure 50. Processus de traitement des erreurs humaines

2 Outillage du processus dans Hamsters

Dans le Chapitre 1.3.2, on a présenté HAMSTERS, qui est un outil de modélisation et de simulation de modèles de tâches. Dans le Chapitre 4.3 nous avons présenté les extensions rajoutées à l'outil HAMSTERS pour la représentation des erreurs humaines (Phénotype et Génotype). Dans cette section nous allons seulement présenter les extensions apportées pour le déroulement du sous processus d'identification systématique des erreurs humaines.

La Figure 51 représente l'outil HAMSTERS. La partie entourée de vert représente le panel d'aide à l'identification des erreurs humaines basé sur le processus TASSE. Ce panel contient un tableau, qui représente le tableau HEECA pré-remplie, un champ pour filtrer le tableau et deux boutons, un pour rafraîchir le tableau « Refresh Table », et un pour actualiser le modèle de tâches avec les erreurs identifiées « Generate in model ».

- Le tableau permet d'afficher la liste des tâches feuille du modèle de tâches, mais pas toute les tâches, seulement les tâches du même type du tableau de correspondances (voir Figure 52). Les tâches sont listées par ordre d'exécution gérer par les opérateurs temporels. On y retrouve 7 colonnes, la première colonne représente la tâches feuille précédé par l'icône représentante du type de la tâche, la deuxième colonne décrit le type de la tâche, la troisième le type de génotype, la quatrième le sous-type de génotype, la cinquième la description du génotype, la sixième le phénotype de l'erreur, et enfin la dernière représente une case à cocher pour dire si l'erreur est crédible ou non.
- Le champ de filtrage, à gauche en dessous du tableau, permet d'afficher dans le tableau, seulement les lignes qui contiennent le mot clé entré dans le champ. Par exemple si en tape « Slip » dans le champ de filtrage, le tableau va afficher seulement les lignes de tâches feuille qui ont comme génotype potentiel le type « Slip ».

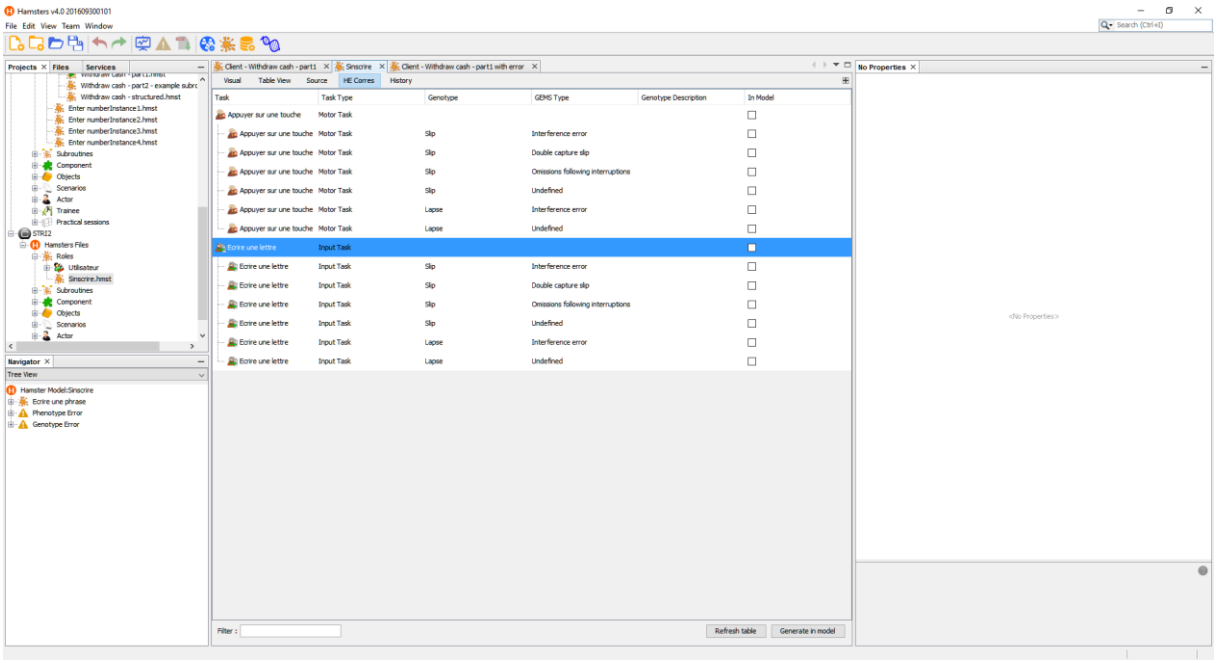


Figure 51. Panel d'aide à l'identification systématique des génotypes dans l'outil HAMSTERS

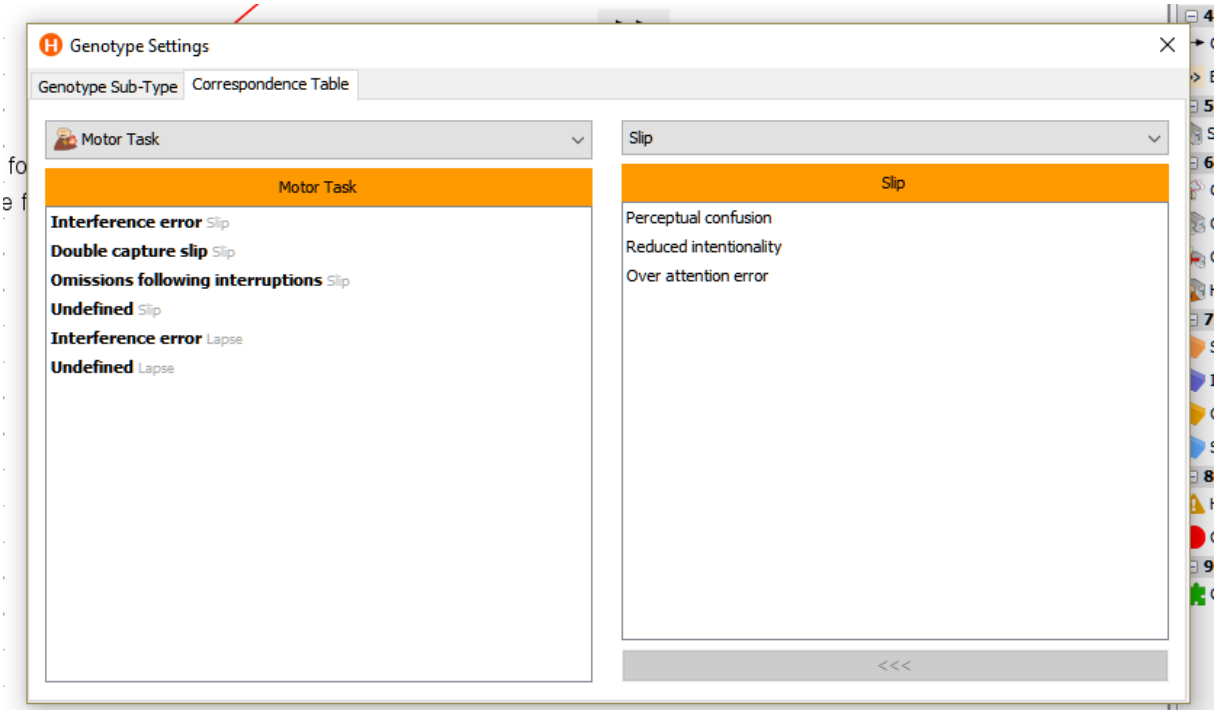


Figure 52. Panel de gestion du tableau de correspondance en types de tâches et types de génotypes

3 Application du processus sur le DAB

Dans cette section nous allons illustrer les processus TASSE à l'aide d'une étude de cas du distributeur automatique de billets et de l'outil d'analyse et de modélisation de tâches HAMSTES. Ici nous allons seulement présenter le déroulement du sous processus TASSE qui est l'identification systématique des erreurs humaines.

La tâche à analyser sera le retrait d'argent à partir d'un distributeur automatique de billets.

3.1 Modèle de tâche de l'utilisation du DAB

Afin d'illustrer le déroulement du processus d'identification systématique des erreurs humaines à partir de modèles de tâches, nous prenons comme exemple de tâche « le retrait d'argent à partir d'un DAB ».

L'exécution de cette tâche est effectuée par un client qui souhaite retirer une somme d'argent désirée à partir d'un compte bancaire désiré et à l'aide d'une carte bancaire et d'un distributeur automatique de billets.

La Figure 53 représente la description de tâches que le client doit suivre afin d'atteindre son but. Ce modèle de tâches ne permet pas au client le traitement des erreurs, c'est un modèle de tâches du plus court chemin afin d'atteindre un but.

Le détail et la description du modèle de tâches a été vue dans le Chapitre 4.

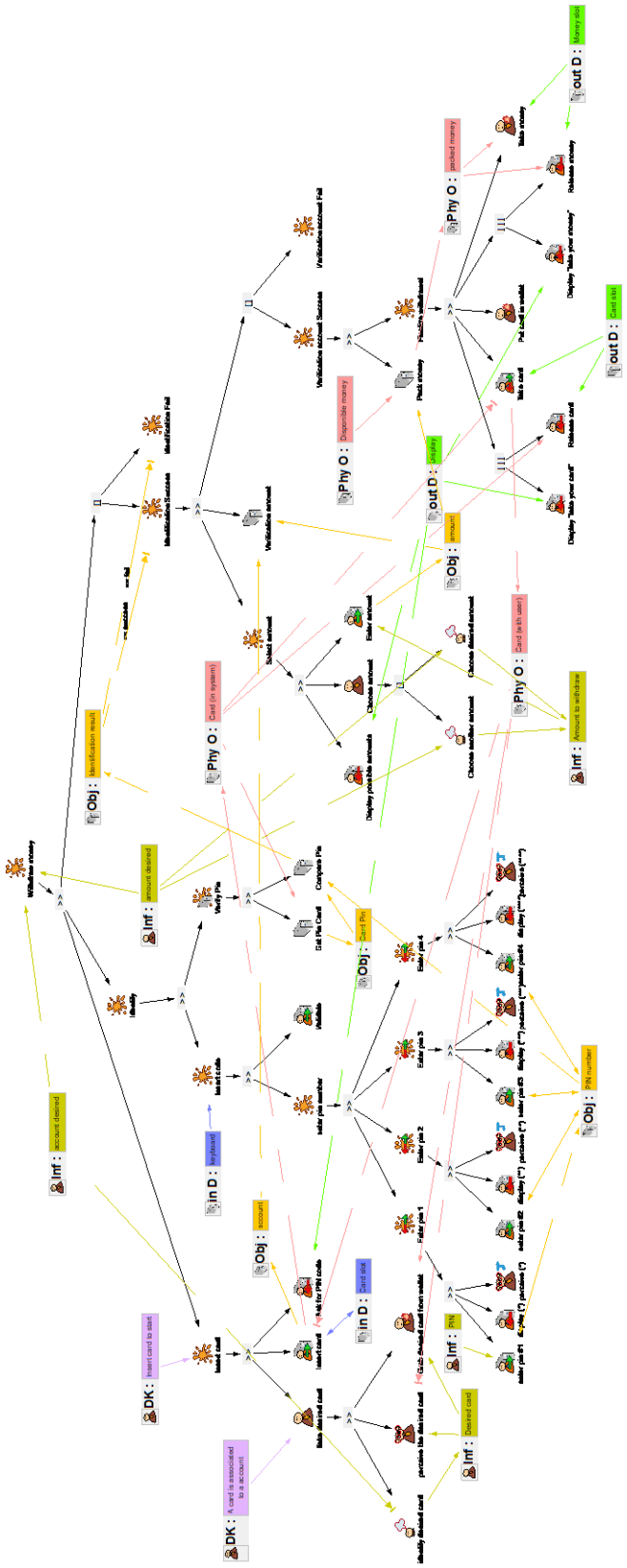


Figure 53. Modèle de tâches de la tâche retirer de l'argent sans les tâches de traitement des erreurs

3.2 Identification systématique des erreurs et analyse de leurs criticités

Pour l'identification systématique des erreurs humaines et l'analyse de leurs criticité, nous avons besoin du sous-processus TASSE (Figure 46 voir pastille 1), du modèle de tâche (voir Figure 53), et de l'outil de modélisation de tâches et d'aide à l'identification d'erreurs humaine HAMSTERS (voir Figure 51).

Une étape du sous-processus est d'effectuer une modélisation de tâches, ici nous avons déjà notre modèle de tâches, nous allons seulement effectuer une identification systématique des erreurs humaines, et leurs représentations dans le modèles de tâches, aussi la production du tableau d'analyse de l'effet et de criticité de l'erreur humaine HEECA. On commence à dérouler le processus à partir de l'étape 4 (voir Figure 48).

L'étape 5 consiste à filtrer d'un côté les tâches humaines et de l'autre les tâches systèmes qui ne seront pas traiter dans ce processus. Nous obtenons une liste de tâches humaines (voir Figure 54)

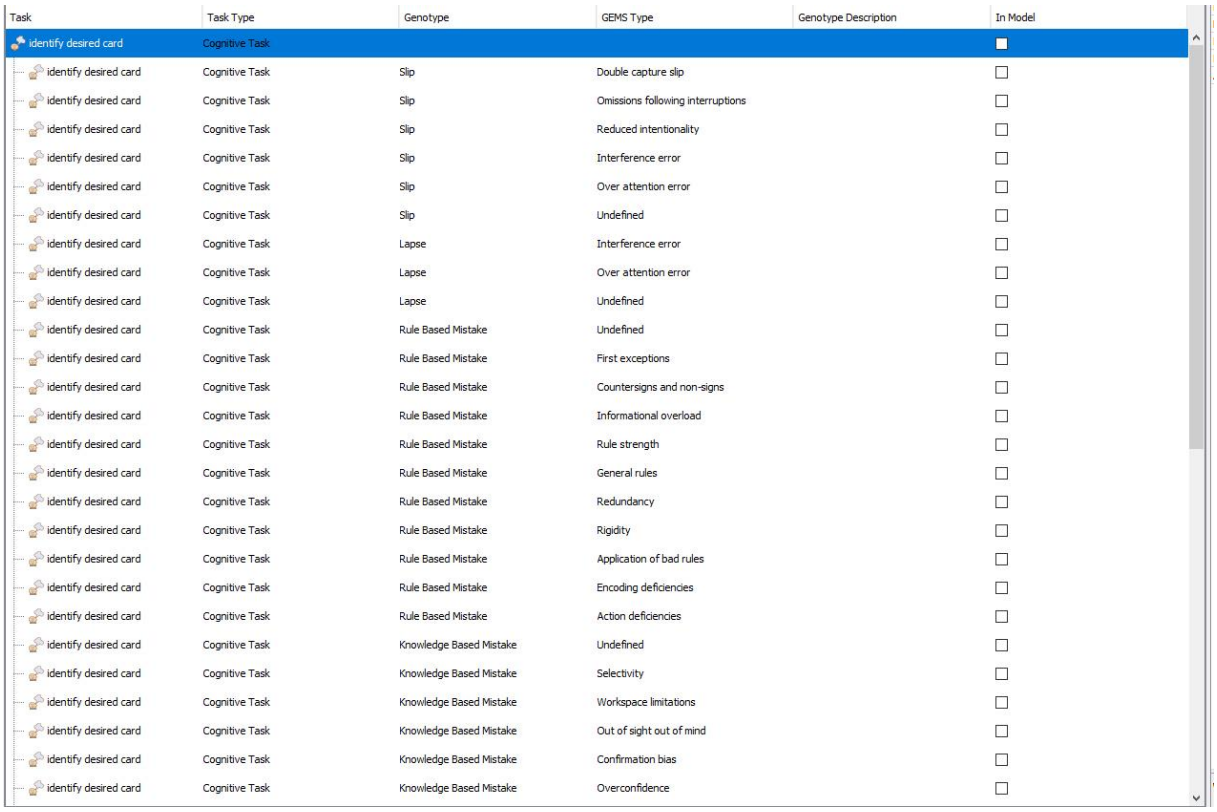
Task	Task Type
identify desired card	Cognitive Task
perceive the desired card	Perceptive Task
Grab desired card from wallet	Motor Task
Insert card	Input Task
Choose another amount	Cognitive Task
Choose desired amount	Cognitive Task
Enter amount	Input Task
Take money	Motor Task
Put card in wallet	Motor Task
enter pin #1	Input Task
perceive (*)	Perceptive Task
enter pin #4	Input Task
perceive (****)	Perceptive Task
enter pin #3	Input Task
perceive (***)	Perceptive Task
enter pin #2	Input Task
perceive (**)	Perceptive Task
Validate	Input Task

Figure 54. Liste des tâches humaines filtrées à partir du modèle de tâches retirer de l'argent à partir du DAB

Etape 6 : prendre une tâche humaine, ici ça sera « identify desired card »

Etape 7 : Nous effectuons une identification des sous-types de génotype associés au type de la tâche (qui est ici « cogitive task »), et cela à l'aide de la classification de types et sous-types de génotypes associés aux types de tâches. Ce tableau est consultable et modifiable dans l'outil HAMSTERS (voir Chapitre 4 section 3.4).

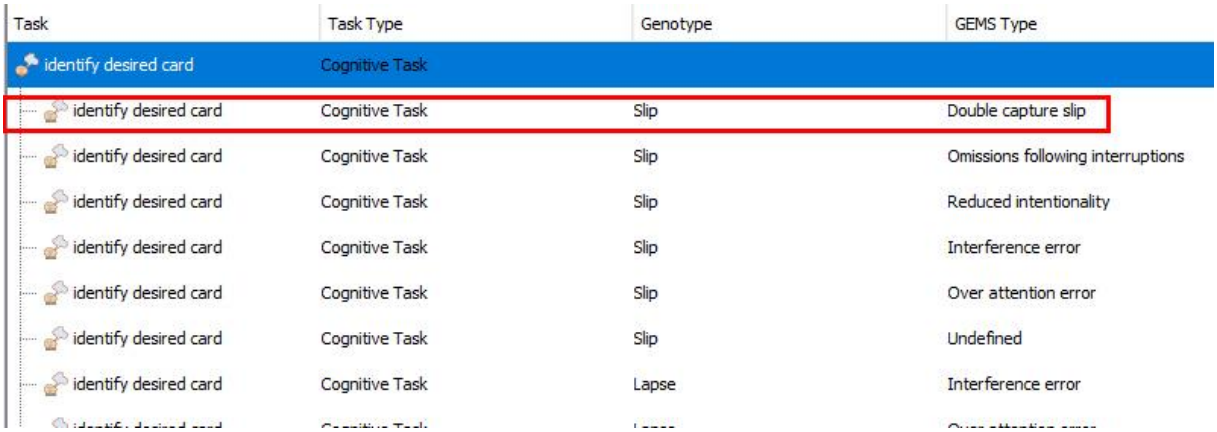
Etape 8 : En cliquant tout simplement sur la tâches « identify desired card », on fait dérouler la liste des sous-types de génotype correspondante à la tâche sélectionnée (voir Figure 55).



Task	Task Type	Genotype	GEMS Type	Genotype Description	In Model
identify desired card	Cognitive Task				☐
identify desired card	Cognitive Task	Slip	Double capture slip		☐
identify desired card	Cognitive Task	Slip	Omissions following interruptions		☐
identify desired card	Cognitive Task	Slip	Reduced intentionality		☐
identify desired card	Cognitive Task	Slip	Interference error		☐
identify desired card	Cognitive Task	Slip	Over attention error		☐
identify desired card	Cognitive Task	Slip	Undefined		☐
identify desired card	Cognitive Task	Lapse	Interference error		☐
identify desired card	Cognitive Task	Lapse	Over attention error		☐
identify desired card	Cognitive Task	Lapse	Undefined		☐
identify desired card	Cognitive Task	Rule Based Mistake	Undefined		☐
identify desired card	Cognitive Task	Rule Based Mistake	First exceptions		☐
identify desired card	Cognitive Task	Rule Based Mistake	Countersigns and non-signs		☐
identify desired card	Cognitive Task	Rule Based Mistake	Informational overload		☐
identify desired card	Cognitive Task	Rule Based Mistake	Rule strength		☐
identify desired card	Cognitive Task	Rule Based Mistake	General rules		☐
identify desired card	Cognitive Task	Rule Based Mistake	Redundancy		☐
identify desired card	Cognitive Task	Rule Based Mistake	Rigidity		☐
identify desired card	Cognitive Task	Rule Based Mistake	Application of bad rules		☐
identify desired card	Cognitive Task	Rule Based Mistake	Encoding deficiencies		☐
identify desired card	Cognitive Task	Rule Based Mistake	Action deficiencies		☐
identify desired card	Cognitive Task	Knowledge Based Mistake	Undefined		☐
identify desired card	Cognitive Task	Knowledge Based Mistake	Selectivity		☐
identify desired card	Cognitive Task	Knowledge Based Mistake	Workspace limitations		☐
identify desired card	Cognitive Task	Knowledge Based Mistake	Out of sight out of mind		☐
identify desired card	Cognitive Task	Knowledge Based Mistake	Confirmation bias		☐
identify desired card	Cognitive Task	Knowledge Based Mistake	Overconfidence		☐

Figure 55. Liste des sous-type de génotype associé à une tâche de type cognitive

Etape 9 : nous prenons un sous-type de génotype, ici le premier non traité est le génotype Slip de sous-type double capture slip (voir Figure 56), et nous analysons le contexte de la tâche pour décider si le sous-type de génotype peut s'appliquer à la tâche.



Task	Task Type	Genotype	GEMS Type
identify desired card	Cognitive Task		
identify desired card	Cognitive Task	Slip	Double capture slip
identify desired card	Cognitive Task	Slip	Omissions following interruptions
identify desired card	Cognitive Task	Slip	Reduced intentionality
identify desired card	Cognitive Task	Slip	Interference error
identify desired card	Cognitive Task	Slip	Over attention error
identify desired card	Cognitive Task	Slip	Undefined
identify desired card	Cognitive Task	Lapse	Interference error

Figure 56. Sélection d'un sous-type de génotype « Double capture slip » d'une tâche cognitive « identify desired card »

Etape 10 : Le sous-type « double capture slip » ne s'applique pas au contexte. On va à l'étape 12.

Etape 12 : on vérifie si y a d'autre sous-type de génotype à analyser

Etape 13 : on remonte à l'étape 9

Etape 9 : nous prenons un sous-type de génotype, ici le premier non traité est le génotype Slip de sous-type Omissions following interruption (voir Figure 56), et nous analysons le contexte de la tâche pour décider si le sous-type de génotype peut s'appliquer à la tâche.

Task	Task Type	Genotype	GEMS Type
identify desired card	Cognitive Task		
identify desired card	Cognitive Task	Slip	Double capture slip
identify desired card	Cognitive Task	Slip	Omissions following interruptions
identify desired card	Cognitive Task	Slip	Reduced intentionality
identify desired card	Cognitive Task	Slip	Interference error
identify desired card	Cognitive Task	Slip	Over attention error
identify desired card	Cognitive Task	Slip	Undefined

Figure 57. Sélection d'un sous-type de génotype « Omissions following interruptions » d'une tâche cognitive « identify desired card »

Etape 10 : Le sous-type « Omissions following interruption» ne s'applique pas au contexte. On va à l'étape 12.

Etape 12 : on vérifie si y a d'autre sous-type de génotype à analyser

Etape 13 : on remonte à l'étape 9

Etape 9 : (on avance jusqu'à ce qu'on tombe sur une erreur qui s'applique) Nous prenons un sous-type de génotype, ici le premier non traité est le génotype Rule based mistake de sous-type Misapplication of good rules (voir Figure 56), et nous analysons le contexte de la tâche pour décider si le sous-type de génotype peut s'appliquer à la tâche.

identify desired card	Cognitive Task	Knowledge Based Mistake	Biased reviewing		☐
identify desired card	Cognitive Task	Knowledge Based Mistake	Illusory correlation		☐
identify desired card	Cognitive Task	Knowledge Based Mistake	Halo effects		☐
identify desired card	Cognitive Task	Knowledge Based Mistake	Problems with causality		☐
identify desired card	Cognitive Task	Knowledge Based Mistake	Problems with complexity		☐
identify desired card	Cognitive Task	Rule Based Mistake	Misapplication of good rules	Bank overdraft	☑

Figure 58. Sélection d'un sous-type de génotype « Misapplication of good rules » d'une tâche cognitive « identify desired card »

Etape 11 : Le sous-type « Misapplication of good rules» s'applique au contexte, on coche la case dans la colonne « erreur potentiel », et en remplis les champs vide de la ligne du sous-génotype sélectionné :

On décrit le sous-type de génotype : découvert bancaire

On décrit l'effet local, l'effet sur le but ainsi que sur la mission : Echec

On identifie la gravité et la probabilité de l'erreur : G = 2, P = 3

On obtient automatique la criticité : 6

On identifie le phénotype correspondant à ce génotype : ici le phénotype est « échec de retrait d'argent »

Etape 12 : on vérifie si y a d'autre sous-type de génotype à analyser, ici y en a plus, on passe à l'étape 14

Etape 14 : On vérifie si y a d'autre tâches à traiter, ici oui (voir Figure 59)

Etape 15 : On remonte à l'étape 6.

Visual | Table View | Source | HE Corres | History

Task	Task Type	Genotype	GEMS Type
identify desired card	Cognitive Task	Knowledge Based Mistake	Workspace limitations
identify desired card	Cognitive Task	Knowledge Based Mistake	Out of sight out of mind
identify desired card	Cognitive Task	Knowledge Based Mistake	Confirmation bias
identify desired card	Cognitive Task	Knowledge Based Mistake	Overconfidence
identify desired card	Cognitive Task	Knowledge Based Mistake	Biased reviewing
identify desired card	Cognitive Task	Knowledge Based Mistake	Illusory correlation
identify desired card	Cognitive Task	Knowledge Based Mistake	Halo effects
identify desired card	Cognitive Task	Knowledge Based Mistake	Problems with causality
identify desired card	Cognitive Task	Knowledge Based Mistake	Problems with complexity
identify desired card	Cognitive Task	Rule Based Mistake	Misapplication of good rules
perceive the desired card	Perceptive Task		
Grab desired card from wallet	Motor Task		
Insert card	Input Task		
Choose another amount	Cognitive Task		

Figure 59. Liste partielle des tâches humaines filtrées à partir du modèle de tâches retirer de l'argent à partir du DAB

Etape 16 : Après avoir analyser toutes les tâches humaines, nous prenons le tableau HEECA de l'outil HMASTERS, et on clique sur « generate in model » afin de générer un modèle de tâche avec les erreurs humaines (voir Figure 60) et à partir du tableau HEECA qu'on a remplie de façon systématique.

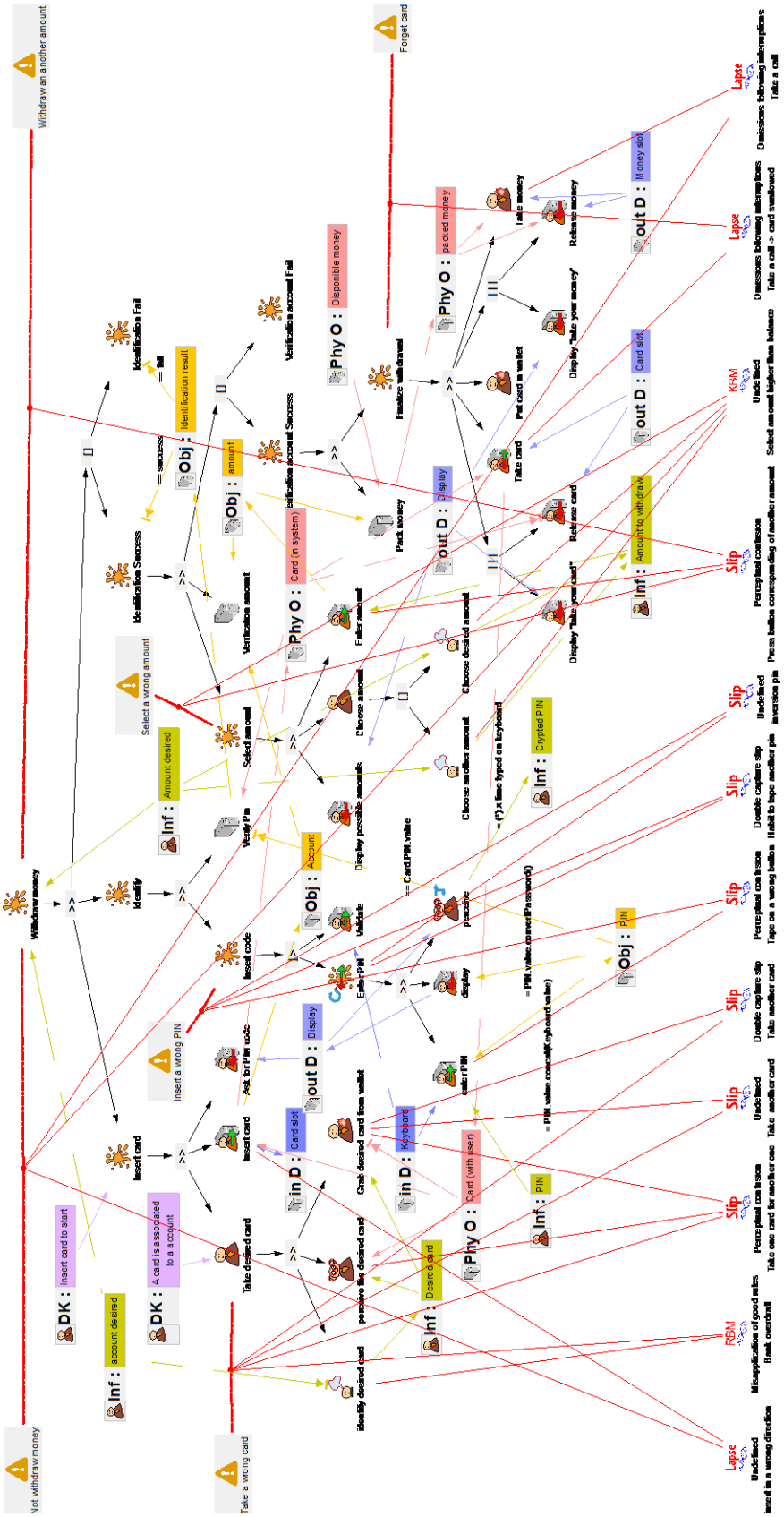


Figure 60. Modèle de tâche "retirer de l'argent" avec les erreurs humaines

4 Conclusion

Dans ce chapitre nous avons présenté le processus TASSE, qui permet l'identification systématique et la représentation des erreurs humaine, en prenant en entré un modèle de tâches, et fourni en sortie un modèle de tâches avec des erreurs humaines. Ce processus est une amélioration du processus HET de (Stanton, et al., 2006). Le processus permet aussi de modifier un modèle de tâche avec des erreurs humaines, afin de proposer une reprise d'erreur. Nous avons aussi présenté les extensions apportées à l'outils de modélisation et de simulation de tâches HAMSTERS, qui permettent d'effectuer une identification systématique des erreurs humaines.

Nous avons maintenant un outil qui permet d'identifier et de représenter les erreurs humaines dans les modèles de tâches, il nous faut le tester sur une étude de cas de taille réelle, et qui est présenté dans le prochain chapitre, avec les améliorations apportées à la notation et l'outil HAMSTERS afin de modéliser des tâches de taille réelle.

Chapitre 6. – Hamsters 4.0 : des extensions de la notation, et une version stable de l'outil

Dans le Chapitre 1 section 3.2, nous avons présenté l'état de la notation et de l'outil HAMSTERS avant le début de la thèse, certains éléments de la notations HAMSTERS n'avait pas encore été implémentés dans l'outil.

Dans ce chapitre nous allons vous présenter l'état de la notation et de l'outil HAMSTERS l'état de la notation et de l'outil HAMSTERS suite aux modifications apportées pour le support l'approche systématique d'identification et de représentation des erreurs humaines dans les modèles de tâches.

Toutes les modifications apportées ont permis de mieux identifier et décrire les erreurs humaines, en prenant en compte des cas d'utilisations de taille réelle ;

On commencera par identifier les besoins d'améliorations et/ou de corrections, puis on s'intéressera aux extensions au pouvoir d'expression, extension pour l'édition et la visualisation, et on finira par une étude de cas de taille réelle, avec laquelle on présentera toutes les extensions apportées à l'outil de modélisation de tâche, et d'identification des erreurs humaines.

1 Identification des besoins d'amélioration et/ou de correction

Comme nous l'avons présenté dans le chapitre 1, certaines descriptions des erreurs humaines nécessitent la description plus détaillée des objets et informations manipulées. L'extension de la notation de modélisation de tâches HAMSTERS pour la représentation des informations ont été proposé dans (Martinie C. , Palanque, Ragosta, & Fahssi, 2013). Afin d'effectuer une analyse systématique des erreurs humaines sur une études de cas de taille réelle, il a fallu apporter une extension à l'outil de modélisation de tâches HAMSTERS pour la représentation et la description des différents objets proposés dans (Martinie C. , Palanque, Ragosta, & Fahssi, 2013).

Un autre besoin identifié lors d'identification des erreurs humaines, est que souvent une même sous tâche est répété à l'identique avec seulement quelques paramètres qui change. Pour illustrer cet exemple nous allons prendre un simple exemple, la tâche qui est de taper un code à 4 chiffres, les actions de l'utilisateur sont les mêmes pour taper sur un chiffre, ce qui change c'est que la touche sur laquelle il tape, cette description s'effectue à l'aide de composants. Pour l'identification des erreurs humaines, il suffirait seulement d'effectuer une analyse sur le composant, et de propager les erreurs humaines identifiées sur les instances créées.

D'autres besoin identifiées lors du début de la thèse lors d'utilisation et de modélisation d'études de cas de tailles réelles, concernent les outils d'aide à l'édition et la visualisation, tel que la possibilité de faire un Undo/Redo, des Copier/Coller et aussi de pouvoir réorganiser et

justifier les modèles de tâches pour une meilleure lisibilité, aussi des fonctions de filtrage des éléments composant les modèles de tâches, tel que la possibilité d'afficher seulement un certain type d'objet, ou de pouvoir mettre en évidence certain types de tâches.

Pour résumer plusieurs concepts et extensions ont été proposer avant la thèse, mais n'étaient pas encore implémenter dans l'outil HAMSTERS, et au fur et à mesures du besoin lors de modélisation, il a fallu qu'on procède à l'implémentation de certaines fonctionnalités, c'est fonctionnalités sont classé en deux catégories, des extensions qui permettent de nous aider à l'édition et modélisation de modèles de tâches, et d'autres qui nous offres des fonctionnalités d'aide à la visualisation.

En plus de ses besoins, un autre besoin a été pris en compte, et qui est assez important, c'est de corriger les bugs identifiés lors de l'utilisation de l'outil pour les besoins de la thèse, ainsi que les bugs identifiés par les autres utilisateurs.

2 Extension au pouvoir d'expression

Dans cette section je vais vous présenter les deux importantes extensions effectuées durant la thèse, et qui sont plus qu'importante pour l'identification et la représentation des erreurs humaines dans des modèles de tâches. La première est la représentation des différents types d'objet et information manipulé dans les modèles de tâches, et la seconde concerne le pouvoir de description des composants.

2.1 Description des DOD

Les DOD (Data, Object and Device) correspondent à l'objet manipulé lors d'exécution d'une tâche. Ces objets peuvent prendre la forme d'objet physique, informatique ou bien dans la tête d'un utilisateur. Elles permettent de mettre en évidence les différents besoins nécessaires pour réaliser une tâche.

Ici nous allons voir les différents types utilisés dans la notation HAMSTERS, leurs utilisations dans l'outil, et on finira par illustrer par un simple exemple.

2.1.1 Notation

Le pouvoir expressif de HAMSTERS va au-delà de la plupart des autres notations de modélisation de tâches en particulier en fournissant des moyens détaillés pour décrire les données requises et manipulées pour accomplir des tâches. La Figure 12 résume les éléments de notation pour représenter les données. L'information ("Inf:" suivie d'une zone de texte) peut être requise pour l'exécution d'une tâche système. Les objets physiques nécessaires à l'exécution d'une tâche peuvent également être représentés ("Phy O") ainsi que le périphérique (entrée et / ou sortie) avec lequel la tâche est exécutée ("i / o D"). Les connaissances déclaratives et situationnelles peuvent également être explicitées par les éléments « SiK » et « StK ».

Figure 61. Représentation des Données, objets et périphériques dans HAMSTERS

Objet (générique + lié à l'implémentation) : permet d'illustrer n'importe quel type d'objet nécessaire ou produit à l'accomplissement d'une tâche ou n'ayant pas encore de type défini.

Information : permet de décrire un élément de connaissance (donnée, image, voix, …) nécessaire à l'exécution d'une tâche ou susceptible d'être produite après l'accomplissement d'une tâche.

Connaissance (de type déclaratif et/ou procédural) : HAMSTERS permet de représenter les connaissances déclaratives et procédurales nécessaires qui se réfèrent respectivement à ce que l'acteur connait et à des habiletés que l'acteur sait exécuter. Cet objet peut être raffiné en deux sous-types : stratégique et situationnel.

- ***Connaissance déclarative***

Hamsters permet de décrire le savoir nécessaire à l'accomplissement d'une tâche.

Tableau 15. Représentation des types de connaissance avec la notation HAMSTERS

Type de connaissance	Représentation HAMSTERS
Connaissance déclarative	DK: Declarative knowledge
Connaissance déclarative stratégique	StK: Strategic knowledge
Connaissance déclarative situationnelle	SiK: Situational Knowledge

Le Tableau 6 représente les différentes présentations de types connaissance ; la connaissance déclarative est raffinée en deux types, stratégique et situationnel.

- ***Connaissance procédurale***

Un modèle de tâche HAMSTERS doit permettre de décrire une activité pour atteindre un but (ceci est une information de type « procédural »).

Il permet donc de décrire des connaissances procédurales (procedural knowledge) nécessaires à l'accomplissement d'une tâche :

- De type Stratégique (voir Figure 13)

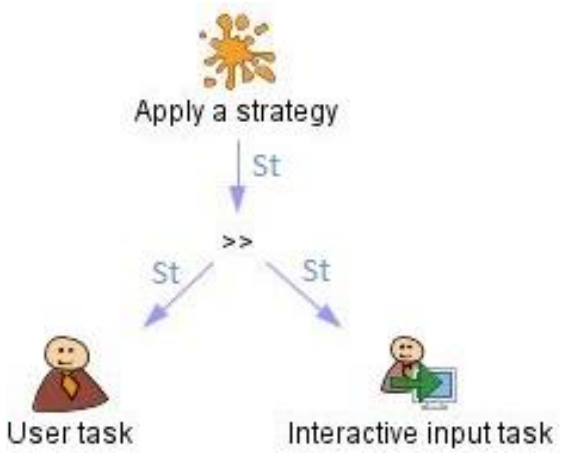


Figure 62.Représentation d'une connaissance procédurale de type stratégique avec la notation HAMSTERS

- De type Situationnel (voir Figure 14)

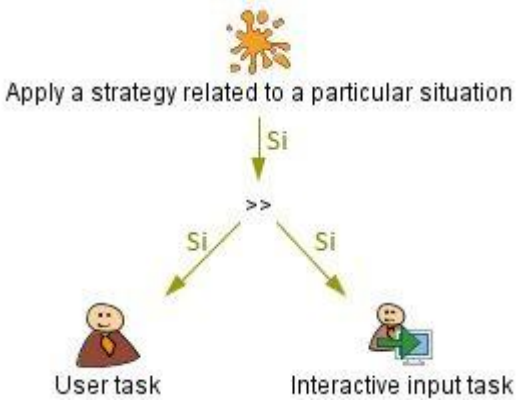


Figure 63.Représentation d'une connaissance procédurale de type situationnel avec la notation HAMSTERS

Application : permet de représenter une application liée ou générée à l'exécution d'une tâche.

Périphérique (device) : permet de représenter un périphérique utilisé pour exécuter une tâche interactive.

Ces DOD sont connecté aux tâches à l'aide de différents types de connexions :

Stocker : permet de dire qu'un objet a été créé (voir Figure 64)

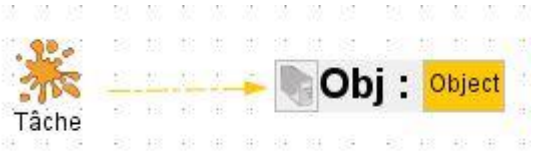


Figure 64. La tâche abstraite crée et stock un objet

Accéder : permet de dire qu'une tâche accède à un objet (voir Figure 65)

Figure 65. La tâche abstraite accède à un objet

Supprimer : permet de dire qu'une tâche consomme un objet (voir Figure 66)

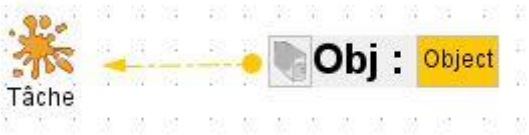


Figure 66. La tâche abstraite consomme un objet

Modifier : permet de décrire qu'une tâche procède à la modification d'un objet (voir Figure 67)

Figure 67. La tâche abstraite modifie un objet

Tester : permet de décrire qu'une tâche effectue un test sur un objet (voir Figure 68)

Figure 68. La tâche abstraite teste un objet

A ces connexions peuvent être associé des opérations d'affectation ou de test :

Affectation : donner une valeur à un objet (voir Figure 69)

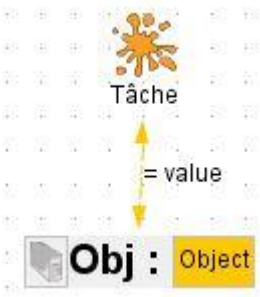


Figure 69. La tâche affecte une valeur à l'objet

Egal : test si la valeur de l'objet est égal à une valeur donnée (voir Figure 70)

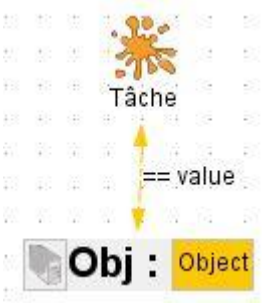


Figure 70. La tâche test si la valeur de l'objet est égale à "value"

Différent : test si la valeur de l'objet est différente d'une valeur donnée (voir Figure 71)

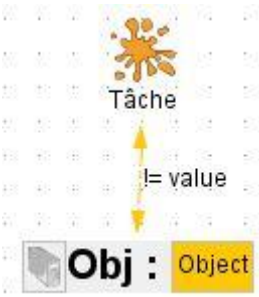


Figure 71. La tâche test si la valeur de l'objet est différente de "value"

Supérieur : test si la valeur de l'objet est supérieur à une valeur donnée (voir Figure 72)

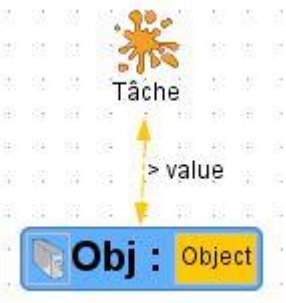


Figure 72.La tâche test si la valeur de l'objet est supérieur à "value"

Inférieur : test si la valeur de l'objet est inférieur à une valeur donnée (voir Figure 73)

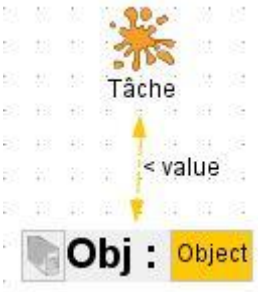


Figure 73. La tâche test si la valeur de l'objet est inférieure à "value"

Supérieur ou égal : test si la valeur de l'objet est supérieur ou égal à une valeur donnée (voir Figure 74)

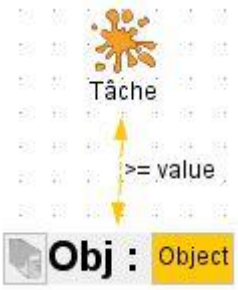


Figure 74. La tâche test si la valeur de l'objet est supérieure ou égale à "value"

Inférieur ou égal : test si la valeur de l'objet est inférieur ou égal à une valeur donnée (voir Figure 75)

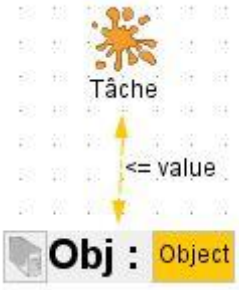


Figure 75. La tâche test si la valeur de l'objet est inférieure ou égale à "value"

2.1.2 Outil

Une extension de l'outil HMASTERS a été effectué afin de pouvoir décrire les différents types de données et de leurs connexions :

Edition des DOD :

Les différents DOD se trouve dans la palette d'outil d'HAMSTERS, dans la catégorie « Data, Object, Device (DOD) » (voir la partie encadrée en rouge dans la Figure 76). Pour utiliser ces objets, il suffit de faire un simple glisser/déposer de la palette vers le modèle de tâche, en sélectionnant l'objet souhaité (voir Figure 77).

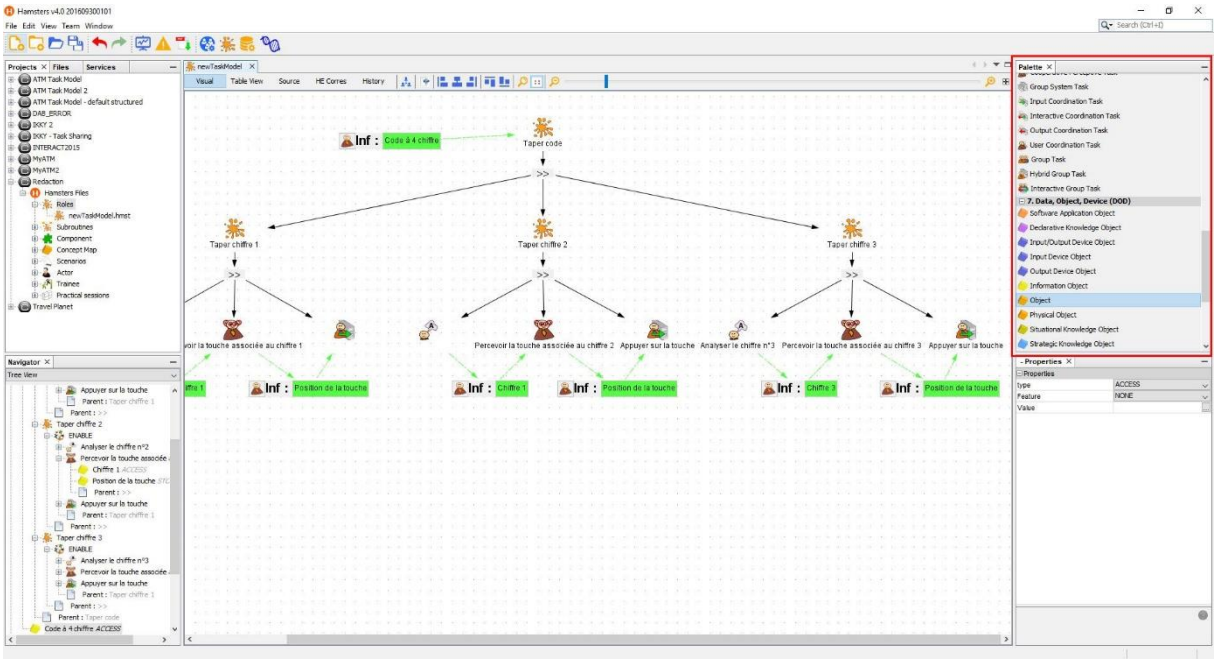


Figure 76. L'onglet Palette de l'outil HAMSTERS

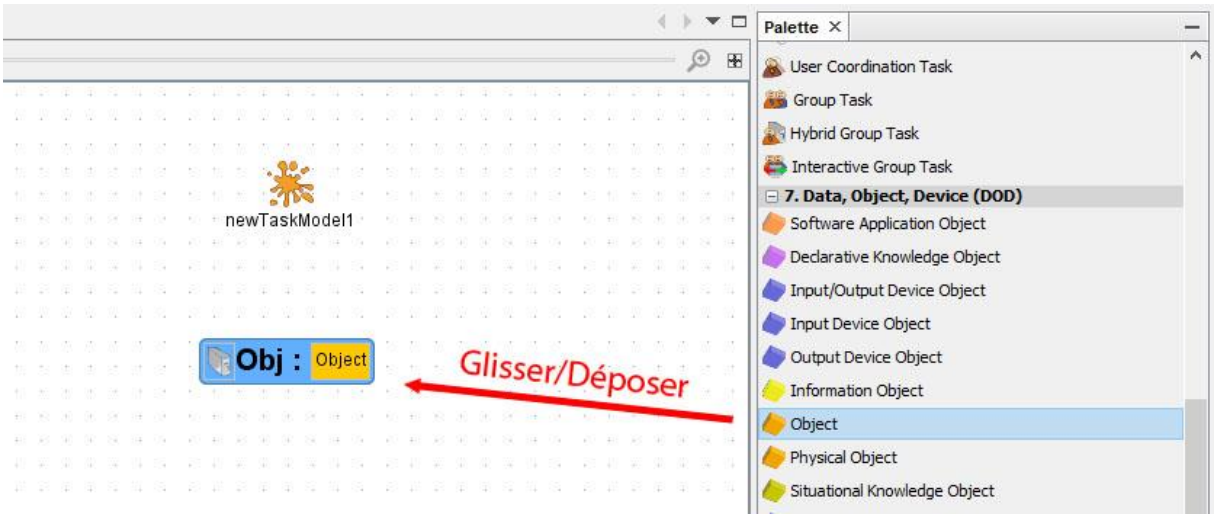


Figure 77. Glisser/déposer un objet à partir de la palette vers le modèle de tâches

Une fois l'objet ajouter dans le modèle de tâche, nous pouvons le modifier :

Modification du type de l'objet :

Méthode 1 : un double clic sur le texte « Obj », puis un menu déroulant s'ouvre, et on choisit le type d'objet (voir Figure 78)

Méthode 2 : Modification dans la partie propriété, en sélectionnant sur l'objet (voir Figure 79)

Modification de la description de l'objet :

Méthode 1 : un double clic sur la description de l'objet, le texte se transforme en un « input text » afin de modifier la description (voir Figure 80).

Méthode 2 : Modification dans la partie propriété, en sélectionnant sur l'objet (voir Figure 81)

Modification de la couleur de fond du type de DOD : la modification de la couleur s'effectue à partir de la fenêtre de propriété. Le changement de la couleur sera répercuté sur tous les DOD du même type (voir Figure 82)

Définir ou non le nombre d'instance de l'objet : Cocher ou décocher la case associée dans la fenêtre de propriété, quand c'est coché la valeur 0 sera prédéfini (voir Figure 83).

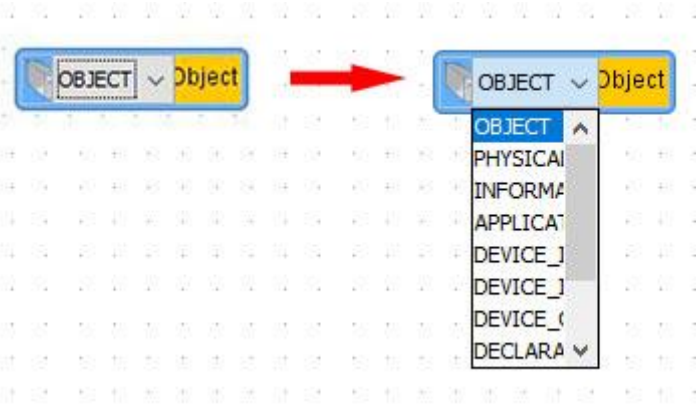


Figure 78. Modification du type d'un DOD depuis la partie graphique

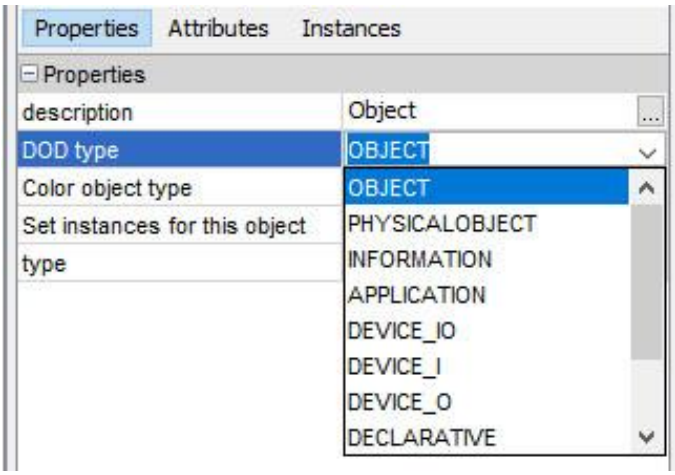


Figure 79. Modification du type d'un DOD depuis le panneau de propriété

Figure 80. Modification de la description d'un DOD à partir de l'interface graphique

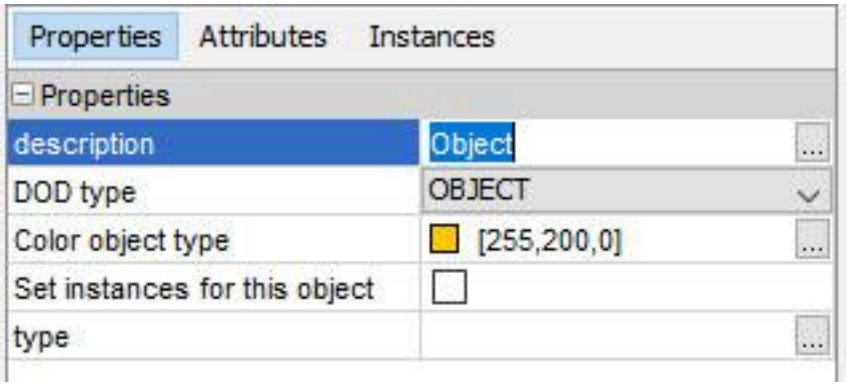


Figure 81. Modification de la description d'un DOD à partir de la fenêtre de propriété

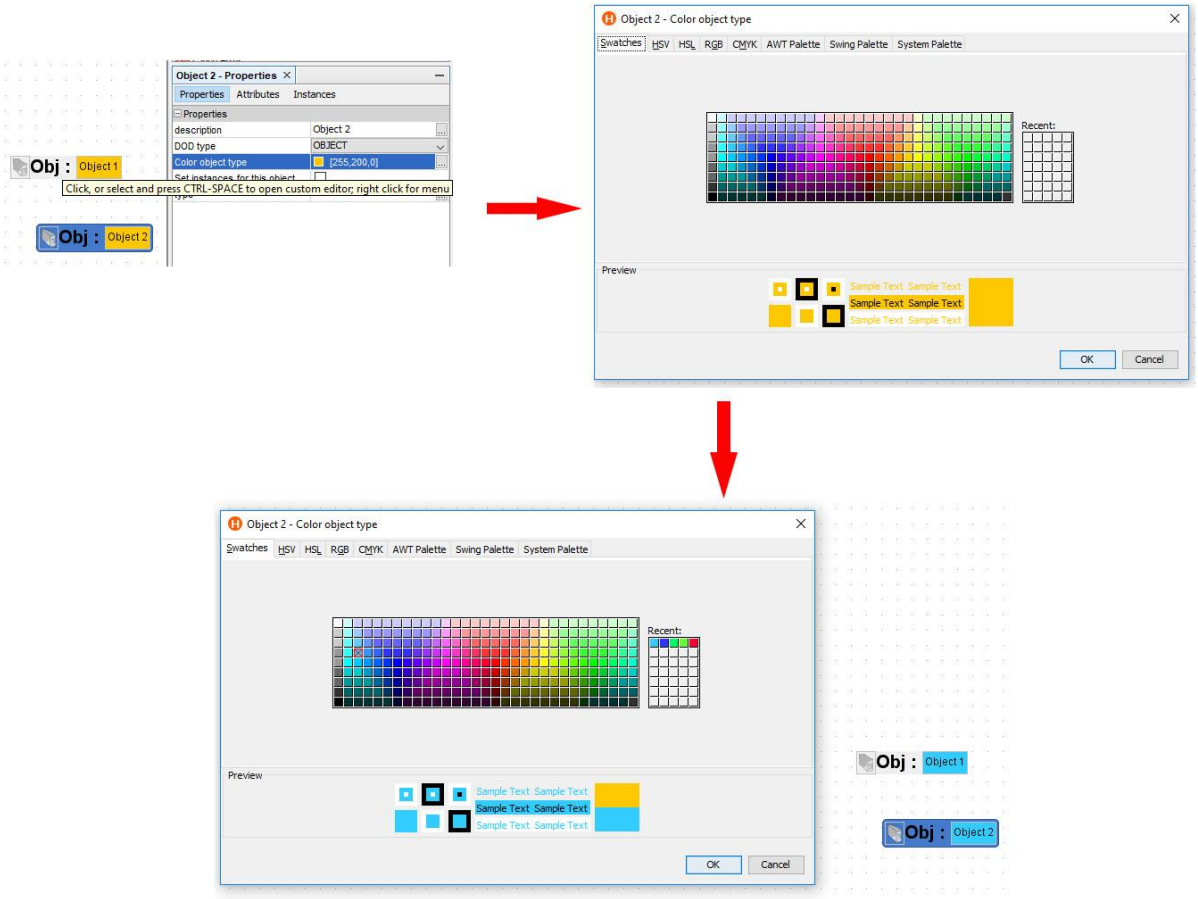


Figure 82. Changement de la couleur de fond d'un type de DOD sélectionné

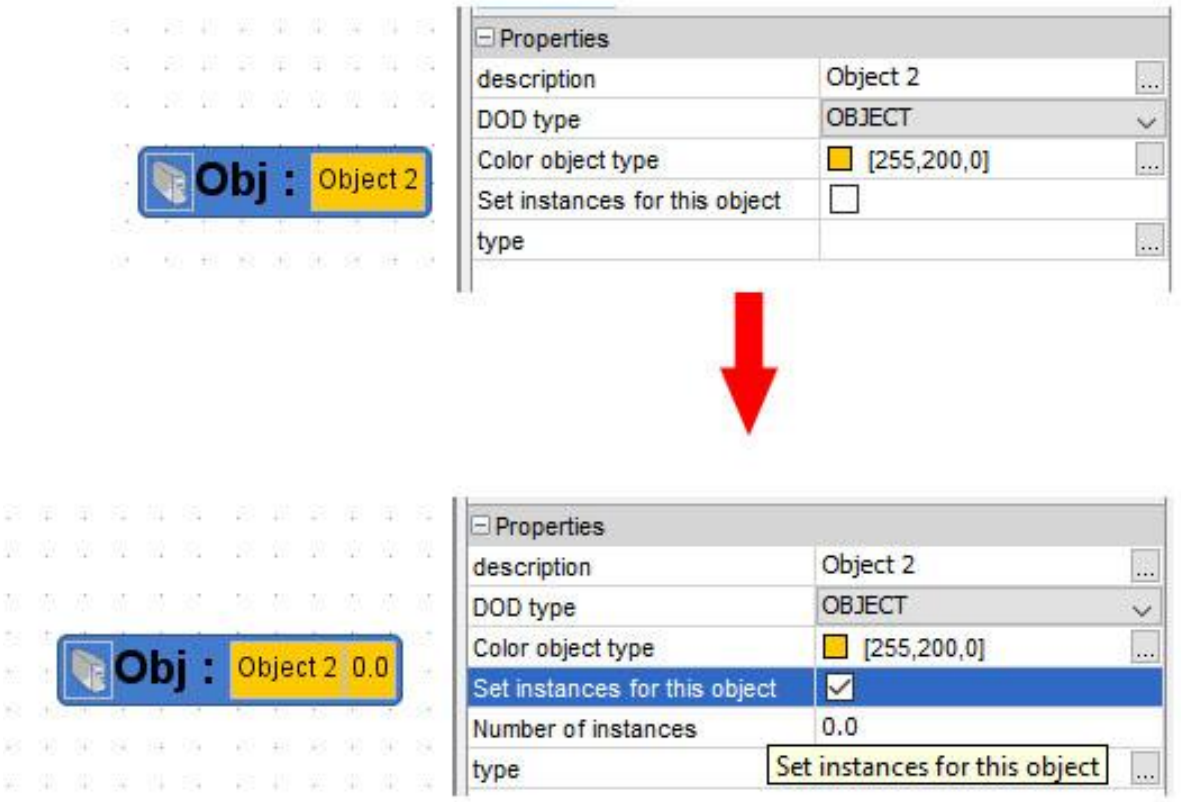


Figure 83. Définir ou non un nombre d'instance de DOD

Connexion des DOD :

Un objet ajouté au modèle de tâches, doit être associé à un ou plusieurs tâches. La connexion s'effectue en restant appuyer sur la touche « CTRL » et en tirant un arc entre l'objet et la tâche. Une fois la connexion créée, nous pouvons procéder à sa modification :

Modification du type de connexions :

Méthode 1 : Sur le modèle de tâche, en double-clic sur la connexion qu'on veut modifier, puis un Pie menu s'ouvre, et là on sélectionne le type voulu (voir Figure 84).

Méthode 2 : à l'aide de la fenêtre de propriété, et en sélectionnant la connexion à modifier, en clic sur le menu déroulant associé au type, puis on sélectionne le type (voir Figure 85).

Modification du type d'opération :

En sélectionnant la connexion voulue, on va dans la fenêtre de propriété, puis on ouvre le menu déroulant associé à « feature » puis en sélectionne l'opération souhaité, et en dessous en rentre la valeur associé à l'opération (voir Figure 86)

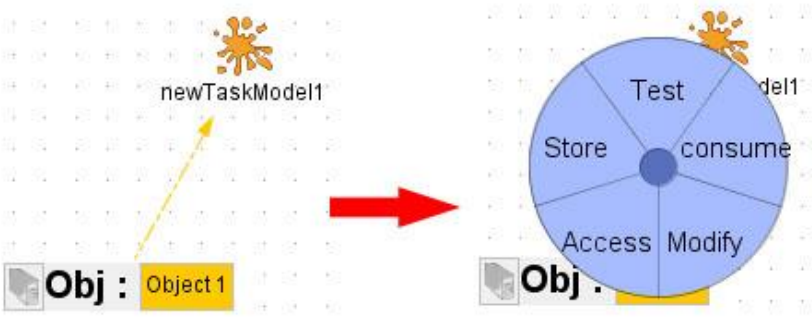


Figure 84. Modification du type de connexion des DOD à partir du modèle de tâches

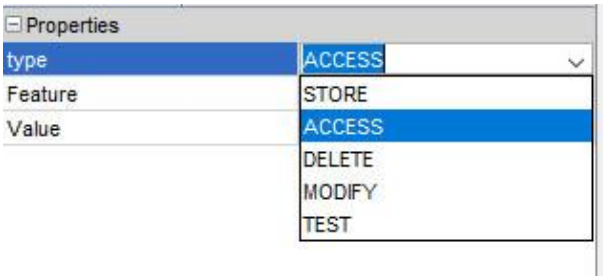


Figure 85. Modification du type de connexion des DOD à partir de la fenêtre de propriété

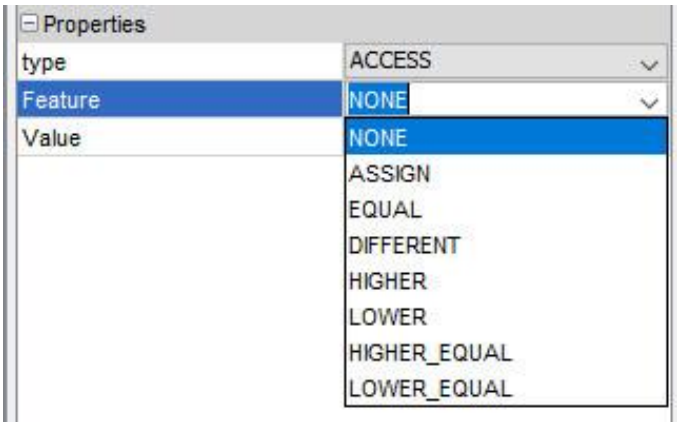


Figure 86. Modification du type d'opération à attribué à la connexion

2.1.3 Exemple illustratif

Pour illustrer l'utilisation des DOD, nous allons prendre un simple exemple de tâche utilisateur qui est de régler la température du thermostat (voir Figure 105).

Dans ce modèle de tâche, l'utilisateur doit régler la température du thermostat à 22°C, cette précision est liée à la connaissance de l'utilisateur qui dit que la température idéale dans un séjour est de 22°C. On représente cette connaissance par un DOD de type « Connaissance déclarative », et qui est connecté à la tâche « Régler la température » dans le modèles de tâches « Régler la température du thermostat ».

Afin de connaitre s'il faut baisser ou augmenter la température, l'utilisateur doit percevoir la température affichée sur le thermostat, et qui devient une information dans la tête de l'utilisateur. Dans le modèle de tâches cette informations est représenter par un DOD de type information, et qui est connecté à la tâche « Percevoir la température indiquer sur le thermostat », son type de connexion est une connexion de stockage, cette information permet de sélectionner la tâche à effectuer entre « baisser le thermostat » et « Augmenter le thermostat », toutes les deux sont des tâches filles de la tâche « Régler la température » et l'opérateur de choix. Chacune de ces deux tâches effectue un test sur l'information « température indiquée » à l'aide d'un opérateur supérieur pour l'une et inférieur pour l'autre, et comme valeur de comparaison, la valeur de la connaissance, qui est de 22°C.

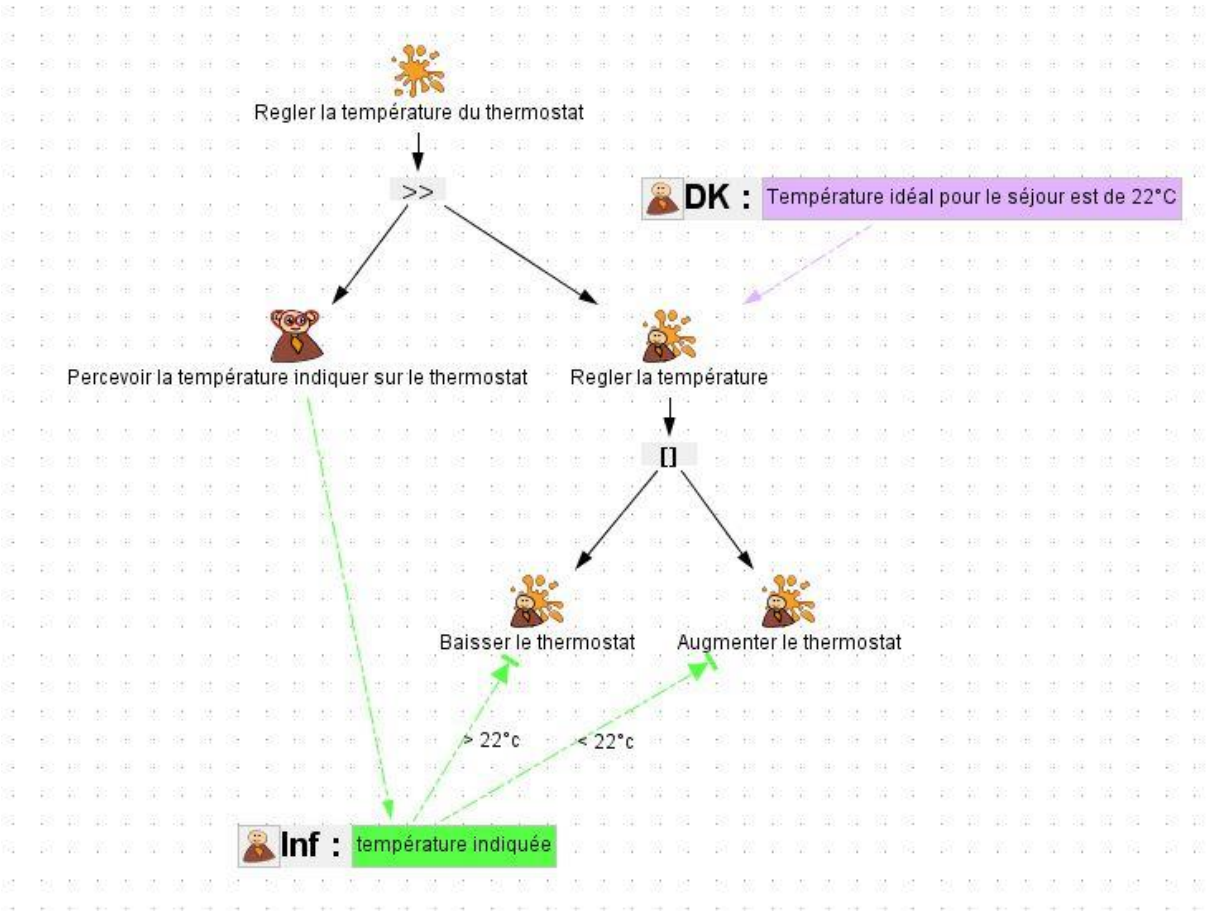


Figure 87. Modèle de tâche "Régler la température du thermostat"

2.2 Description des composants

Les composants permettent de créer un modèle de tâches type paramétré, afin de pouvoir instancier des modèles de tâches identique mais avec des valeurs en paramètre différent. Prenons comme exemple une tâche qui est d'appuyer sur la lettre « A » du clavier. Les actions de cette tâche sont les mêmes que celle de l'appuie sur n'importe quelle touche du clavier, le seul élément qui change est le device sur lequel la tâche est exécutée. Ici nous pouvons seulement définir un modèle de tâches générique paramétré (composant), et mettre en paramètre le device sur lequel est exécuté la tâche.

L'intérêt des composant pour l'identifications systématique des erreurs humaines, c'est de pouvoir effectué une seule fois l'identification sur le composant type, et toutes les erreurs identifiées seront répercuté sur les différents instances créées.

Dans cette sous-section, nous allons présenter comment se présente la description des composants avec la notation HAMSTERS, ainsi comment les décrire à l'aide de l'outil HAMSTERS, et on finira par un exemple illustratif sur la saisie du code PIN dans le distributeur automatique de billet.

2.2.1 Notation

La Figure 88 représente l'utilisation des composants à l'aide de la notation HAMSTERS. Le type de la tâche représentant le but est représenter par la même icône qu'une tâche abstraite, et avec une icône représentant un composant de chaque côté. La description de cette tâche est suivie par une liste de nom de paramètre ([Param1, Param2]). Ces paramètres sont réutilisés dans le reste du modèle à l'aide de deux symbole « < » et « > », ici <Param1> pour la réutilisation du premier paramètre.

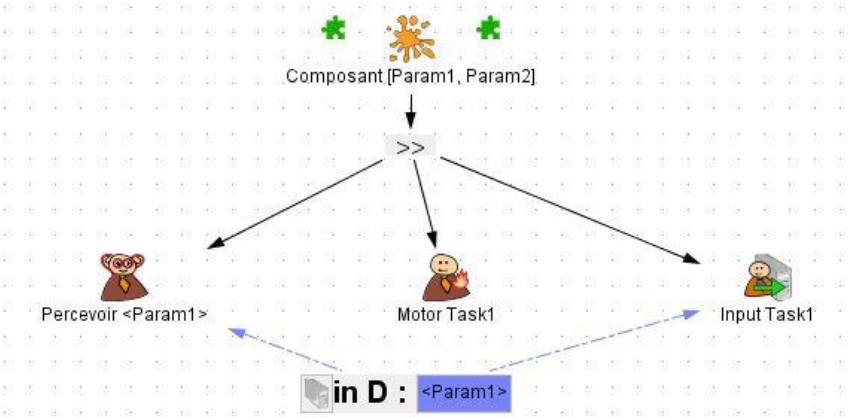


Figure 88. Représentation du composant avec la notation HAMSTERS

2.2.2 Outil

L'outil HAMSTERS permet la création, la personnalisation et l'instanciation de composants. Pour commencer il suffit de créer un nouveau modèle de tâches de type composant dans le dossier « composants » (voir Figure 89), en faisant un clic droit sur le dossier, puis « New » puis « Component.hmst ». Puis on donne une description du composant.

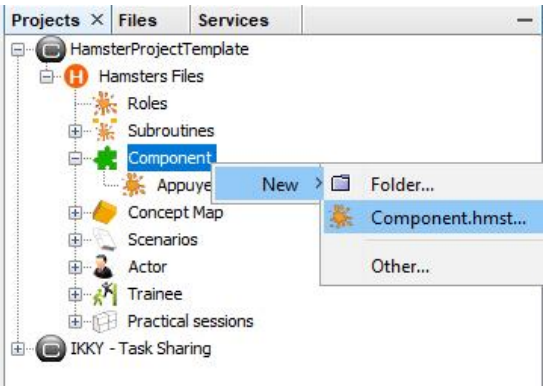


Figure 89. Pop-up menu permettant la création d'un nouveau composant type

2.2.3 Exemple illustratif

Afin de mieux expliquer l'utilisation de l'outil HAMSTERS pour la création de composant, nous allons illustrer tout cela par un exemple illustratif qui est la tâche de « taper le code PIN d'une CB » (voir Figure 90).

Le code PIN d'une carte bleu est composé de 4 chiffres, afin d'introduire ce code, l'utilisateur devra effectuer 4 fois la même action sur des touches différentes. Pour cette exemple le code PIN est 8457.

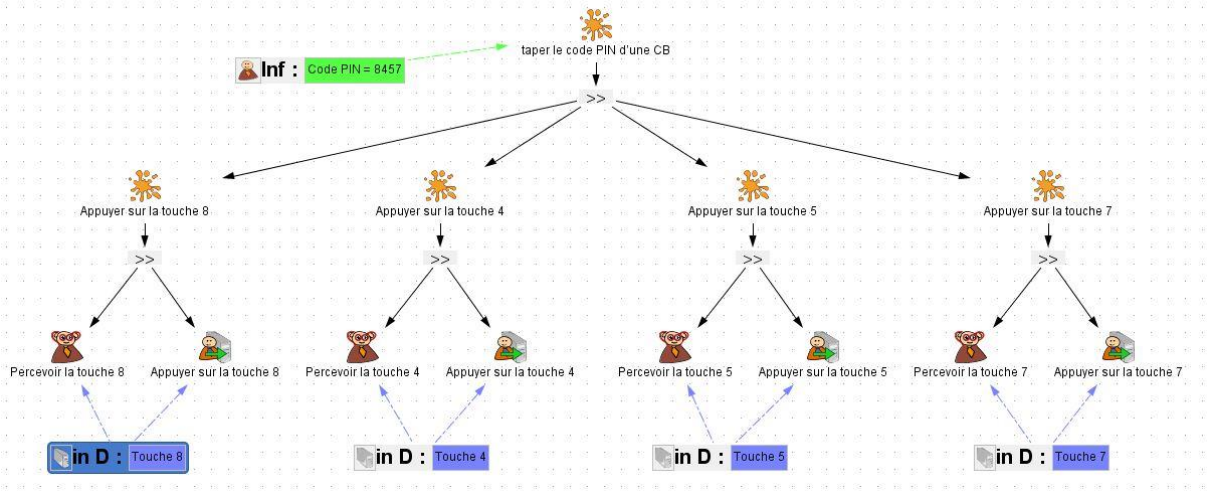


Figure 90. Modèle de tâches "Taper le code PIN d'une CB"

Nous créons un nouveau composant avec comme description « Appuyer sur une touche ». Puis nous éditons le modèle de tâches du composant à l'aide de la palette d'outil pour obtenir un modèle de tâches identique à celui de l'une des branches vues dans la Figure 90, afin d'obtenir un modèle de tâche type d'appui sur une touche (voir Figure 91).

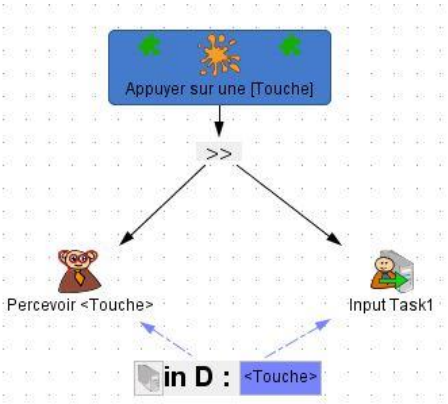


Figure 91. Modèle de tâche du composant "Appuyer sur une touche"

Pour ajouter ou éditer les paramètres en entrée du composant, il suffit de sélectionner le composant à la racine du modèle, puis dans le panneau de propriété de la tâche, on clique sur « Component » et là on peut soit rajouter des paramètre, soit modifier le nom du paramètre (voir Figure 92)

Figure 92. Panneau de propriété d'une tâche "composant"

Maintenant que le modèle de tâches du composant type est créé, nous allons l'instancier dans notre modèle de tâches « taper le code PIN d'une CB » (voir Figure 90). Nous remplaçons les branches « Appuyer sur la touche 8, 4, 5 et 7 » par une tâche de type composant (voir Figure 93)

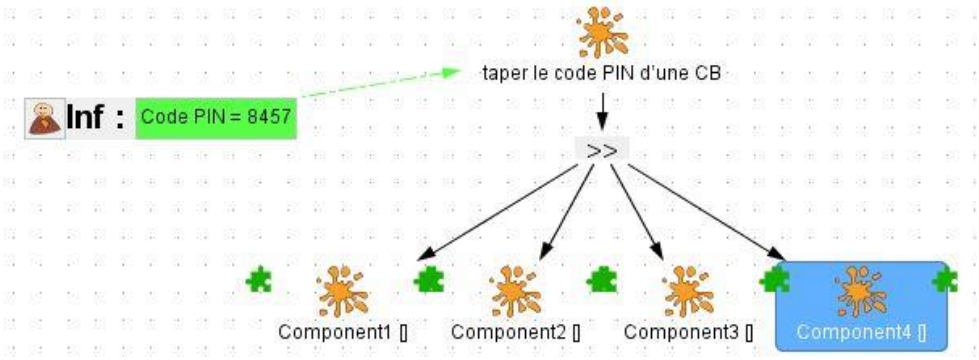


Figure 93. Remplacement des branche "Appuyer sur la touche 8, 4, 5 et 7" par des composants dans le modèle de tâches "taper le code PIN"

Après, afin de créer une instance pour chaque branche, il suffit de cliquer avec le bouton droit sur chaque tâche composant, puis cliquer sur « Instantiate component » (voir Figure 94)

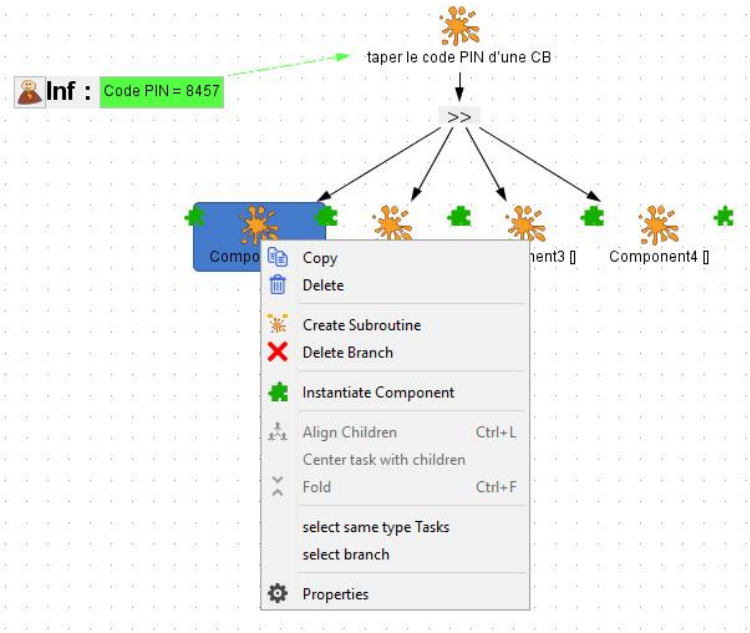


Figure 94. Pop-up menu d'une tâche composant

En cliquant sur « Instantiate Component », Un panneau s'ouvre afin de choisir le composant à instancier, puis donner des valeurs à tous les paramètres du composant, et finir par cliquer sur « Instantiate » (voir Figure 95).

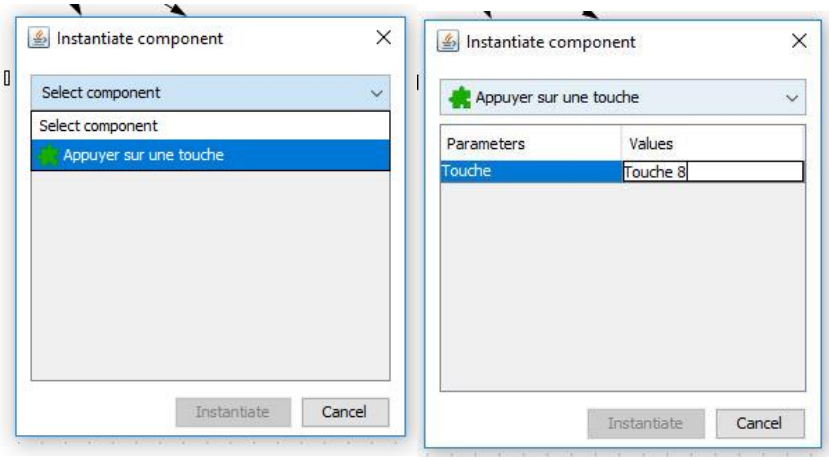


Figure 95. Panneau d'instanciation d'un composant avec le renseignement de la valeur de chaque paramètre

Une instance du composant « Appuyer sur une touche » se crée (voir Figure 96), et la tâche « component1[] » devient « Appuyer sur une [Touche 8] » (voir Figure 97). Après avoir fait ça pour tous les composant, on obtient un modèle de tâches avec des composants instanciés (voir Figure 98)

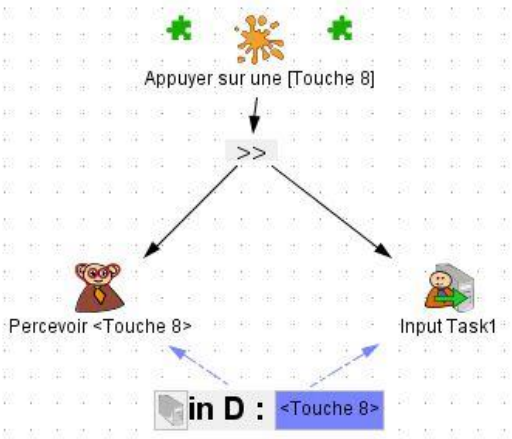


Figure 96. Instance du composant "Appuyer sur une touche" avec comme paramètre "Touche 8"

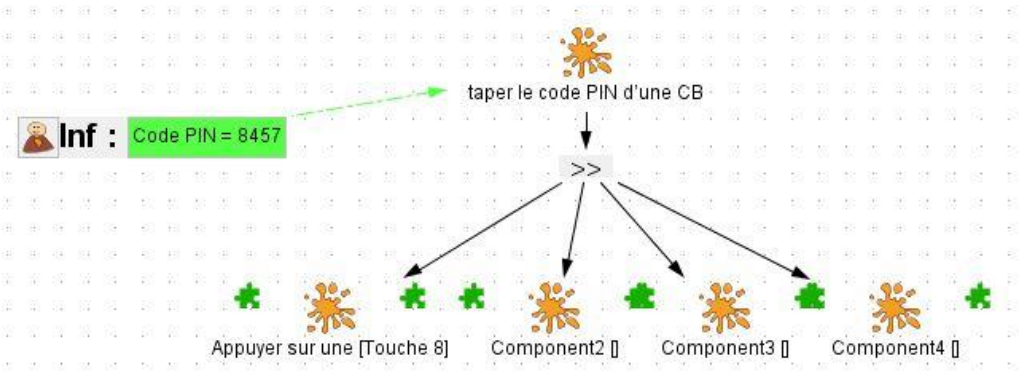


Figure 97. Modèle de tâche "Taper le code PIN d'une CB" avec un composant instancié

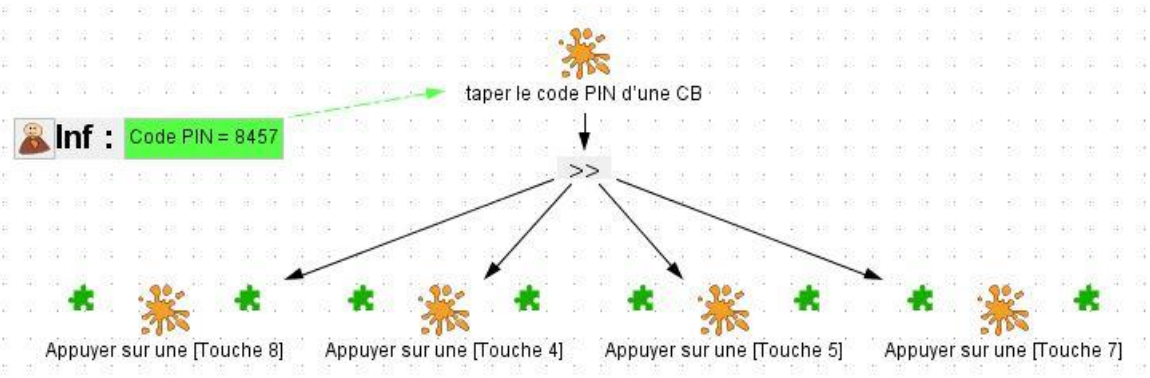


Figure 98. Modèle de tâche "Taper code PIN d'une CB" avec les 4 composants instanciés

3 Extension pour l'édition et la visualisation

Pour une étude de cas de taille réelle, tel que celle qu'on verra dans la prochaine section, plus de 400 éléments de la notation sont édités. Les éléments qu'on retrouve dans cette étude de cas, sont les différents types de tâches, d'opérateurs et de DODs, ainsi que l'édition des liens entre ces éléments. Pour cela il a fallu implémenter plusieurs fonctionnalités permettant à aider l'utilisateur à éditer ses modèles de tâches.

Aussi, avec un nombre si important d'éléments de la notation à manipuler, impossible d'avoir une vue globale sur tous ces éléments sur un simple écran d'ordinateur. Pour cela on a implémenté plusieurs fonctionnalités qui permettent d'aider l'utilisateur à visualiser ses modèles de tâches.

Après avoir éditer les modèles de tâches, certains utilisateurs procède à l'analyse de ces modèles. Nous avons proposé pour cela, quelques fonctionnalités permettant à effectuer une analyse des modèles de tâches. Les fonctionnalités implémentées sont seulement ceux d'ont on avait besoin pour notre étude de cas.

3.1 Fonctionnalité d'aide à l'édition

Dans cette sous-section, nous allons vous présenter les différentes fonctionnalités permettant d'aider à l'édition de modèles de tâches.

3.1.1 Copier/Coller

La fonctionnalité du copier/coller est utilisé dans la plupart des outils d'édition de modèles. C'est une fonctionnalité assez importante pour l'édition de modèles de tâches. Nous avons implémenté cette fonctionnalité sur l'outil HAMSTERS de deux manières déférentes, la première est de pouvoir sélectionner une ou plusieurs tâches, et faire copier/coller sur le même modèle ou un autre et ça copie seulement la ou les tâches sélectionnées. La deuxième est en sélectionnant une tâche et en faisant un copier/coller la branche, cela permet de copier toute l'arborescence de la tâche sélectionnée, ainsi que les DODs connectés à la branche.

L'utilisation de cette fonctionnalité se fait de la même manière que n'importe quel outil d'édition, avec un simple CTRL+C / CTRL+V, ou bien faire un clic droit sur un élément et sélectionné l'option de copie souhaitée (tâche ou arbre).

3.1.2 Undo/Redo

La fonctionnalité du Undo/Redo est l'une des fonctionnalités la plus importante dans les outils d'édition et de modélisation, il est important de fournir la possibilité de récupérer une erreur, en cas d'erreur, sans avoir besoin de recommencer le travail depuis le début.

L'utilisation de cette fonctionnalité se fait aussi de la même manière que ce qui existe, en faisant un CTRL+Z pour le Undo, et CTRL+Y pour le Redo.

3.1.3 Justification des modèles

Lors de l'édition de gros modèles, on ne se retrouve avec pas mal d'éléments graphiques à réordonner et agencer, une tâche qui prend assez de temps, surtout si on doit dé zoomer et scroller pour visualiser le modèle en entier.

Pour remédier à cela, nous avons implémenté une fonctionnalité, qui permet de sélectionner une tâche, et réorganiser automatiquement la branche de cette dernière, en faisant un clic droit sur la tâche mère, puis cliquer sur « Align children » (voir Figure 99), pour obtenir un modèle organisé, avec des éléments alignés (voir Figure 100).

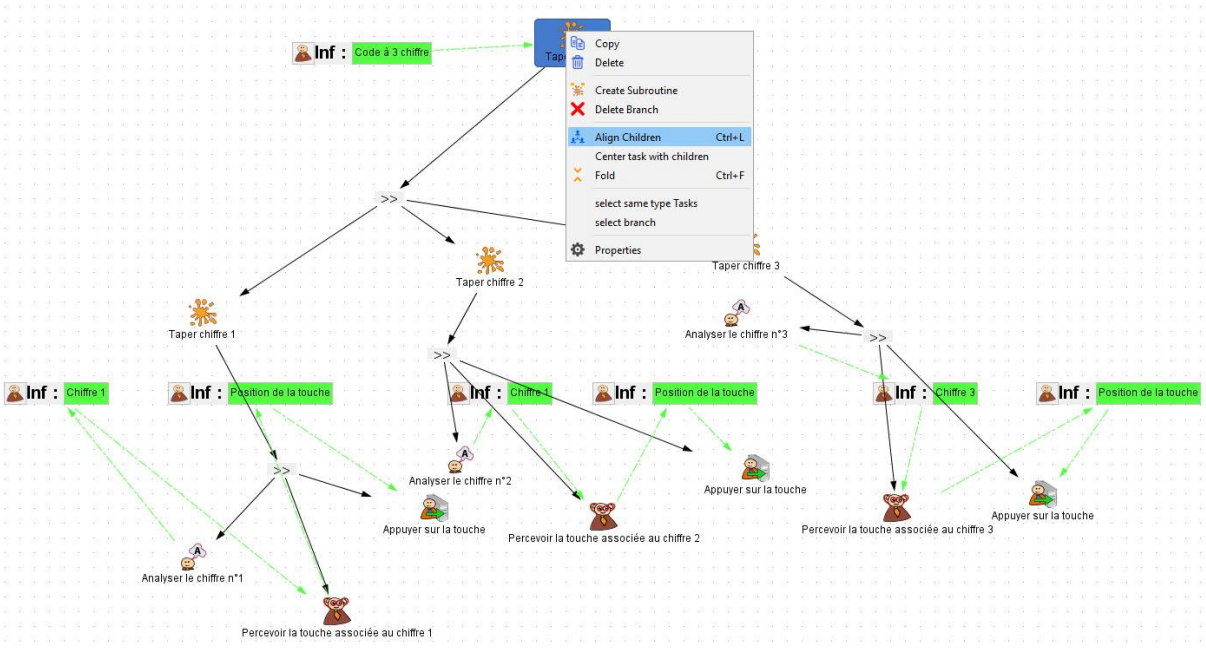


Figure 99. Modèle de tâche « taper code » avant la réorganisation

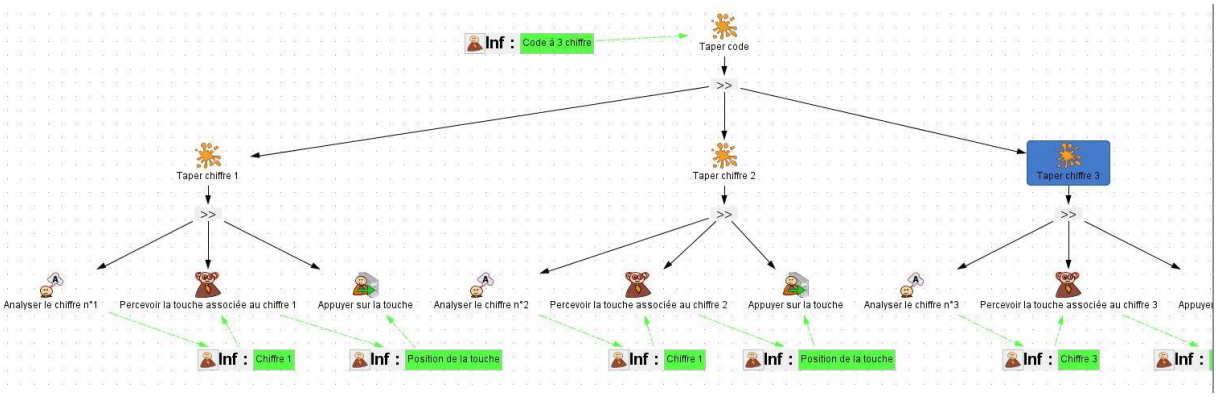


Figure 100. Modèle de tâche « taper code » après la réorganisation

3.2 Fonctionnalité d'aide à la visualisation

Dans cette section nous allons présenter les différentes fonctionnalités qui aide l'utilisateur à visualiser les modèles de tâches.

3.2.1 Filtrage

Un modèle de tâches se compose de de trois éléments, les tâches, les opérateurs et les DODs. Et maintenant on peut y rajouter des éléments nécessaires à la représentation des erreurs humaines. Ces éléments-là sont connectés entre par des flèches, ce qui peut donner en résultat final, un modèle de tâches assez complexe à lire. Pour cela nous avons implémenter une fonctionnalité de filtrage des éléments hamsters.

Ce filtrage se base sur un système de calques, on a le premier calque de base qui contient les éléments de tâches ainsi que les opérateurs.

Le deuxième calque contient les différents types de DODs. Dans ce calque, on peut masquer tous les types de DODs, ou bien juste quelques-uns (voir Figure 101).

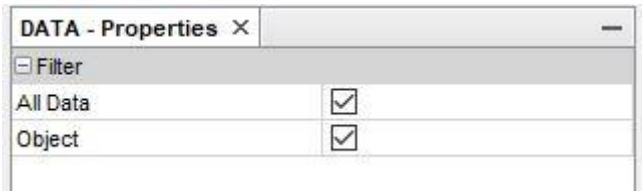


Figure 101. Figure représentant le système de filtrage des DODs

Le troisième calque permet d'afficher ou de masquer les différents types d'erreurs humaines, tous les types de génotype (voir Figure 102 a)), et le phénotype (voir Figure 102 b)).

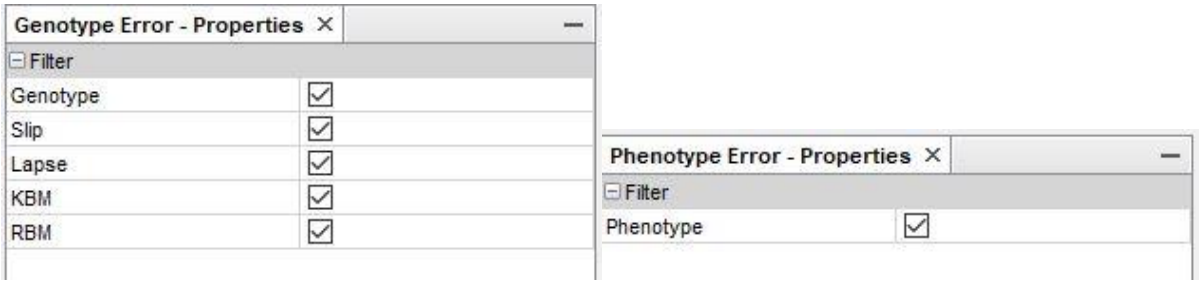


Figure 102. a) Système de filtrage des génotypes b) Système de filtrage du phénotype

3.2.2 Iconnification des icônes en format vectoriel et exportation des modèles en différents formats

L'analyse de modèles de tâches peut s'effectuer à partir de support papier. D'où l'intérêt d'avoir transformé toutes les icônes en format vectoriel, afin d'obtenir une meilleure qualité d'image lors des impressions sur papier, ou lors du zoom sur les modèles.

Les modèles de tâches peuvent être exporter en image, afin d'être insérer dans des documents ou être imprimer. Pour cela nous avons implémenter un panneau d'exportation (voir Figure 103), qui permet de choisir différents formats de sortie (jpeg, png, pdf, svg), et aussi de personnaliser la taille de sortie, ou de cadrer une zone.

Dans le panneau de la Figure 103, nous retrouvons deux sous panneaux, celui de droite qui permet de visualiser le modèle à exporter, et celui de gauche permet de paramétrer l'exportation, dans ce dernier, nous retrouvons de haut en bas :

- Choix de format de sortie (jpeg, png, pdf ou svg)
- Chemin de sauvegarde du fichier de sortie
- Cadrage ou non de la zone pour le fichier de sortie, si la case est cochée, l'image en sortie sera seulement celle qui est affichée dans le panneau de droite. La qualité de l'image de sortie
- La taille de l'image, soit en sélectionnant des paramètre prédéfini (niveau de zoom actuel, taille actuelle, adapté à la fenêtre ou bien personnaliser en choisissant la largeur et la hauteur de l'image, ainsi que le zoom)

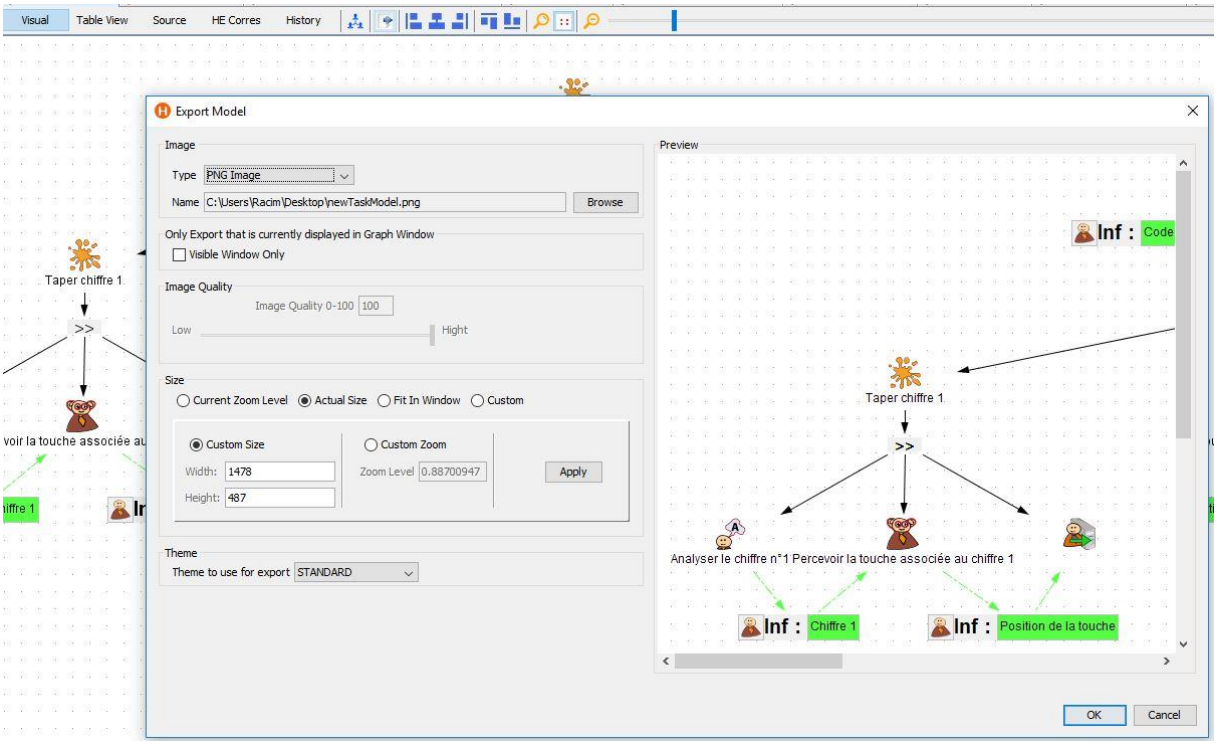


Figure 103. Panneau d'exportation de modèles de tâches

3.2.3 Implémentation d'une vue tabulation des modèles

La notation de modélisation de tâches HAMSTERS permet de représenter les modèles de tâches en arborescence (nœud, branche et feuille), cela dit ce format de représentation, ne permet pas d'avoir une vue d'ensemble sur tous les nœuds ou feuilles avec leurs paramètres, d'où l'implémentation de la vue tabulation des modèles de tâches (voir Figure 104).

Cette vue permet d'avoir une vue globale sur les paramètre des tâches. Les colonnes de gauche à droite représentent la description de la tâche, le type de la tâche, est-ce une tâche itérative ou non, est-ce qu'elle est optionnelle ou non, le temps minimum pour exécuter la tâche et le temps maximum.

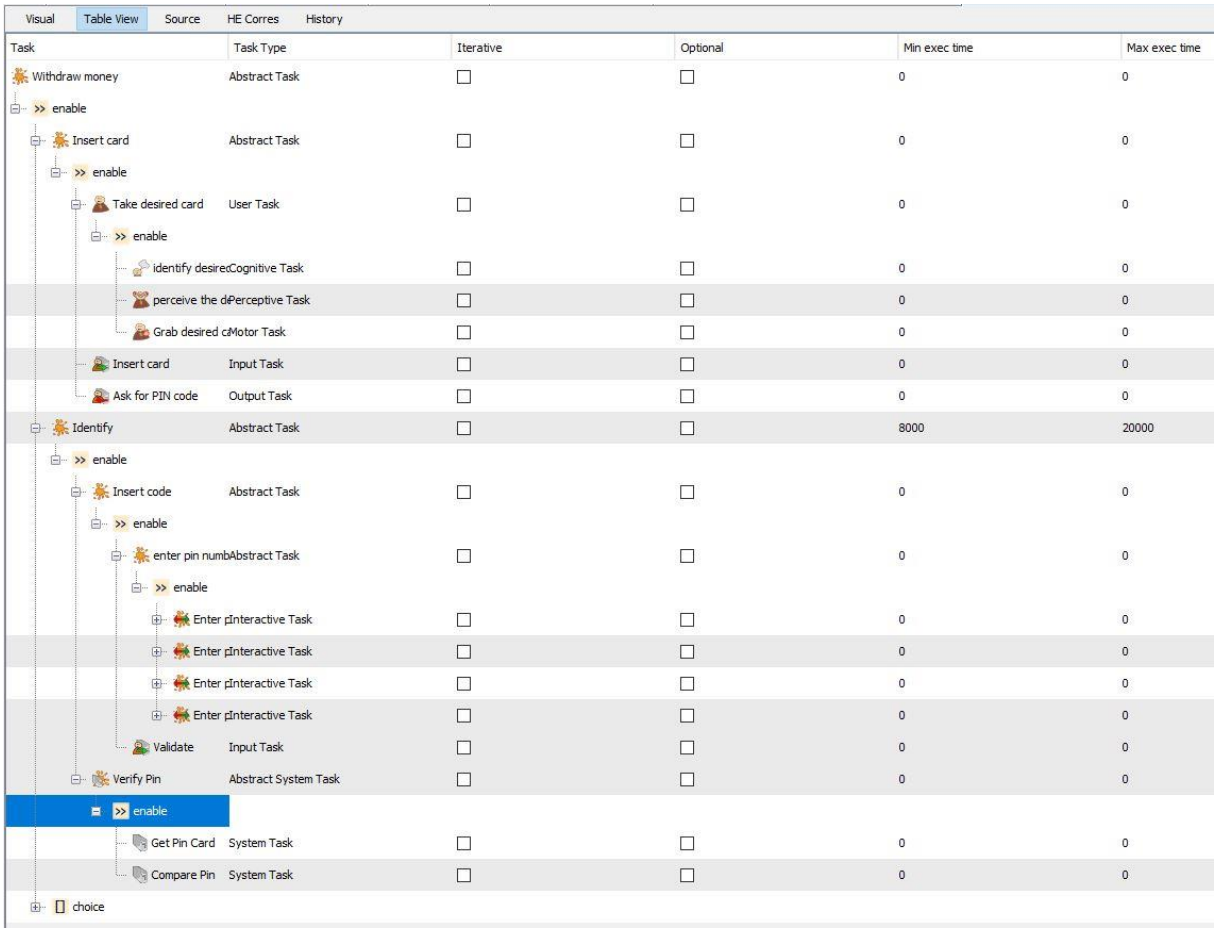


Figure 104. Vue tabulation du modèle de tâches "retirer de l'argent"

3.3 Fonctionnalité d'aide à l'analyse

3.3.1 Implémentation d'un panel de statistique

Le panel de statistique permet de donner une représentation graphique sur le nombre de de tâches que contient un modèle de tâche, le nombre de chaque type de tâches, le nombre de DOD manipulé dans le modèles et par type, ainsi que le nombre d'opérateur utilisé dans le modèle

Il permet aussi d'obtenir le nombre de tâches exécutées lors d'une simulation, et faire une comparaison entre le nombre total de tâches contenu dans le modèles, et le nombres de tâches exécutées à partir d'un scénario.

Pour accéder au panel de statique en sélectionne le modèle ou le projet souhaitant analyser, puis en clic dans la barre d'outils sur l'icône  puis le panel s'ouvre (voir Figure 105).

Dans le panel « statistics » de la Figure 105 contient 4 sous panel, les deux du haut concernent le nombre d'élément d'un ou des modèles de tâche, et ceux du bas concernent les éléments du ou des scénarios. Dans le panel du haut-gauche, on retrouve : une liste déroulante pour choisir le modèle à analyser ou la totalité des modèles du projet, on retrouve aussi le nombre total des tâches, le nombre total des opérateurs, le nombre d'opérateurs de séquence utilisés, ainsi que

les opérateurs de désactivation, de concurrence, de choix, de stop/reprise et d'ordre indépendant, et les dernières lignes concernent le nombre total des objets manipulés et du nombre de chaque type d'objet, dans l'ordre (Connaissance déclarative, connaissance stratégique, connaissance situationnelle, périphérique d'entrée, périphérique de sortie, périphérique d'entrée/sortie, information, objet, objet physique, application et temps).

Dans le panel du haut-droit, on retrouve un graphe à bar, permettant de visualiser le nombre de chaque type de tâches utilisé dans le modèle ou le projet qu'on analyse, ces types sont identifiable par leur icône en dessous de chaque bar. On retrouve le même graphe en dessous (bas-droit) mais qui concerne le ou les scénarios à analyser.

Et pour finir, le panel du bas-gauche permet de choisir un ou tous les scénarios d'un ou des modèles de tâche.

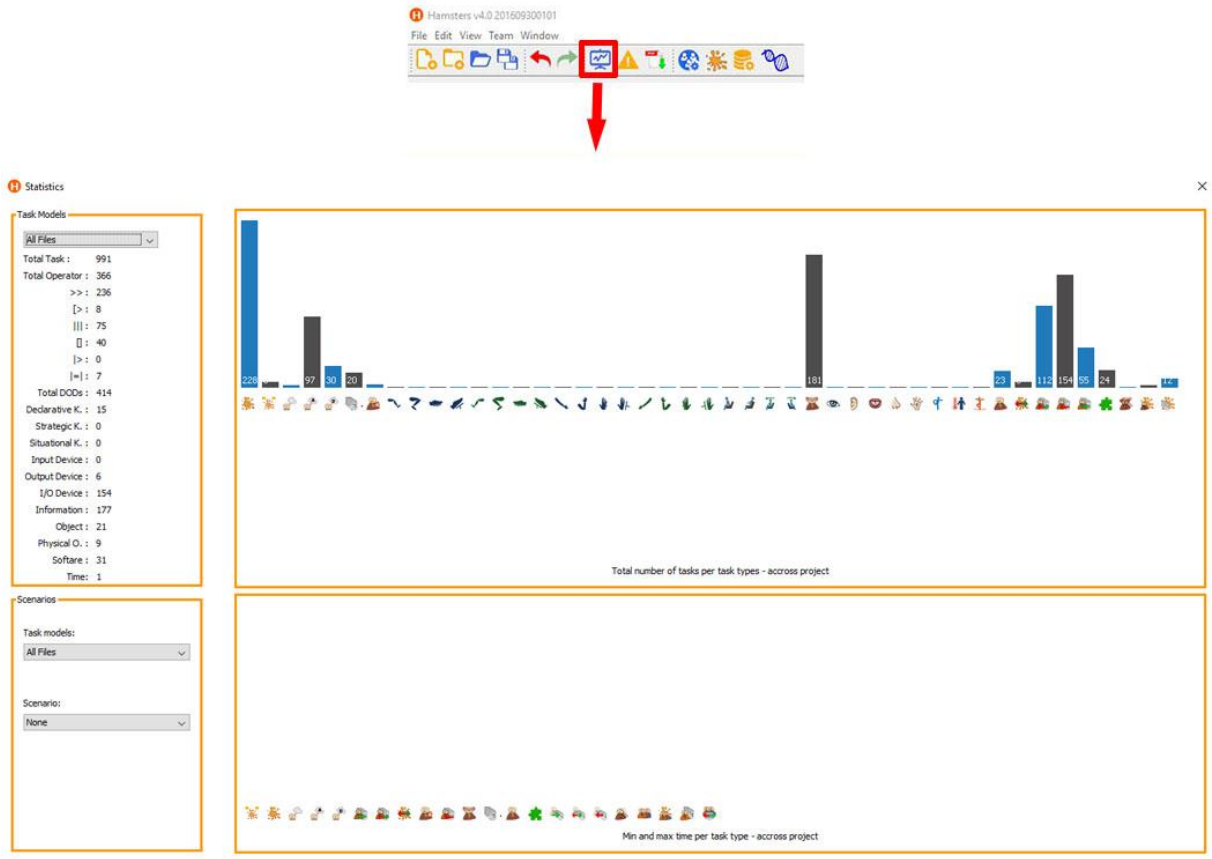


Figure 105. Ouverture du panel Statistique dans l'outil HAMSTERS

3.3.1 Identification de tâche du même type

La notation de tâches HAMSTERS compte plus de 75 types de tâches, afin d'aider à l'analyse des différents types de tâches et voir à quel endroit sont exécutés, nous avons proposé une fonction qui permet de sélectionner toutes les tâches du même type de la tâche sélectionnée. Un simple clic droit sur un type de tâche voulu, et clic sur « select same type task » (voir Figure 106) permet de sélectionner tous les autres tâches du même type et les mettre en évidence (voir Figure 107).

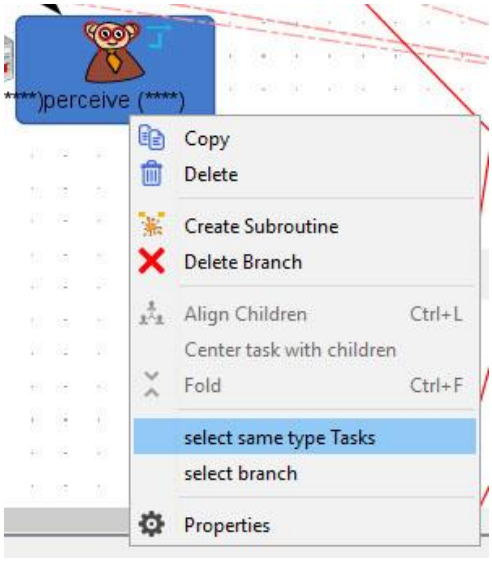


Figure 106. Pop-up menu pour la sélection de tâches du même type

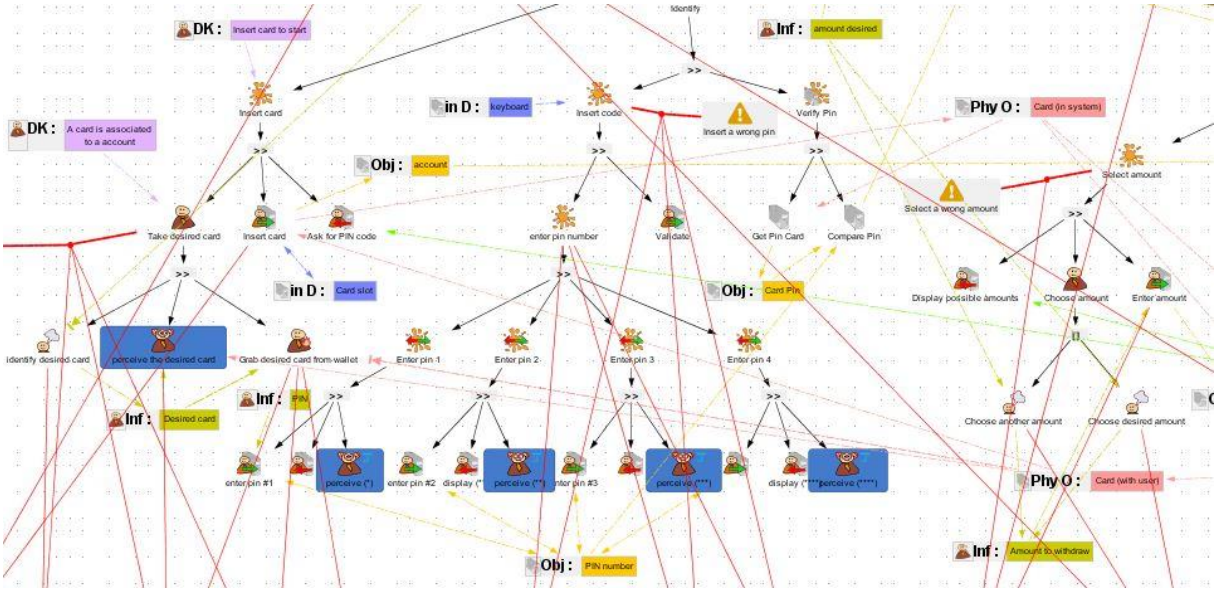


Figure 107. Figure représentant la sélection de toutes les tâches perceptives

4 Exemple de taille réelle « IKKY »

Un des problèmes souvent mis en avant dans les approches à base de modèle concerne la capacité à décrire des informations dans le cadre de systèmes de taille réelle et en tous cas de systèmes allant au-delà des exemples jouets étant décrits dans les articles de recherche.

Cette section présente la mise en œuvre sur une étude de cas industrielle des extensions apportées à HAMSTERS pour décrire les données et structurer les descriptions de tâches. En particulier, elle présente les modèles de tâches produits pour décrire les activités d'un pilote.

Ces activités sont effectuées avec une application conçue dans le cadre du projet IKKY (présenté ci-dessous) et concerne les tâches devant être réalisées par les pilotes des avions

commerciaux (du style Airbus ou Boeing) lors de la préparation de leur appareil avant le départ. Cette tâche appelée « primary cockpit préparation » est complexe, requérant la manipulation de multiples équipements. Le but de cette section est d'illustrer uniquement les extensions décrites dans ce chapitre. Les informations concernant les activités du pilote proviennent du manuel d'opérations de l'avion commercial A350.

4.1.1 Présentation du projet IKKY

Le CORAC est le COnseil pour la Recherche Aéronautique Civile. Il a été créé en juillet 2008 à partir d'engagements pris fin 2007 lors du « Grenelle de l'Environnement ». La mise en place du CORAC s'inscrit dans une volonté de mise en cohérence des efforts de recherche et d'innovation dans le domaine aéronautique, notamment pour la préservation de l'environnement et le développement durable (http://aerorecherchecorac.com/).

Cette étude de cas s'inscrit dans le cadre d'un projet de recherche financé par le CORAC sur le thème des cockpits du futur. Ce projet, intitulé « Modélisation de l'état opérationnel de l'avion et de ses défaillances » a pour but de définir des notations, méthodes et outils pour la conception d'application interactives devant être déployées dans des cockpits pour la gestion contextuelle de l'état de l'avion. Ce projet est mené en partenariat avec Airbus (service R&T Opérations Cockpit) et l'équipe ICS (http://www.irit.fr/recherches/ICS/index.html) appartenant l'Institut de Recherche en Informatique de Toulouse IRIT.

Dans le cadre de ce projet il a été décidé de modéliser les tâches actuelles de l'équipage et de s'en servir de base de comparaison avec la nouvelle interface devant être proposée. Cette comparaison a vocation à aller au-delà de la comparaison quantitative des tâches (nombre de tâches cognitives, nombre de tâches interactives, quantité d'informations à mémoriser, …) mais aussi de prendre en compte la propension de l'interface à déclencher des erreurs chez les opérateurs.

4.1.2 Présentation du FCOM

Les manuels d'utilisation des aéronefs et les manuels d'utilisation des équipages de conduite FCOM (Flight Crew Operating Manual) constituent la référence de l'équipage de conduite principal pour l'utilisation d'un avion dans des conditions normales, anormales et d'urgence. Ces publications comprennent des descriptions de systèmes, des procédures normales et d'urgence, des techniques supplémentaires et des données de performance. Parallèlement à la formation initiale, l'AOM / FCOM constitue la première introduction d'un stagiaire à son nouvel avion. Ceci est normalement suivi d'une formation sur simulateur à base fixe ou complète et, en fin de compte, de l'utilisation de l'aéronef proprement dit. Les manuels d'utilisation doivent répondre aux besoins de la formation initiale, de la formation de transition et des opérations en ligne.

Le manuel d'utilisation de l'équipage de conduite (FCOM) est une documentation d'appui pour l'équipage de conduite. Son but est de :

- Fournir toutes les limitations opérationnelles, procédures, techniques, performances et informations sur le système dont l'équipage de conduite a besoin pour faire fonctionner

un avion de manière sûre et efficace dans des situations normales, anormales et d'urgence.
- Servir directement comme manuel d'utilisation de l'équipage, ou de servir de base aux compagnie aérienne pour élaborer leur propre manuel d'utilisation.
- Servir de guide de référence complet lors de la formation initiale et continue des équipages de conduite.

Le FCOM a 4 section :

- Systèmes avioniques : Cette section comprend une description spécifique de chaque système et de ses interfaces de poste de pilotage associées.
- Procédures : Cette section comprend les procédures normales incluant des SOPs et des procédures complémentaires, les procédures anormales et d'urgence, ainsi que les opérations spéciales.
- Limitations : Cette section présente les limites de l'avion et du système que l'équipage de conduite doit connaître ou consulter lors des opérations.
- Performance : Cette section comprend les performances de l'avion pour chaque phase de vol.

4.1.3 Présentation des procédures d'utilisation standard

Les procédures d'utilisation standard ou SOPs (Standard Operating Procedures) comprennent des inspections, des préparations et des procédures normales. Tous les éléments d'une procédure donnée sont listés dans une séquence qui suit un balayage normalisé des panneaux du poste de pilotage, sauf si cette séquence va à l'encontre de la logique de priorité d'action, ceci afin de s'assurer que l'équipage effectue toutes les actions de la manière la plus efficace.

4.1.4 Présentation de la tâche de préparation préliminaire du cockpit

L'équipage de conduite effectue la préparation préliminaire du poste de pilotage pour s'assurer que toutes les vérifications de sécurité requises sont effectuées.

La tâche de préparation préliminaire du cockpit comprend :

- La mise sous tension de l'avion
- Le démarrage de l'APU (Auxilary Power Unit)
- L'allumage et la vérification de l'ECAM et du carnet de contrôle
- Initialisation et préparation du système d'information embarqué

4.1.5 Edition et analyse du modèle de tâches

L'édition du modèle de tâches « Preliminary Cockpit Preparation » a été réalisé à partir du FCOM de l'airbus A350. La Figure 108 représente la vue éclatée du modèle de tâches de la préparation préliminaire du cockpit. Comme on peut le remarquer, ce modèle est illisible, il faudrait zoomer pour pouvoir l'analyser, et même en zoomant, il sera impossible d'avoir une vue d'ensemble du modèle. Le modèle compte plus de 1700 éléments de la notation, 991 tâches, 366 opérateurs et 414 objets manipulé (voir la Figure 109). L'outil Hamsters propose plusieurs fonctionnalités qui permettent d'éditer un tel modèle de tâche si complexe, en offrant la possibilité de structuration à l'aide de subroutines, composants, copies data et copies tâche.

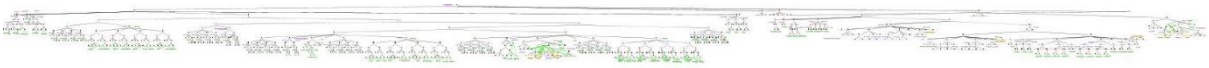


Figure 108. Modèle de tâches "Preliminary Cockpit Preparation"

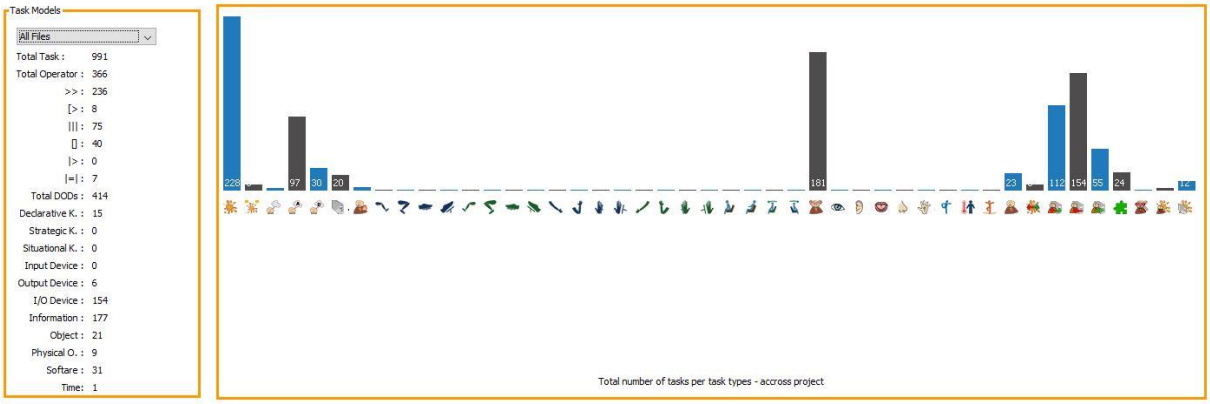


Figure 109. Panneau de statistique du modèle de tâche "Preliminary Cockpit Preparation"

Nous avons commencé à structurer le modèle de tâches à l'aide de subroutine, pour obtenir des sous but au but principal qui est la préparation préliminaire du cockpit. La Figure 110 représente le modèle de tâches décomposé en subroutine, on y retrouve les sous buts présenté dans la section précédente. Pour éxecuter cette tâche, le pilote devra effectuer en séquence : la mise sous tension de l'avion (Aircraft power-up), l'initialisation su système d'information embarqué (OIS initialization), l'allumage et la vérification de l'ECAM et du carnet de contrôle (ECAM/Logbook check), démarrage de l'APU (APU start) et la préparation du système d'information embarqué. Chaque sous but de ce modèle est de type « subroutine », qui sont décrit dans des modèles à part.

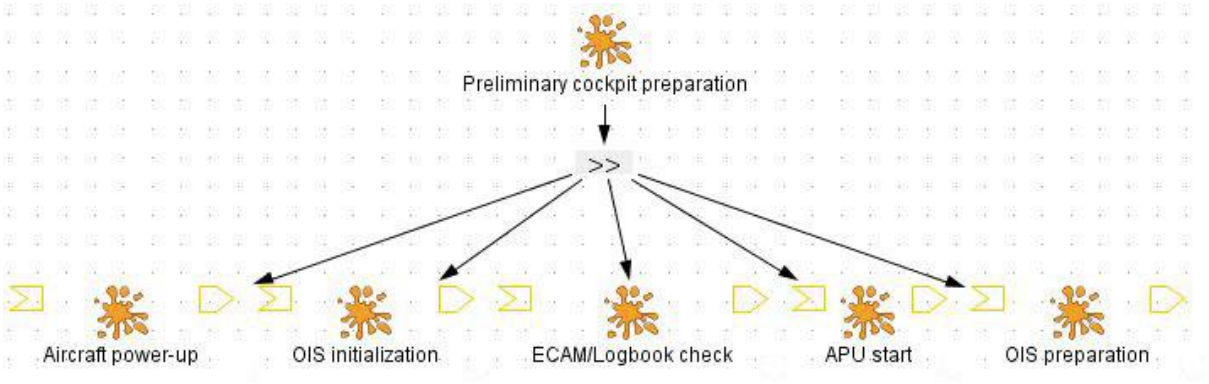


Figure 110. Modèle de tâches "Preliminay cockpit preparation" décomposé à l'aide de subroutine

Pour la suite de cette section, nous allons seulement détailler la subroutine « Aircraft power-up ». La Figure 111 représente un des sous buts de la tâche « Preliminary cockpit preparation », qui est elle-même décomposé en d'autres sous buts sous forme de subroutine. On y retrouve les sous buts suivants : vérification des leviers des moteurs (Check ENG 1, 2 MASTER LEVERS), vérification de la position du sélecteur de démarrage des moteurs (Check ENG START selector), vérification que le levier du train d'atterrissage est en position DOWN (Down LG lever), vérifier ou mettre à Off les deux sélecteurs des essuies glace (Set OFF both WIPER selectors), vérifier la charge des batteries et les actionner (Check BAT 1, BAT EMER 1, BAT

EMER 2, BAT 2 and set on), actionner le bouton de l'alimentation électrique extérieur, mettre les sélecteurs de la référence inertiel en position NAV (All IR MODE selectors), et enfin vérifier ou allumer les lumières du cockpit (Set-Check COCKPIT LIGHT).

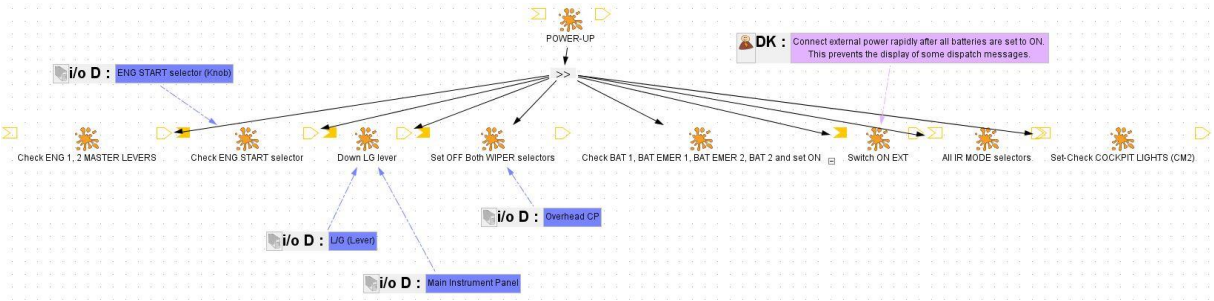


Figure 111. Modèle de tâches "Aircraft power-up"

Dans le but d'illustrer l'utilisation des composants, nous allons seulement détailler la subroutine « Check BAT 1, BAT EMER 1, BAT EMER 2, BAT 2 and set ON » (voir Figure 112) et plus particulièrement les modèles de tâches « Switch ALL BAT [OFF] » et « Switch ALL BAT [ON] », les autres subroutine sont disponible dans les annexes.

La Figure 112 décrit la tâche de vérification de l'état de charge des batteries, deux batteries principales et deux batteries de secours, et les mettre en marche. En séquence, l'utilisateur dois vérifier l'état de charge des batteries puis les charger s'il le faut avant de les mettre en marche. Pour recharger les batteries, il faut les mettre à l'état « OFF », et à l'aide du sélecteur de batterie, on vérifie si toutes les batteries sont à au moins 25 volt. Si au moins une des batteries n'est pas à 25 volt, il faut les charger. Pour charger les batteries, l'utilisateur met les batteries en état « ON », puis sélectionne dans l'ordre de priorité le générateur extérieur (s'il est disponible) ou bien l'APU, pour recharger les batteries. Au bout de 45 minutes, il faut revérifier l'état de charge des batteries, s'ils ne sont pas chargés, on change de dispositif de charge, par exemple si la charge s'effectuait à l'aide du générateur extérieur, cette fois ci on les charge avec l'APU, et vis vers ça, jusqu'à ce que les batteries soient à au moins 25 volt, puis on met en marche les batteries.

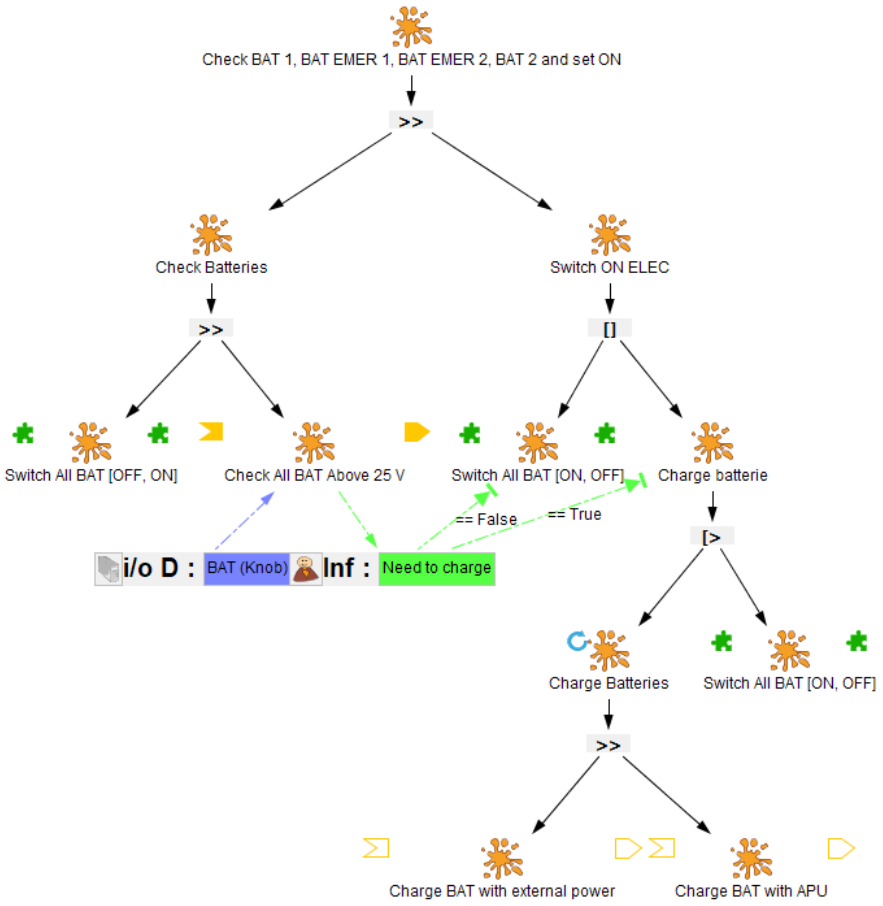


Figure 112. Modèle de tâches de vérification de l'état de charge des batteries

Dans le modèle de tâches de la Figure 112, on remarque que deux sous tâches se répète plusieurs fois, qui sont les actions d'allumer ou d'éteindre les batteries. Les séquence d'actions de ces deux tâches sont presque les mêmes, et sont réutilisé à plusieurs reprises. L'utilisation des composants est utile dans ce cas. Dans la Figure 113 on décrit le template du composant « Switch All BAT » qui est de changer l'état des batteries, vers « ON » s'il étaient à « OFF », ou à « OFF » s'ils étaient à « ON ».

La Figure 113 représente le composant de changement d'état des batteries, et avec deux paramètre en entrée, le paramètre « State » qui représente l'état sur laquelle on veut mettre les batteries, et le paramètre « LightButton » qui décrit l'état de la lumière qui s'affiche sur les boutons qu'on manipule, si les batteries sont dans l'état « OFF » alors les lumières sur les boutons sont allumés, et s'il sont à « ON » alors ils sont éteint. Les DODs en bleu dans le modèle de tâches représentent les 4 périphérique d'entrée/sortie qui sont les boutons des batteries. Le modèles de tâches se décompose en 4 sous branches exécutable en séquence, chaque branche correspond au modèle de tâches qui est de changer l'état de l'un des 4 batteries.

Pour changer d'état de la batterie 1 (voir la branche « Switch <State> BAT 1 pb-sw » de la Figure 113), l'utilisateur dois percevoir le bouton « BAT 1 (pb-sw) », analyser si la lumière « OFF » du bouton est allumée ou non, tout dépend de la valeur passer en paramètre « LightButton », si l'information (DOD en vert) « OFF light » est égal à la valeur du paramètre

« State », alors l'utilisateur dois changer l'état du bouton, en appuyant sur le bouton « BAT 1 (pb-sw) », la lumière du bouton s'allumer ou s'éteint, et l'utilisateur le perçois.

Les 3 autres branches « Switch BAT EMER 1 pb-sw », « Switch BAT EMER 2 pb-sw » et « Switch BAT 2 pb-sw » sont identique à « Switch BAT 1 pb-sw », à part que le bouton manipuler n'est pas le même. Ici on pourra remplacer chaque branche par une instance d'un composant, avec en paramètre le périphérique manipulé.

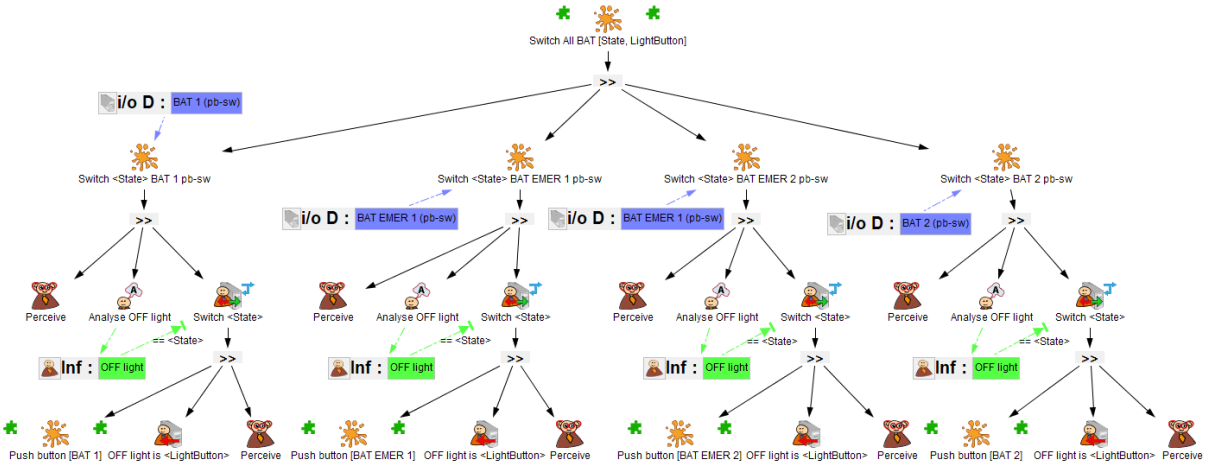


Figure 113. Modèle de tâches du composant "Switch All BAT"

Le composant de la Figure 113 est instancié dans le modèle de tâches de la Figure 112 avec des paramètres différents. La Figure 114 représente une des instances possibles du composant « Switch All BAT », avec comme valeur « OFF » du paramètre « State », et « ON » du paramètre « LightButton ». Cette instance permet de décrire les actions à effectuer pour mettre à « OFF » l'état des batteries, ce qui veut dire que la lumière « OFF » sur les boutons doit être allumé.

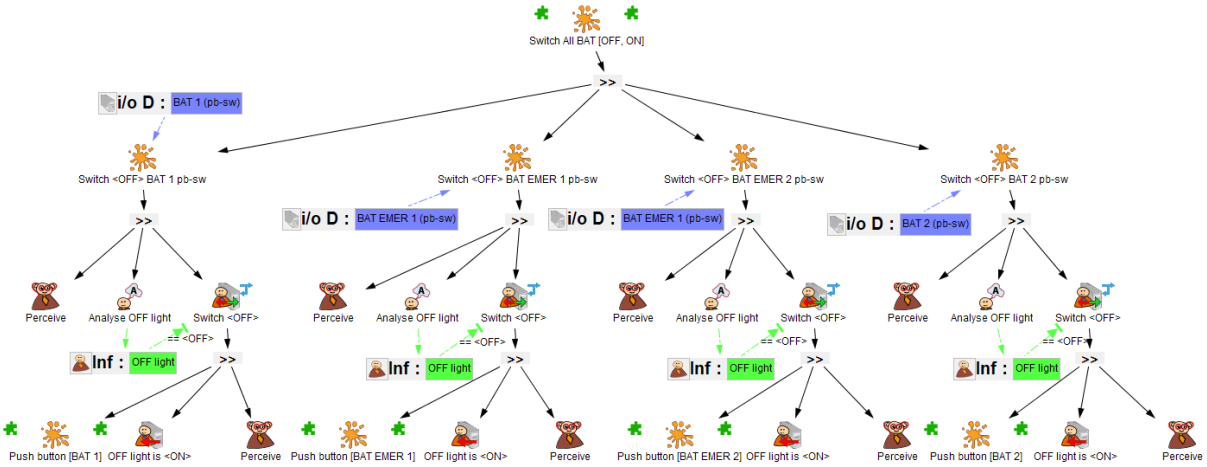


Figure 114. Modèle de tâches de l'instance du composant " Switch All BAT", qui décrit la tâche pour éteindre les batteries

Conclusion

Les modèles de tâches fournissent un support à la conception centrée utilisateur car ils permettent de s'assurer que le système interactif offre les fonctionnalités nécessaires à l'utilisateur au moment où elle/il en a besoin. Les travaux présentés dans cette thèse ont pour but de montrer que les modèles de tâches peuvent aussi fournir un support à la prise en compte pendant la conception d'un système interactif, des erreurs humaines possibles lors de l'utilisation du système. Les principales contributions présentées dans cette thèse sont:

- Les extensions apportées à la notation HAMSTERS qui permettent de décrire les erreurs humaines et leurs causes dans les modèles de tâches, avec en particulier l'élément de notation permettant de décrire l'action erronée que l'on peut observer, appelé phénotype, et l'élément de notation permettant de décrire la cause de l'erreur, appelé génotype (chapitre 4)
- Les extensions apportées à l'outil d'édition et de simulation des modèles de tâches HAMSTERS pour permettre la représentation des erreurs humaines possibles et de leurs causes (chapitre 4)
- Le processus TASSE permettant d'effectuer les étapes suivantes de manières systématiques : identification des erreurs humaines possibles, analyse de leur impact, utilisation de cette analyse et de l'analyse de déviations opérationnelles constatées afin de concevoir une nouvelle version du système interactif en supprimant les éléments de conception pouvant être à la source de ces erreurs (chapitre 5)
- Les extensions apportées à l'outil d'édition et de simulation des modèles de tâches HAMSTERS pour fournir un support au processus TASSE, avec en particulier la fonction d'aide à l'identification systématique des génotypes possibles pour chaque tâche et la fonction d'ajout d'autres types de classification d'erreurs humaines (chapitre 5)
- Les extensions à la notation et à l'outil HAMSTERS qui augmentent le pouvoir d'expression de la notation et sont nécessaires pour décrire les données liées à certaines erreurs dans les modèles de tâches, avec en particulier les éléments de notations permettant de décrire les informations et connaissances requises et manipulées lors de l'exécution de tâches (chapitre 6)
- L'application des extensions de description des données apportées à la notation et à l'outil HAMSTERS sur une étude de cas industrielle dans le domaine d'application de l'aéronautique (chapitre 6)

Les limitations des contributions présentées dans cette thèse sont liées au périmètre de validation du processus TASSE ainsi qu'au support fourni pour la modélisation :

- Le processus TASSE, ainsi que les extensions apportées à la notation et à l'outil ont été partiellement validées avec une étude de cas industrielle d'un domaine d'application

particulier, l'aéronautique. La validation de ces contributions est donc partielle et restreinte à un seul domaine d'application.

- La mise en œuvre du processus TASSE est actuellement possible avec la notation et l'outil HAMSTERS mais pas avec d'autres notations et outils. Des modèles de tâches produits avec d'autres notations et outils ne peuvent pas être utilisés directement. Cependant, les principes des extensions à la notation et à l'outil sont transférables et réutilisables aux autres notations et outils existants.
- Malgré les fonctions ajoutées dans l'outil HAMSTERS pour gérer la quantité d'informations manipulées dans les modèles de tâches, les modèles peuvent devenir compliqués à lire, à vérifier et à valider.

Perspectives

Des perspectives à court et moyen terme peuvent tout d'abord être dégagées à partir des limitations des contributions présentées dans cette thèse :

- Le processus TASSE, ainsi que les extensions apportées à la notation et à l'outil HAMSTERS doivent être appliquées entièrement avec des **études de cas de domaines d'applications divers**, comme par exemple les applications de bureau et le commerce électronique.
- Afin de faciliter l'utilisation du processus TASSE dans le cas où le concepteur possède déjà des modèles de tâches produits avec d'autres outils, des fonctions d**'import de modèles de tâches** pourraient être ajoutées à l'outil HAMSTERS.
- Concernant les difficultés de lecture, vérification et validation des modèles de tâches contenant des descriptions d'erreurs humaines, l'étude de la **rentabilité d'une représentation explicite des erreurs humaines** permettrait de fournir des recommandations sur la pertinence de l'utilisation du processus en fonction des contraintes de conception et développement du système interactif.

De plus, d'autres pistes sont envisageables à court et moyen terme :

La prise en compte, dans le processus TASSE de la **fréquence d'exécution des tâches**. La description de la fréquence d'exécution des tâches dans les modèles permettrait de quantifier des risques d'erreurs et ainsi d'optimiser le traitement de la prise en compte de ces erreurs potentielles dans les phases de conception.

La prise en compte, dans le processus TASSE, des **contraintes temporelles** auxquelles l'utilisateur doit faire face. (Palanque & Basnyat, 2004) a montré que les contraintes temporelles augmentait les risques d'erreurs. L'identification et la représentation dans les modèles de tâches des contraintes temporelles pour l'exécution des tâches et des relations entre ces contraintes temporelles et les risques d'occurrence des erreurs permettrait ainsi d'optimiser le traitement de la prise en compte de ces erreurs potentielles dans les phases de conception.

La prise en compte de l'**environnement de travail de l'utilisateur**. Des travaux préliminaires nous ont permis de montrer qu'il est possible de mettre en relation des modèles en trois dimensions de l'environnement de travail de l'utilisateur avec des représentations des tâches et des données manipulées (Fahssi, Martinie, & Palanque, 2016), ceci dans le but d'analyser l'impact de la disposition des différents objets requis pour l'exécution d'une tâche sur la performance de l'utilisateur. En se basant sur ces travaux préliminaires, il serait intéressant d'étudier comment prendre en compte l'impact de l'environnement physique de l'utilisateur sur les risques d'erreurs.

Ainsi que l'amélioration de l'outil de modélisation de tâches pour la prise en compte de la génération automatique des erreurs humaines, et cela à partir d'un apprentissage de l'outil au court des analyses effectuées. La proposition d'une aide d'identification des erreurs lors de

simulation des modèles de tâches, et aussi l'amélioration de la notation et de l'outil HAMSTERS pour gérer les différents opérations de manipulation des données et objets.

A plus long terme, il serait intéressant d'élargir le périmètre actuel des travaux présentés dans cette thèse. Le processus TASSE concerne les tâches d'un seul utilisateur. Quelles approches et quels outils permettraient de prendre en compte les erreurs humaines lors de la conception de collecticiels ? La notation et l'outil HAMSTERS permettent actuellement de décrire les **activités collaboratives** (MARACCASS, 2015), mais il faudrait étudier si les contributions présentées dans cette thèse sont suffisantes pour identifier et décrire les erreurs humaines de collaboration. Il serait aussi intéressant de fournir les moyens aux concepteurs d'analyser comment la collaboration entre utilisateur peut fournir un support à la résolution d'erreurs (Reason, 1990)

Publications personnelles

Célia Martinie, Philippe Palanque, Racim Fahssi, Jean-Paul Blanquart, Camille Fayollas, Christel Seguin. **Task Model-Based Systematic Analysis of Both System Failures and Human Errors**. IEEE Transactions on Human-Machine Systems vol:462. p:243-254. IEEE.

Racim Fahssi, Célia Martinie, Philippe Palanque. **Embedding explicit representation of cyber-physical elements in task models**. IEEE International Conference on Systems, Man and Cybernetics 2016. p:1969-1974. IEEE Systems, Man, and Cybernetics Society.

Camille Fayollas, Célia Martinie, Philippe Palanque, Racim Fahssi. **Task Models for Supporting Function Allocation between Operators and Autonomous Systems: Application to Collision Avoidance Operations for Spacecraft**. AAAI 2015 Spring Symposium on Intelligent systems for supporting distributed human teamwork 2016. AAAI Press.

Camille Fayollas, Célia Martinie, Philippe Palanque, Racim Fahssi. **Accounting for Organisational faults in Task Model Based Systematic Analysis of System Failures and Human Errors**. IFIP WG 13.5 Workshop on Resilience, Reliability, Safety and Human Error in System Development 2015. p:101-116. University of Bamberg Press.

Racim Fahssi, Célia Martinie, Philippe Palanque. **Enhanced Task Modelling for Systematic Identification and Explicit Representation of Human Errors**. IFIP TC13 Conference on Human-Computer Interaction 2015 (INTERACT). Springer-Verlag.

Philippe Palanque, Racim Fahssi, Célia Martinie, Marco Winckler, Michel Galindo. **Retour d'expérience sur l'enseignement de la modélisation des taches au Master Interaction Homme-Machine de Toulouse**. GT Modèles de Tâches - IHM 2015.

Camille Fayollas, Célia Martinie, David Navarre, Philippe Palanque, Racim Fahssi. **Fault-Tolerant User Interfaces for Critical Systems: Duplication, Redundancy and Diversity as New Dimensions of Distributed User Interfaces**. Workshop on Distributed User Interfaces 2014 (DUI). ACM.

Peter Forbrig, Célia Martinie, Philippe Palanque, Marco Winckler, Racim Fahssi. **Rapid Task-Models Development Using Sub-models, Sub-routines and Generic Components**. Human-Centered Software Engineering 2014 (HCSE). p:144-163. Springer Berlin / Heidelberg.

Célia Martinie, Eric Barboni, David Navarre, Philippe Palanque, Racim Fahssi, Erwann Poupart, Eliane Cubero-Castan. **Multi-Models-Based Engineering of Collaborative Systems: Application to Collision Avoidance Operations for Spacecrafts**. ACM SIGCHI conference Engineering Interactive Computing Systems 2014 (EICS). ACM DL.

Racim Fahssi, Célia Martinie, Philippe Palanque. **HAMSTERS : un environnement d'édition et de simulation de modèles de tâches (Démo)**. Interaction Homme-Machine 2014 (IHM). ACM DL.

Célia Martinie, Philippe Palanque, Martina Ragosta, Racim Fahssi. **Extending Procedural Task Models by Explicit and Systematic Integration of Objects, Knowledge and Information**. European Conference on Cognitive Ergonomics 2013 (ECCE). ECCE '1323. p:1-10. ACM.

Références

André, F., Azzouzi, Y., & Hoarau, R. (2009). *Edition centrée utilisateur de modèle: application au Modèle de tâches.* Toulouse.

Annett, J. (2003). Hierarchical task analysis. *Handbook of cognitive task design, 2*, 17-35.

Annett, J., & Duncan, K. D. (1967). *Task analysis and training design.*

Baber, C., & Stanton, N. A. (1994). Task analysis for error identification: a methodology for designing error-tolerant consumer products. *Ergonomics, 37*(11), 1923-1941.

Barboni, E., Ladry, J.-F., Navarre, D., Palanque, P., & Winckler, M. (2010). Beyond modelling: an integrated environment supporting co-execution of tasks and systems models. *Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems* (pp. 165-174). Berlin, Germany: ACM.

Baron, M., Lucquiaud, V., Autard, D., & Scapin, D. (2006). K-MADe: un environnement pour le noyau du modèle de description de l'activité. *Proceedings of the 18th Conference on l'Interaction Homme-Machine* (pp. 287-288). ACM.

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.

Brown, M., & Leveson, N. G. (1998). odeling controller tasks for safety analysis. *Workshop on Human Error and System Development.*

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*(6), 569-593.

Card, S. K. (2017). *The psychology of human-computer interaction.* CRC Press.

Card, S. K., Moran, T. P., & Newell, A. (1980). The keystroke-level model for user performance time with interactive systems. *Communications of the ACM, 23*(7), 396-410.

Card, S. K., Newell, A., & Moran, T. P. (1983). *The Psychology of Human-Computer Interaction.*

Card, S., Moran, T., & Newell, A. (1986). The model human processor: An engineering model of human performance. *John Wiley & Sons*.

Card, S., Newell, A., & Moran, T. P. (1983). *The psychology of human-computer interaction.* New Jersey, USA: L. Erlbaum Associates Inc.

Diaper, D. (2001). Task analysis for knowledge descriptions (TAKD): a requiem for a method. *Behaviour & Information Technology, 20*(3), 199-212.

Diaper, D., & Stanton, N. (2003). *The handbook of task analysis for human-computer interaction.* CRC Press.

EASA. (2007). *CS 25: Certification for large Aeroplanes.*

Embrey, D. (1986). SHERPA: A systematic human error reduction and prediction approach. *Proceedings of the international topical meeting on advances in human factors in nuclear power systems, 18,* pp. 184-193. Knoxville, USA.

(1986). *Expert report to the IAEA on the Chernobyl accident 61.* Atomic Energy.

Fabio, P., Breedvelt-Schouten, I., & de Koning, N. (1999). Deriving presentations from task models. *Engineering for Human-Computer Interaction* (pp. 319-337). Boston: Springer.

Fahssi, R., Martinie, C., & Palanque, P. (2015). Enhanced task modelling for systematic identification and explicit representation of human errors. *INTERACT* (pp. 192-212). Bamberg: Springer.

Fahssi, R., Martinie, C., & Palanque, P. (2016). Embedding explicit representation of cyber-physical elements in task models. *Systems, Man, and Cybernetics (SMC)* (pp. 001969-001974). IEEE International Conference on. IEEE.

Forbrig, P., Martinie, C., Palanque, P., Winckler, M., & Fahssi, R. (2014). Rapid Task-Models Development Using Sub-models, Sub-routines and Generic Components. *HCSE 2014 Proceedings of the 5th IFIP WG 13.2 International Conference on Human-Centered Software Engineering. 8742*, pp. 144-163. Paderborn, Germany: Springer.

Giese, M., Mistrzyk, Pfau, A., & Von Detten, M. (2008). AMBOSS: a task modeling approach for safety-critical systems. *Engineering Interactive Systems* (pp. 98-109). Berlin: Springer.

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.

Hartson, R. H., Siochi, A. C., & Hix, D. (1990). The UAN: A user-oriented representation for direct manipulation interface designs. *ACM Transactions on Information Systems (TOIS), 8*(3), 181-203.

Hollnagel, E. (1991). The Phenotype of Erroneous Actions: Implications for HCI Design. *Human-ComputerInteraction and Complex Systems*.

Hollnagel, E. (1993). The phenotype of erroneous actions. *International Journal of Man-Machine Studies, 39*(1), 1-32.

Hollnagel, E. (1998). *Cognitive reliability and error analysis method (CREAM).* Elsevier.

Hoppe, H. U. (1987). A grammer-based approach to unifying task-oriented and system-oriented interface descriptions. *Selected papers of the 6th Interdisciplinary Workshop on Informatics and Psychology: Mental Models and Human-Computer Interaction 1* (pp. 353-374). North-Holland: Publishing Co.

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.

Johnson, P., Drake, K., & Wilson, S. (1991). A framework for integrating UIMS and user task models in the design of user interfaces. *User Interface Management and Design* (pp. 203-216). Berlin: Springer.

Johnson, P., Johnson, H., Waddington, R., & Shouls, A. (1988). Task-related knowledge structures: analysis, modelling and application. *BCS HCI*, (pp. 35-62).

Johnson, P., Wilson, S., Markopoulos, P., & Pycock, J. (1993). Adept: Advanced design environment for prototyping with task models. *Proceedings of the INTERACT'93 and CHI'93 Conference on Human Factors in Computing Systems* (p. 56). ACM.

Jourde, F., Laurillau, Y., & Nigay, L. (2010). COMM notation for specifying collaborative and multimodal interactive systems. *Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems* (pp. 125-134). Berlin, Germany: ACM.

Jourde, F., Laurillau, Y., & Nigay, L. (2010). COMM notation for specifying collaborative and multimodal interactive systems. *Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems* (pp. 125-134). ACM.

Kirwan, B. (1992). Human error identification in human reliability assessment. Part 2: detailed comparison of techniques. *Applied ergonomics, 23*(6), 371-381.

Kirwan, B. (1994). *A Guide to Practical Human Reliability Analysis.* Taylor & Francis.

Kirwan, B., & Ainsworth, L. K. (1992). *A guide to task analysis: the task analysis working group.* CRC press.

Lawley, H. G. (1973). Hazard and operability studies. *Chem Eng Process, 8*(5), 105-116.

Lee, G., Howard, J., & Anderson, P. (2002). Safety-critical requirements specification and analysis using spectrm. *Proc. 2nd Meeting of the US Soft. Syst. Safety WG, 260.*

Leplat, J. (1985). *Erreur humaine, fiabilité humaine dans le travail.* Paris: Armand Colin.

Lucquiaud, V. (2005). Proposition d'un noyau et d'une structure pour les modèles de tâches orientés utilisateurs. *Proceedings of the 17th Conference on l'Interaction Homme-Machine* (pp. 83-90). ACM.

Manca, M., Paternò, F., & Santoro, C. (2016). Collaborative Task Modelling on the Web. *Human-Centered and Error-Resilient Systems Development* (pp. 317-334). Springer.

MARACCASS. (2015). *Models and Architectures for the.*

Martinie, C., Barboni, E., Navarre, D., Palanque, P., Fahssi, R., Poupart, E., & Cubero-Castan, E. (2014). Multi-models-based engineering of collaborative systems: application to collision avoidance operations for spacecraft. *Proceedings of the 2014 ACM SIGCHI symposium on Engineering interactive computing systems (EICS)* (pp. 85-94). Rome, Italy: ACM.

Martinie, C., Palanque, P., & Winckler, M. (2011). Structuring and composition mechanisms to address scalability issues in task models. *IFIP TC13 Conference on Human-Computer Interaction (INTERACT 2011). 6948,* pp. 589-609. Lisbon, Portugal: Springer.

Martinie, C., Palanque, P., & Winckler, M. (2011). Structuring and composition mechanisms to address scalability issues in task models. *IFIP Conference on Human-Computer Interaction (INTERACT)* (pp. 589-609). Lisbon, Portugal: Springer.

Martinie, C., Palanque, P., Barboni, E., & Ragosta, M. (2011). Task-model based assessment of automation levels: application to space ground segments. *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on* (pp. 3267-3273). Anchorage, AK, USA: IEEE.

Martinie, C., Palanque, P., Fahssi, R., Blanquart, J.-P., Fayollas, C., & Seguin, C. (2016). Task model-based systematic analysis of both system failures and human errors. *IEEE Transactions on Human Machine Systems, special issue on Systematic Approaches to Human-Machine Interface: Improving Resilience, Robustness, and Stability, to appear, 46*(2), 243-254.

Martinie, C., Palanque, P., Ragosta, M., & Fahssi, R. (2013). Extending procedural task models by systematic explicit integration of objects, knowledge and information. *Proceedings of the 31st European Conference on Cognitive Ergonomics* (p. 23). Toulouse, France: ACM.

McCreary, J., Pollard, J., Stevenson, K., & Wilson, M. B. (1998). *Human factors: Tenerife revisited.*

Mori, G., Paternò, F., & Santoro, C. (2002). CTTE: support for developing and analyzing task models for interactive system design. *IEEE Transactions on software engineering, 28*(8), 797-813.

Norman, D. (2002). *The design of everyday things.* Basic Books.

Norman, D. A. (1981). Categorization of action slips. *Psychological review, 88*(1), 1.

Norman, D. A. (1988). *The psychology of everyday things.* New York: Basic books.

Norman, D., & Draper, S. (1986). *User Centred System Design.* U.S.: L. Erlbaum.

Norman, D., & Drapper, S. (1986). *User Centred System Design.* U.S.: L. Erlbaum.

Palanque, P., & Basnyat, S. (2004). Task patterns for taking into account in an efficient and systematic way both standard and erroneous user behaviours. *Human Error, Safety and System Development (HESSD)* (pp. 109-130). Toulouse, France: Springer.

Parasuraman, R., Sheridan, T. B., & Wickens, C. D. (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.

Paris, C., Tarby, J.-C., & Vander Linden, K. (2001). A flexible methodology and support environment for building task models. *People and Computers XV—Interaction without Frontiers* (pp. 313-329). London: Springer.

Paté-Cornell, E. M. (1993). Learning from the piper alpha accident: A postmortem analysis of technical and organizational factors. *Risk Analysis, 13*(2), 215-232.

Paternò, F. (2004). ConcurTaskTrees: an engineered notation for task models. *The handbook of task analysis for human-computer interaction*, 483-503.

Paternò, F., & Santoro, C. (2001). User Interface Evaluation When User Errors May Have Safety-Critical Effects., *1*, pp. 270-277.

Paternò, F., & Santoro, C. (2002). Preventing user errors by systematic analysis of deviations from the system task model. *International Journal of Human-Computer Studies, 56*(2), 225-245.

Paternò, F., Mancini, C., & Meniconi, S. (1997). ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models. *INTERACT*. Sidney, Autralie: Chapman & Hall.

Pocock, S., Harrison, M., Wright, P., & Johnson, P. (2001). THEA: a technique for human error assessment early in design. *In Human-Computer Interaction: INTERACT*, (pp. 247-254).

Praetorius, G., Lundh, M., & Lützhöft, M. (2011). Learning from the past for pro-activity–A re-analysis of the accident of the MV Herald of Free Enterprise. *Proceedings of the fourth Resilience Engineering Symposium*, (pp. 217-225).

Ragheb, M. (2001). *Three Mile Island Accident.*

Rajan, J., & Redmill, F. (1997). *Human factors in safety-critical systems.* Butterworth Heinemann.

Rasmussen, J. (1983). Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models. *Proceedings of the International Conference on Systems, Man, and Cybernetics* (pp. 257-266). New Delhi, India: IEEE.

Rasmussen, J. (1990). Human error and the problem of causality in analysis of accidents. *Phil. Trans. R. Soc. Lond*, (pp. 449-462).

Reason, J. (1987). Generic error-modelling system (GEMS): A cognitive framework for locating common human error forms. *New technology and human error, 63*, 86.

Reason, J. (1990). *Human error.* Cambridge university press.

Scapin, D. L., & Pierret-Golbreich, C. (1989). Towards a method for task description: MAD. *Work with DisplayUnits*, (pp. 27–34).

Shepherd, A. (2014). *Hierarchial task analysis.* CRC Press.

Siochi, A. C., & Hartson, R. H. (1989). Task-oriented representation of asynchronous user interfaces. *20*(SI), 183-188.

Stanton, N. A., Harris, D., Salmon, P. M., Demagalski, J. M., Marshall, A., Young, M. S., . . . Waldmann, T. (2006). Predicting design induced pilot error using HET (Human Error Template)--A new formal human error identification method for flight decks. *The Aeronautical Journal, 110*(1104), 107-115.

Stuart, J., & Penn, R. (2004). TaskArchitect: taking the work out of task analysis. *Proceedings of the 3rd annual conference on Task models and diagrams* (pp. 145-154). ACM.

Stuart, J., & Penn, R. (2004). TaskArchitect: taking the work out of task analysis. *Proceedings of the 3rd annual conference on Task models and diagrams* (pp. 145-154). ACM.

Swain, A. D. (1989). Comparative evaluation of methods for human reliability analysis. *Gesellschaft fuer Reaktorsicherheit mbH (GRS)*.

Swain, A. D., & Guttmann, H. (1964). Technique for human error rate prediction (THERP). *Symposium on the Quantification of Human Performance*.

Tarby, J.-C., & Barthet, M.-F. (1996). The Diane+ method. *nd International Workshop on Computer-Aided Design of User Interfaces, 96*, pp. 95-119.

The Center for Chemical Process Safety. (1994). Guidelines for preventing human error in process safety. *New York: American Institute of Chemical Engineers.*

U.S. Department of Defence. (1980). *Procedures for Performing a Failure Mode, Effects and Criticality Analysis.* MIL-STD-1629A.

Van Der Veer, G. C., Lenting, B. F., & Bergevoet, B. A. (1996). GTA: Groupware task analysis—Modeling complexity. *Acta psychologica, 91*(3), 297-322.

van Welie, M., Van der Veer, G. C., & Eliëns, A. (1998). Euterpe-Tool support for analyzing cooperative environments. *Proceedings of the Ninth European Conference on Cognitive Ergonomics,* (pp. 24-26).

Vicente, K. J. (1999). *Cognitive work analysis: Toward safe, productive, and healthy computer-based work.* CRC Press.

Villaren, T., Coppin, G., & Leal, A. (2012). Modeling task transitions to help designing for better situation awareness. *Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems.* (pp. 195-204). ACM.

Villaren, T., Coppin, G., & Leal, A. (2012). Modeling task transitions to help designing for better situation awareness. *Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems* (pp. 195-204). ACM.

Villemeur, A. (1988). *Sureté de fonctionnement des systèmes industriels : fiabilité, Facteur humain, informatisation.* Paris: Eyrolles.

Williams, J. C. (1988). A data-based method for assessing and reducing human error to improve operational performance. *IEEE Fourth Conference on. IEEE* (pp. 436-450). Human Factors and Power Plants.

Williams, K. E. (2005). Computer-aided GOMS: A description and evaluation of a tool that integrates existing research for modeling human-computer interaction. *International Journal of Human-Computer Interaction, 18*(1), 39-58.

Abstract

In user-centered approaches, the techniques, methods, and development processes used aim to know and understand the users (analyze their needs, evaluate their ways of using the systems) in order to design and develop usable systems that is in line with their behavior, skills and needs. Among the techniques used to guarantee usability, task modeling makes it possible to describe the objectives and activities of the users. With task models, human factors specialists can analyze and evaluate the effectiveness of interactive applications. This approach of task analysis and modeling has always focused on the explicit representation of the standard behavior of the user. This is because human errors are not part of the users' objectives and are therefore excluded from the job description. This vision of error-free activities, widely followed by the human-machine interaction community, is very different from the Human Factor community vison on user tasks. Since its inception, Human Factor community has been interested in understanding the causes of human error and its impact on performance, but also on major aspects like the reliability of the operation and the reliability of the users and their work. The objective of this thesis is to demonstrate that it is possible to systematically describe, in task models, user errors that may occur during the performance of user tasks. For this demonstration, we propose an approach based on task models associated with a human error description process and supported by a set of tools. This thesis presents the results of the application of the proposed approach to an industrial case study in the application domain of aeronautics.

Keywords: task modeling, human errors, interactive critical systems, certification

Résumé

Dans les approches centrées utilisateur, 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. Parmi les techniques employées pour garantir l'utilisabilité, la modélisation des tâches permet de décrire les objectifs et acticités des utilisateurs. Grâce aux modèles produits, les spécialistes des facteurs humains peuvent analyser et évaluer l'efficacité des applications interactives. Cette approche d'analyse et de modélisation de tâches a toujours mis l'accent sur la représentation explicite du comportement standard de l'utilisateur. Ceci s'explique par le fait que les erreurs humaines ne font pas partie des objectifs des utilisateurs et qu'ils sont donc exclus de la description des tâches. Cette vision sans erreurs, suivie largement par la communauté en Interaction Homme-Machine, est très différente de celle de la communauté en Facteur Humain qui, depuis ses débuts, s'intéresse à comprendre les causes des erreurs humaines et leur impact sur la performance, mais aussi sur des aspects majeurs comme la sureté de fonctionnement et la fiabilité des utilisateurs et de leur travail. L'objectif de cette thèse est de démontrer qu'il est possible de décrire de façon systématique, dans des modèles de tâches, les erreurs pouvant survenir lors de l'accomplissement de tâches utilisateur. Pour cette démonstration, nous proposons une approche à base de modèles de tâches associée à un processus de description des erreurs humaines et supportée par un ensemble d'outils. Cette thèse présente les résultats de l'application de l'approche proposée à une étude de cas industrielle dans le domaine d'application de l'aéronautique.

Mots-clés: modélisation de tâches, erreur humaine, systèmes interactifs critiques, certification

Annexes

1 Annexe 1 : Raffinement des tâches collaboratives

Une tâche de groupe (Group task) doit pouvoir être raffinée en : Tâche de groupe abstraite, humaine, interactive, hybride (voir Tableau 16).

Tableau 16. Types de tâches de groupe de la notation HAMSTERS

Type de tâche	Représentation HAMSTERS
Tâche de groupe abstraite	 Play Game of 15
Tâche de groupe humaine	 Players agree to stop
Tâche de groupe interactive	 Play Game of 15
Tâche de groupe hybride	 Play Game of 15

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

Une **tâche de groupe humaine** (Human group task) permet de décrire une tâche qui doit être accomplie par un ensemble d'individus.

Une **tâche de groupe interactive** (Interactive group task) permet de décrire un passage d'informations entre les utilisateurs et le système.

Une **tâche de groupe hybride** (Hybrid group task) permet de décrire une tâche qui doit être accomplie par un ensemble d'entités.

Une tâche de groupe d'utilisateurs peut être raffinée en une ou plusieurs tâches humaines et de coordination (voir Tableau 17).

Une tâche de coordination est une tâche individuelle distincte, effectuée par un acteur en lien direct avec une autre tâche de coordination effectuée par un autre acteur.

Tableau 17. Types de tâches de coordination de la notation HAMSTERS

Type de tâche	Représentation HAMSTERS
Tâche de coordination humaine	 Asks for permission to start
Tâche de coordination interactive	 Exchange information about current CSM
Tâche de coordination interactive d'entrée	 Asks for permission to start
Tâche de coordination interactive de sortie	 Is asked for permission to start

Une tâche de coordination doit pouvoir être raffinée en :

- Tâche de **coordination humaine** : permet de décrire une action effectuée par un seul acteur dans une tâche de groupe.
- Tâche de **coordination interactive** : permet de décrire un échange d'informations avec un ou plusieurs autres membres du groupe de travail via un système.
- Tâche de **coordination interactive d'entrée** : permet de décrire une réception d'informations via un système.
- Tâche de **coordination interactive de sortie** : permet de décrire un envoi d'informations via un système.

Aspects spatio-temporels

Les aspects spatio-temporels d'une tâche de groupe doivent pouvoir être décrits :

- Aspect temporel (Synchrone, Asynchrone)

Figure 115.Représentation de l'aspect synchrone ou asynchrone d'une tâche de groupe

La Figure 115 représente l'aspect synchrone et asynchrone d'une tâche de groupe :

- Une tâche synchrone est menée simultanément par les différents participants,

- Une tâche asynchrone permet à chaque participant de travailler quand il en a la possibilité.

  - Aspect spatial (Local, Distant)

Figure 116.Représentation de l'aspect local ou distant d'une tâche de groupe

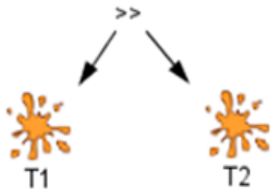
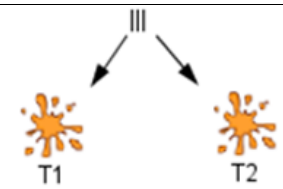
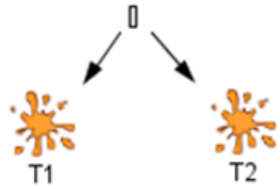
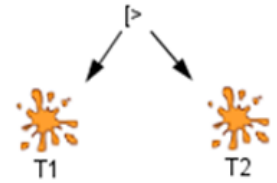
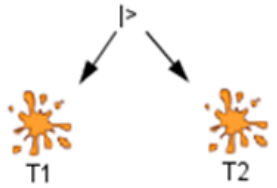
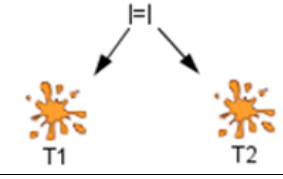
La Figure 116 représente l'aspect local et distant d'une tâche de groupe :

- Une tâche locale est effectuée dans le même lieu (co-localisation)
- Une tâche distante est effectuée à distance (présence virtuelle, télé-présence ou présence à distance)

2 Annexe 2 : Ordonnancement temporel

Le Tableau 18. Opérateurs d'ordonnancement temporel utilisés par la notation HAMSTERS présente l'ensemble des opérateurs permettant de décrire les relations temporelles entre les tâches.

Tableau 18. Opérateurs d'ordonnancement temporel utilisés par la notation HAMSTERS

Type d'opérateur	Symbole	Utilisation
Enable	>>	
Concurrent	|||	
Choise	[]	
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.

3 Annexe 3 : Classification des erreurs humaines

Le Tableau 19 représente les différents types d'erreurs humaines étudiés durant la these. La première colonne "méthode utilisée" décrit le nom de la méthode que les auteurs utilisent pour décrire leurs types d'erreurs humaines, suivie de la colonne des auteurs de la classification ainsi que la date, puis les 4 dernières colonnes décrivent les types d'erreurs humaine.

Tableau 19. Classification des types d'erreurs humaines

Methode utilisée	Auteur(s)	Année	Niveau 4	Niveau 3	Niveau 2	Niveau 1
Human Error Reference Tables	Sandra Basnyat	2004			Double-capture slips	Strong-habit intrusion
						Strong-habit captureI
						Strong-habit exclusion
						Branching errors
						Overshooting a stop rule
						Program Counter Failures Counting
					Omissions following interruptions	Trips
					Reduced Intentionality	Detached Intentions
						Environmental Capture
						I-should-be-doing-something-but-I-can't-remember-what
						Fumbles
						Post-completion error
						Under-motivation
					Perceptual Confusion	Description error
						Input or Misperception errors
					Interference Errors	Data driven error
						Associative-activation error

					Mistimed Checks	Omission
						Repetition
						Reversal
						Insertion
						Replacement
						Premature Action
						Order Errors
						Over-Motivation
The human Error Template (HET)	Marshall & al	2003			External error mode (EEM)	Fail to execute
						Task execution incomplete
						Task executed in the wrong direction
						Wrong task executed
						Task repeated
						Task executed on the wrong interface element
						Task executed too early
						Task executed too late
						Task executed too much
						Task executed too little
						Misread information
						Other
Technique for the Retrospective and Predictive Analysis of Cognitive Errors in Air Traffic Control (TRACEr)	Kirwan	2002			Selection and Quality	Omission
						Action Too much
						Action too little
						Action in wrog direction
						Wrong action on right object
						Right action on wrong object

						Wrong action on wrong object
						Extraneous act
					Timing and Sequence	Action too long
						Action too short
						Action too early
						Action too late
						Action repeated
						Mis-ordering
					Communication	Unclear info transmitted
						Unclear info recorded
						Info not sought/obtained
						Info not transmitted
						Info not recorded
						Incomplete info transmitted
						Incomplete info recorded
						Incorrect info transmitted
						Incorrect info recorded
Situation Awareness (SA)	Endsley	1998			Failure to correctly perceive information	Data not available
						Data hard to discriminate or detect
						Failure to monitor or observe data
						Misperception of data
						Memory loss
					Failure to correctly integrate or comprehend information	Poor mental model
						Use of incorrect mental model
						Over-reliance on default values

						Other
					Failure to project future actions or state of the system	Poor mental mode
						Over-projection of current trends
						Other
					General	Failure to maintain multiple goals
						Executing habitual schema
Error Taxonomy from the Model of Internal Human Malfunction	O'Hare et al.	1994				Error other than human (structural, mechanical, electrical, etc.)
						Information error
						Diagnostic error
						Goal Setting error
						Strategy Selection error
						Procedure error
						Action error
Generic error-modelling system (GEMS)	Reason	1990	Slips & Lapse	Skill based errors	slips/lapse	Perceptual confusion
						Double capture slip
						Omissions following interruptions
						Reduced intentionality
						Interference error
						Over-attention errors
			Mistake	Rule based mistakes	Misapplication of good rules	First exceptions
						Countersigns and non-signs
						Informational overload
						Rule strength

						General rules
						Redundancy
						Rigidity
					Application of bad rules	Encoding deficiencies
						Action deficiencies
				Knowledge based mistakes		Selectivity
						Workspace limitations
						Out of sight out of mind
						Confirmation bias
						Overconfidence
						Biased reviewing
						Illusory correlation
						Halo effects
						Problems with causality
						Problems with complexity
Error taxonomy for identifying improvements in system design, based on cognitive control mechanisms	RASMUSSEN, Jens et VICENTE, Kim J	1989		Effects of learning and adaptation	Skill-based	optimisation of motor skill needs feedback from boundaries of acceptable performance
					Rule-based	the law of least effort may lead to underspecified cues
					Knowledge-based	search for information and test hypotheses in novel situations may lead to acts which are judged as errors after the fact
				Interference among competing control structures	Skill-based	capture by frequently used motor schemata
					Rule-based	functional fixation

						adherence to familiar rules
					Knowledge-based	false analogies
						interference in means-end hierarchy
				Lack of resources	Skill-based	lack of speed
						precision
						force
					Rule-based	inadequate memory for rules.
					Knowledge-based	limitations of linear reasoning in causal networks
						insufficient knowledge
						time
						force
				Stochastic variability	Skill-based	variability of attention
						variability of motor parameters
						motor noise
					Rule-based	erroneous recall of data or parameters related to rules
					Knowledge-based	slips of memory in mental models
The systematic human error reduction and prediction approach (SHERPA)	Embrey	1986			Actions errors	Operation too long/short
						Operation mistimed
						Operation in wrong direction
						Operation too little/much
						Misalign
						Right operation on wrong object

						Wrong opreation on right object
						Operation omitted
						Operation imcomplete
						Wrong operation on wrong object
					Checking errors	Check omitted
						Check imcomplete
						Right check on wrong object
						Wrong check on right object
						Check mistimed
						Wrong check on wrong object
					Retrieval errors	Information not obtained
						Wrong information obtained
						Informaton retrieval incomplete
					Communication errors	Information not communicated
						Wrong information communicated
						Information communication incomplete
					Selection errors	Selection omitted
						Wrong selection made
The causes of causes: Determinants and background variables of human factor incidents and accidents	Gerbert & Kemmler	1986				Vigilance errors
						Information processing errors
						Perception errors
						Sensorimotor/handling errors
	Ramsey	1985		ANATOMICAL	Dimensions of the Body	Body weight and height

Ergonomic components - areas of consideration					Length and circumference of body members
					Reach envelopes
				Strength and Application of Forces	Push, pull and grip strength for body members
					Impact forces from body movement
				Balance and Center of Mass	Loss of control
					Vestibular factors
					Tip-overs
			PHYSIOLOGICAL	Work Physiology/Medical	Fatigue
					Endurance
					Energy expenditure
					Peak loads
					Muscle, tissue, blood flow
				Environments and Environmental Physiology	Climate
					Light
					Radiation
					Noise
					Vibration
			PSYCHOLOGICAL	Sensory Perception	Visual
					Auditory
					Tactual
					Proprioception
					Other sensory inputs
				Perceptual Motor Skills	Motor skill development

						Learning
						Motor control
						Speed and accuracy
					Information Processing	Simultaneous inputs
						Overload
						Feedback
						Attention
						Uncertainty
					Attitude/Behavior	Motivation
						Risk taking
						Activity patterns
						Decision making
						Consumer/user behavior
Analysis and Classification of Human Error	Rouse, W. B., & Rouse, S. H.	1983			Observation of System State	excessive
						misinterpreted
						incorrect
						incomplete
						inappropriate
						lack
					Choice of Hypothesis	inconsistent with observations
						consistent but very unlikely
						consistent but very costly
						functionally irrelevant
					Testing of Hypothesis	incomplete
						false acceptance of wrong hypothesis

						false rejection of correct hypothesis
						lack
					Choice of Goal	incomplete
						incorrect
						unnecessary
						lack
					Choice of Procedure	incomplete
						incorrect
						unnecessary
						lack
					Execution of Procedure	step omitted
						step repeated
						step added
						steps out of sequence
						inappropriate timing
						incorrect discrete position
						incorrect continuous range
						incomplete
						unrelated inappropriate action
A taxonomy for describing human malfunction	RASMUSSEN	1982			Discrimination	Stereotype fixation
						Familiar short-cut
						Stereotype take-over
						Familiar pattern not
						Recognized
					Input information processing	Information not received

						Misinterpretation
						assumption
					Recall	Forget isolated act
						Mistake alternative
						Other slip of memory
					Inference	Condition or side effect not considered
					Physical coordination	Motor variability
						Spatial misorientation