Interactive critical systems

Stagiaire : Marie Perreu

Stage de Licence 3 Informatique
Equipe iCS - IRIT

Automatisation de tests de non-régression pour le logiciel HAMSTERS

Responsables : Célia Martinie et Philippe Palanque

Tutrice : Armelle Bonenfant

Du 4 avril 2016 au 30 juin 2016

Année universitaire : 2015 - 2016

REMERCIEMENTS

Je tiens à remercier tout particulièrement et à témoigner toute ma reconnaissance aux personnes suivantes pour l'expérience enrichissante qu'elles m'ont permis de réaliser.

Monsieur Philippe Palanque, professeur et responsable de l'équipe iCS, pour avoir accepté de m'accueillir au sein de l'équipe iCS et pour m'avoir permis de concrétiser mon souhait de confirmer mon projet d'étude.

Madame Célia Martinie, maître de conférences et enseignante-chercheuse, pour son implication et l'attention particulière qu'elle m'a accordée pour me permettre de réaliser ma mission de stage dans les meilleures conditions.

Monsieur Eric Barboni, ingénieur de recherche, pour m'avoir aidé et s'être rendu disponible sachant m'orienter vers de meilleures solutions pour accomplir ma mission.

Madame Camille Fayollas et Monsieur Racim Fahssi pour m'avoir accueilli dans leur bureau avec Célia Martinie, pour m'avoir aidé en cas de doutes et pour m'avoir permis de m'intégrer au sein de l'équipe grâce à leur extrême gentillesse.

L'ensemble des membres de l'équipe iCS pour leur accueil sympathique et leur coopération professionnelle tout au long de mon stage.

Madame Armelle Bonenfant, enseignante-chercheuse et tutrice, pour sa contribution à la réussite de ce stage.

Je remercie, pour finir, mes proches et amis pour leur soutien et leur aide pour la finalisation de ce rapport.

SOMMAIRE

INTRODUCTION

A. Contexte du stage

Cette première partie est consacrée au **contexte** dans lequel j'ai évolué durant mon stage à commencer par l'**Institut de Recherche en Informatique de Toulouse** (IRIT). Cette entité est l'une des plus imposantes **Unité Mixte de Recherche** [1](UMR) au niveau national et est l'un des piliers de la **recherche** en Languedoc-Roussillon Midi-Pyrénées. Elle a un impact scientifique, elle interagit fortement avec les autres domaines et elle constitue une force structurante du paysage de l'informatique grâce notamment à ses **travaux de pointe**. Le laboratoire se situe au centre de l'**Université Paul Sabatier** de Toulouse et traite de différents thèmes pour lesquels plusieurs équipes travaillent. Comme l'équipe iCS (interactive Critical Systems) qui se consacre à la sûreté de développement du logiciel et qui m'a offert la possibilité d'effectuer un stage de **trois mois** pour clôturer ma formation de Licence 3 en Informatique. Cette dernière prévoit une durée minimum de deux mois de stage, ce rapport présentera donc l'ensemble de cette période mais aussi les perspectives futures pour le troisième mois d'immersion dans le **monde de la recherche**.

B. Justifications de l'intérêt

Pour cette mission professionnelle j'ai fait le choix de confirmer mon **projet futur** qui est de postuler pour le **Master IHM** (Interface Homme-Machine)[2] proposé par l'Université Paul Sabatier. Ce domaine m'attire tout particulièrement car l'objectif est d'étudier la façon dont l'**Homme** interagit avec les ordinateurs pour concevoir des systèmes **efficaces**, **faciles** d'utilisation et **adaptés** aux conditions de travail de l'utilisateur. Il faut s'intéresser au **comportement** de l'humain et lui procurer des techniques d'interactions de plus en plus **naturelles**, comme la reconnaissance de parole, de gestes ou la synthèse vocale. Et pourquoi ne pas découvrir le monde des systèmes interactifs **critiques** où les échanges entre l'interface et l'utilisateur se font à travers une situation représentant un **risque** majeur, au niveau **humain** notamment.

C. Sujet et objectifs du travail

Pour ce stage il m'a été confié la tâche de réaliser **l'automatisation de tests de non-régression** pour le logiciel **Hamsters**. Les tests de non-régression permettent de s'assurer que des défauts n'ont pas été introduits ou découverts dans des parties du logiciel lors de sa modification. Ces tests ne doivent pas faire **régresser** le logiciel, autrement-dit ils ne doivent pas réduire la probabilité de déceler de nouvelles **défaillances** dans l'ancienne version. J'ai donc dû **automatiser la simulation** de différentes

[1] Entité administrative créée par la signature d'un contrat d'association d'un ou plusieurs laboratoires de recherche d'un établissement d'enseignement supérieur ou d'un organisme de recherche.
[2] C'est un ensemble de dispositifs matériels et logiciels permettant à un utilisateur de communiquer avec un système informatique.

fonctionnalités du logiciel pour que lorsque qu'un membre de l'équipe modifie le logiciel, il puisse alors lancer la simulation et vérifier que ce qui marchait avant **fonctionne toujours** après ses modifications.

D. Structuration du rapport

Dans ce rapport je vais tenter de vous montrer en quoi cette l'immersion dans cette équipe représente une expérience **constructive** et **impliquée** dans mon projet d'étude. Ainsi je vais commencer par détailler ma **contribution** au sein de l'équipe, précédée par une ébauche **technique** du logiciel expérimenté pour une meilleure appréhension de ma mission. Cette partie sera suivie d'une **analyse** relative à mon travail et d'un bilan concernant mes **résultats** pour l'équipe de recherche et un bilan personnel. Je terminerai par une conclusion générale sur mon immersion dans le **monde de la recherche** et je donnerai un aperçu sur le travail restant à effectuer pour le dernier mois de stage.

Je joins en fin de ce rapport des **annexes** regroupant des documents plus techniques et concrets, référencés par la suite, qui illustreront différents points relatifs à mon travail durant ces deux premiers mois de stage.

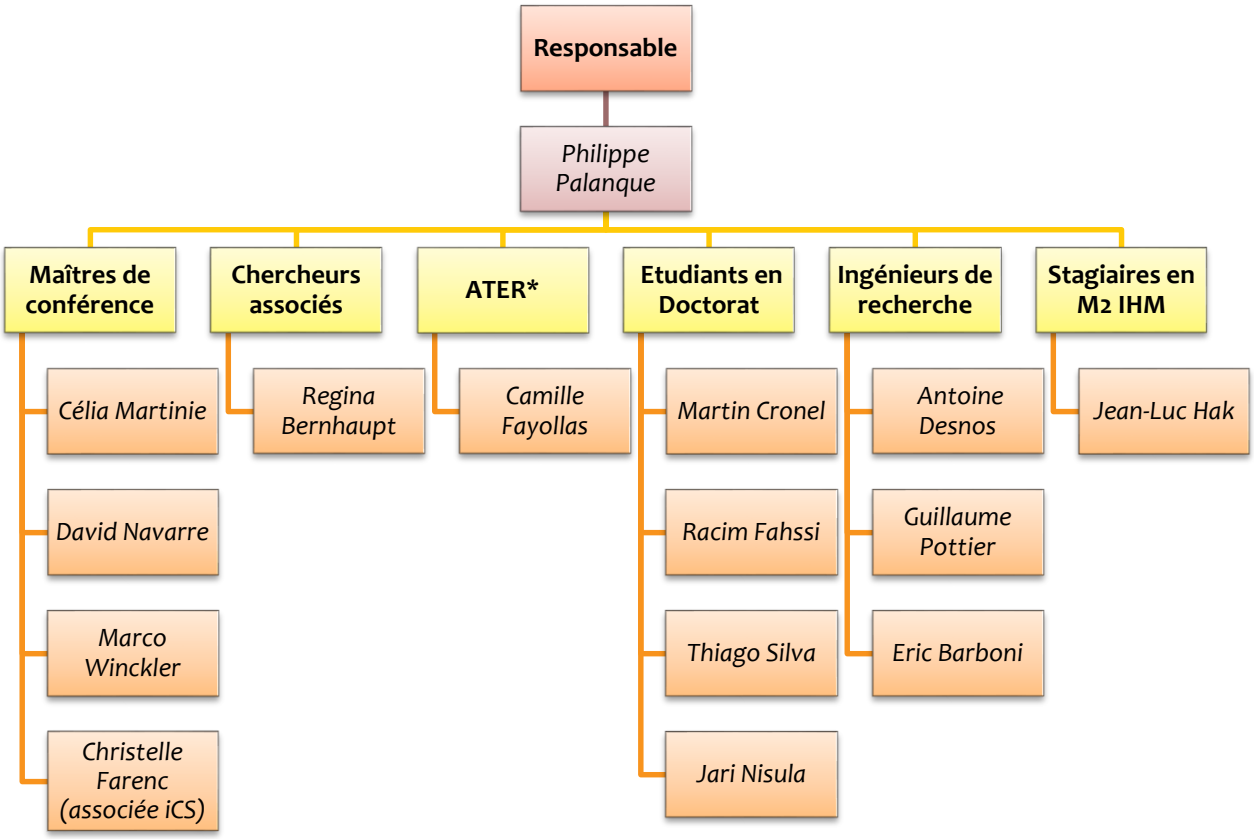
1. Contexte

A. Présentation de l'équipe iCS

L'équipe iCS réalise des recherches centrées sur le domaine des **interactions Homme-Machine**. Ses principales contributions scientifiques traitent des notations, des processus et des outils pour le **design**, **l'implémentation**[3] et **l'évaluation** des systèmes interactifs, si bien qu'elle a une forte influence concernant l'ingénierie des systèmes interactifs. Cette tendance de travail est utilisée dans divers secteurs d'application comme la **gestion du trafic aérien** avec une focalisation sur les applications interactives pour les communications par liaisons de données entre les cockpits et le sol. L'équipe de recherche travaille également pour des sujets comme les **cockpits interactifs**, les secteurs terriens pour les **satellites** (commander et contrôler les systèmes pour différents types de satellites) et la commande et le contrôle des systèmes **UAV** (Unmanned Aerial Vehicle)[4], plus communément appelés drones. Au-delà de ces bonnes connaissances dans la **sécurité** des systèmes critiques, l'équipe iCS s'adresse également à des domaines pour lesquels les coûts potentiels d'une faille sont plus élevés par rapport au coût de développement. Cela inclut des domaines d'application tel que les **applications web** et le marché de masse des systèmes de **divertissement** à domicile et les **jeux vidéo** sur ordinateur.

Voici un tableau présentant les membres de l'équipe iCS et leur fonction :

[3] Mise en œuvre d'un produit fini à partir de documents de conception, de spécification et/ou d'un cahier des charges.
[4] Ce sont ce que l'on appelle les drones, qui sont en fait des avions qui fonctionnent sans l'action d'un pilote à bord. Ces avions sont contrôlés par diverses autonomies comme, par exemple, des ordinateurs de bord. Ils sont utilisés notamment lors d'actions dangereuses ou risquées, c'est pourquoi on les retrouve le plus souvent dans des opérations spéciales et militaires.

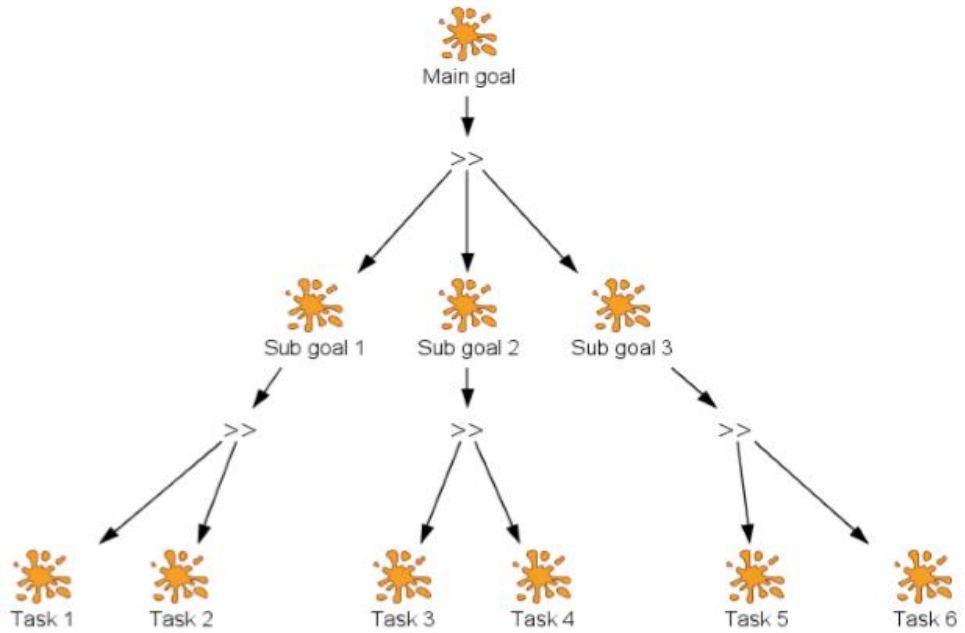


*Attaché temporaire d'enseignements et de recherche

B. Présentation technique du logiciel Hamsters

Cette partie va vous permettre de savoir en quoi consiste le logiciel sur lequel j'ai travaillé durant mon stage. Qui l'a conçu ? Quand ? Pourquoi ? Quels sont les principaux concepts utilisables ? A quoi servent-ils ? … Je vais expliquer des **notions** essentielles que j'illustrerai par un exemple concret d'utilisation : le distributeur automatique de billets. Ainsi cela vous aidera à mieux comprendre et appréhender les résultats que j'ai obtenus au bout de ces deux mois.

Le logiciel Hamsters est un outil conçu en **2009** par des étudiants en deuxième année de Master IHM pour la **modélisation** d'activités humaines et **l'étude** et la **conception** d'interfaces homme-machine. Il permet d'éditer des modèles de **tâches** et de produire des **scénarios** d'utilisation associés à ces modèles grâce à un **simulateur**. Ce logiciel utilise des mécanismes de notations de modélisations de tâches existantes, à travers des concepts de **décomposition hiérarchique**, de buts en sous-buts avec des raffinements au niveau du type des tâches.



1. La notion de tâche

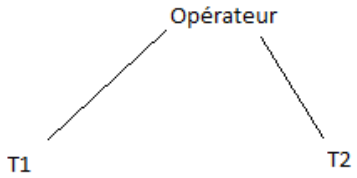
Le principe de Hamsters est la modélisation et la simulation de **tâches utilisateur**. La notion de tâche est **générique**, elle peut être un but ou une action utilisateur qui peut elle-même se décomposer en un sous-ensemble de tâches. Il y a différents types de tâches génériques comme les **tâches abstraites** (regroupant plusieurs sous-tâches), les **tâches système** (fonction exécutée par le système), les **tâches utilisateur** (activité de l'utilisateur ou rassemblant un ensemble d'activités utilisateur) et les **tâches interactives** (passage d'information entre l'utilisateur et le système ou rassemblant un ensemble de tâches interactives).

Le **raffinement des tâches interactives** permet de caractériser la manière dont le flux d'information transite entre l'utilisateur et le système (entrée, sortie ou entrée et sortie).

Le **raffinement des tâches utilisateur** permet de prendre en compte les différents aspects d'une activité menée par un utilisateur. Ces tâches sont de natures différentes selon la manière dont l'utilisateur traite l'information. Une **tâche motrice** est un mouvement physique de l'utilisateur (appuyer sur un bouton). Une **tâche perceptive** est une action sensorielle de récupération d'informations extérieures (percevoir une demande de code : l'utilisateur doit voir qu'il doit entrer son code). Une **tâche cognitive** est un traitement de l'information contextuelle (se rappeler du code : l'utilisateur doit se souvenir de son code pour le fournir au système).

2. La notion d'opérateur

Les opérateurs décrivent **les relations temporelles qualitatives** entre les tâches. Avec l'opérateur "**Enable**" les tâches T1 et T2 se déroulent en séquence, avec l'opérateur "**Concurrent**" les tâches se déroulent simultanément, avec l'opérateur "**Choice**" l'utilisateur effectuera l'une des deux (désactivation de la tâche non choisie), avec l'opérateur "**Disable**" le démarrage de T1 entraîne l'arrêt définitif de T2, avec l'opérateur "**Suspend-resume**" le démarrage de T2 entraîne l'arrêt provisoire de T1 et avec l'opérateur "**Order independent**" l'utilisateur peut choisir d'effectuer la tâche 1 ou la tâche 2 en premier.

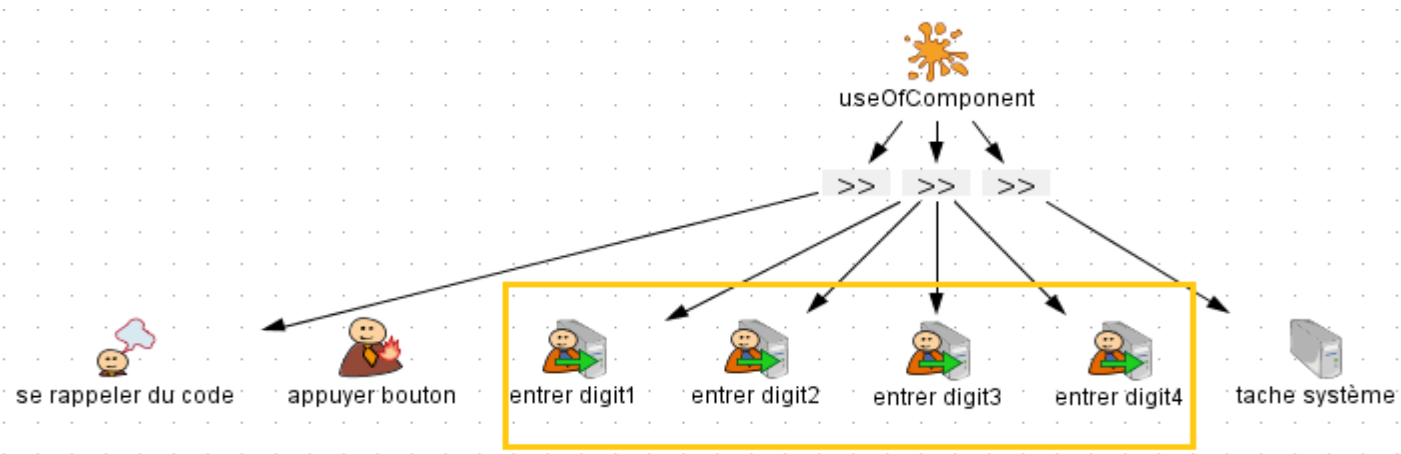


Opérateur	Symbole
Enable	>>
Concurrent	|||
Choice	[]
Disable	[>
Suspend-resume	|>
Order independant	|=|

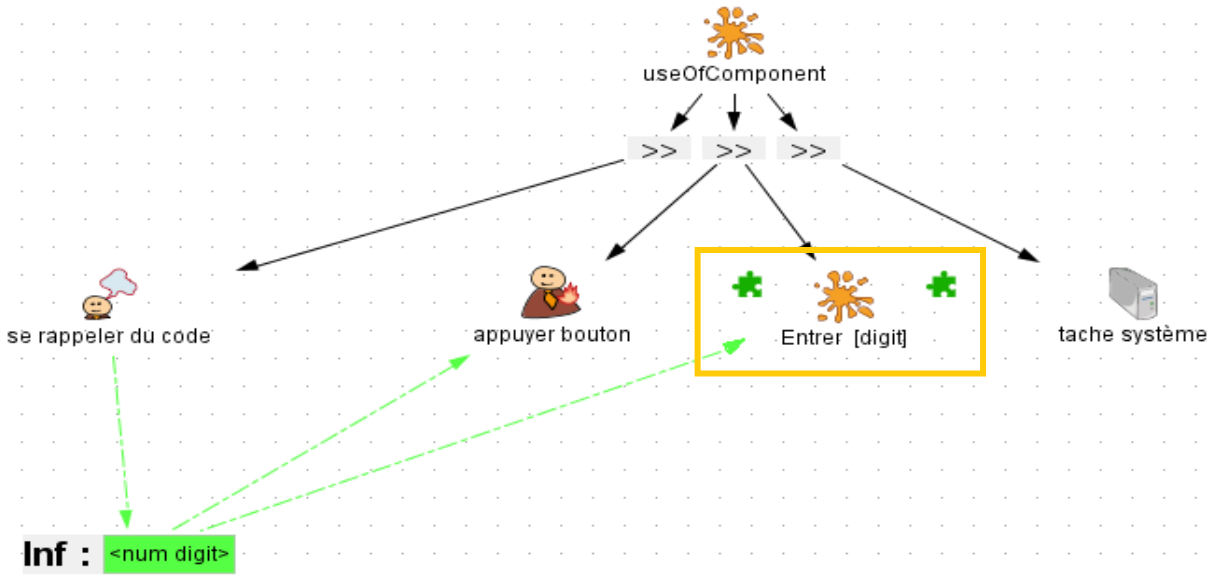
3. La notion de composant

Les composants permettent de **réutiliser** des actions de modélisation. Ils fournissent des paramètres en entrée pour améliorer la réutilisation des modèles de tâches. Ainsi, quand on rencontre un processus qui va se **répéter** plusieurs fois, on va pouvoir en faire un composant qui prendra en paramètre seulement les informations qui différent selon l'opération. Par exemple, lors de l'utilisation d'un distributeur automatique de billets, l'utilisateur va être amené à entrer les quatre chiffres qui composent son code de carte de crédit. Il va donc reproduire l'action d'entrer un chiffre quatre fois et donc effectuer par exemple **la tâche cognitive** consistant à se rappeler son code, puis il va faire la **tâche interactive** (de l'utilisateur vers le système) pour appuyer sur la touche du premier chiffre et enfin **une tâche système** va se produire, etc. … La seule information qui va différer sera le **numéro d'ordre** du chiffre à entrer (le premier, le second, le troisième ou le quatrième). On va donc pouvoir créer un **composant générique** qui va effectuer ce processus pour un paramètre donné, soit le numéro d'ordre du chiffre à entrer.

↳ Sans le composant :



↳ Avec le composant :



4. La notion de subroutine

Les **subroutines** sont utilisées dans le but de **structurer** les modèles de tâches et de définir les informations qui sont passées entre ces modèles. Ce **mécanisme** a pour but de réutiliser un **sous arbre** dans un modèle de tâches. En effet, un groupe de tâches - représenté comme un arbre - pourrait être utilisé plusieurs fois avec à chaque fois des petites différences. C'est pourquoi, la subroutine permet de décrire des comportements **récurrents** et d'en dégager les **paramètres** entrants et/ou sortants et d'expliciter comment ils influencent le comportement du modèle de tâches. Ainsi elle est un **groupe d'activités** que les utilisateurs exécutent plusieurs fois, peut-être dans différents contextes, possédant des icônes colorées ou non en fonction des paramètres qui leurs sont donnés. Il est difficile de faire la différence entre une **subroutine** et un **composant**, néanmoins une subroutine **explicite** les paramètres en entrée **et** en sortie alors qu'un composant indique un paramètre en entrée.

Aucun paramètre

Un ou plusieurs paramètres en entrée

Un ou plusieurs paramètres en sortie

Un ou plusieurs paramètres en entrée et en sortie

2. Présentation détaillée du sujet et ses objectifs

Maintenant que vous connaissez bien mon contexte de stage et le logiciel sur lequel j'ai pu réaliser mon apprentissage, nous allons pouvoir passer à une présentation détaillée de mon **rôle** au sein de l'équipe iCS.

Comme indiqué plus haut, l'objectif principal de ma mission était de réaliser l'**automatisation de tests de non-régression** pour un des logiciels développés par des membres de l'équipe de recherche. Ce logiciel est utilisé autant au niveau de la **recherche** qu'au niveau de l'**enseignement**. En effet, sur le plan de la recherche **Hamsters** est utilisé dans des projets réalisés avec le **CNES**[5] (Centre National d'Etudes Spatiales), pour les satellites dans le but d'analyser et de décrire des tâches. L'iCS travaille également avec **Airbus**[6] de manière générale pour comprendre les tâches des pilotes, analyser les différentes situations **critiques** qui peuvent subvenir et réfléchir à des issues. Cette analyse permet de savoir combien de tâches sont à effectuer, combien d'informations l'utilisateur doit garder en tête et à quel moment, combien cela va coûter, déceler les **erreurs humaines**, etc.

Pour ce qui est de l'**enseignement**, ce logiciel est fréquemment utilisé pour **l'analyse et la description de tâches** dans le secteur de **l'IHM**. L'objectif majeur est de dégager l'**utilisabilité** du logiciel, dont un critère qui est **l'efficacité**. Le principe est de prendre le cas d'un utilisateur (un pilote par exemple), avec un objectif à réaliser, pour lequel on va décrire les différentes tâches à exécuter pour effectuer au mieux cette action.

En somme, le logiciel Hamsters est largement **répandu** et utilisé dans différents contextes, c'est pourquoi l'équipe iCS se doit de faire **évoluer** leur produit et de le rendre le plus **robuste** possible pour finalement en **éviter la régression**. C'est à ce moment que mon **intervention** dans l'équipe se concrétise. En effet, par manque de temps et de moyens l'iCS ne peut pas améliorer le logiciel comme elle l'aurait souhaité. Mon rôle a donc été dans un premier temps de porter un **œil neuf** sur l'utilisation de Hamsters et d'en tester manuellement les différentes **fonctionnalités**. Durant cette **prise en main** j'ai pu déceler quelques **incohérences** et donc proposer par la suite des **suggestions** d'amélioration à

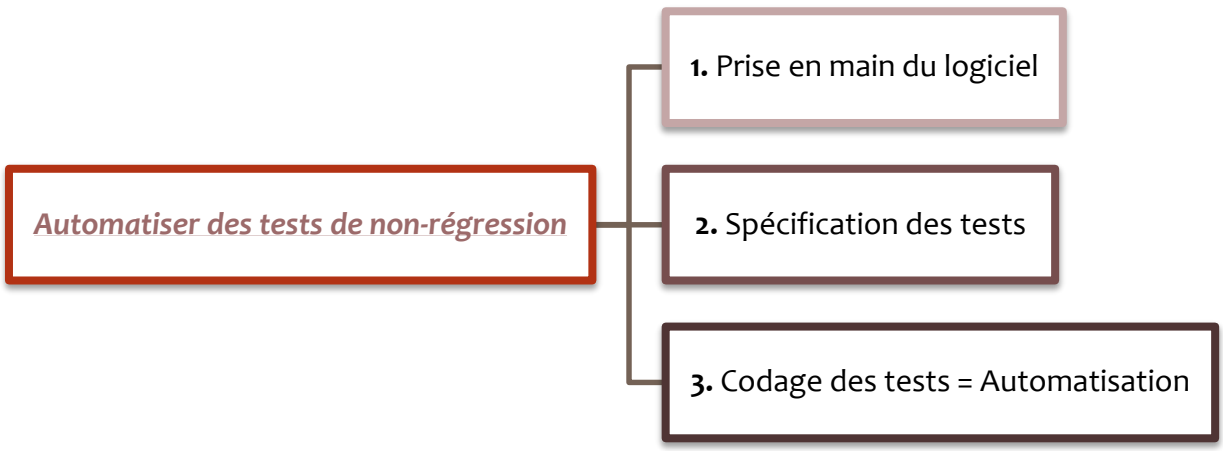
[5] Organisme officiel de la recherche spatiale en France.
[6] Constructeur aéronautique européen situé à Blagnac, dans la banlieue de Toulouse en France (siège social).

l'équipe. Ensuite, j'ai créé plusieurs documents de **spécification** de tests et de **suivi** pour avoir un **support** de développement et une **traçabilité** de mon travail. Enfin, j'ai pu passer à l'étape concrète de mon stage : la **programmation des tests d'interface**. J'ai réalisé cette tâche en totale **autonomie** pour apprendre les concepts de tests d'interface à l'aide de la bibliothèque **JellyTools** de NetBeans[7]. En effet, l'équipe n'ayant pas les connaissances concernant le développement avec cet **outil**, j'ai dû réaliser des recherches approfondies pour en acquérir une **maitrise totale**. C'est pourquoi l'équipe tenait à ce que, lors de cet apprentissage, je conserve la **documentation** qui m'a été utile pour leur fournir un document plus complet et pour leur présenter en fin de stage les différents **concepts** essentiels à la programmation des **tests d'interface**.

3. Méthode de travail

A. Démarche méthodologique détaillée

Ma mission a était découpé en plusieurs sous objectifs :



La **prise en main** du logiciel est une étape nécessaire car sans la connaissance et la maîtrise des fonctionnalités je ne peux pas entreprendre le codage de tests d'interface. C'est pourquoi, dans un premier temps, j'ai utilisé la **documentation** qui était mise à ma disposition pour comprendre l'utilité et le fonctionnement de l'outil Hamsters. De plus, il m'a était fourni un projet existant et fonctionnel sur lequel j'ai pu tester les différentes **fonctionnalités** apprises. Lors de cet apprentissage j'ai également créé un projet personnel (cf. annexe n°1) pour voir comment construire un projet en partant de zéro de manière à **maîtriser** l'outil et en déduire une spécification de tests des plus **cohérentes**.

La **spécification des tests** est aussi une étape nécessaire car c'est durant celle-ci que j'ai posé clairement les fonctionnalités à vérifier et quels sont leurs résultats. Pour cette étape j'ai eu à tenir à

[7] Environnement de développement intégré, placé en open source. En plus de Java, NetBeans permet également de supporter différents autres langages, comme C, C++, JavaScript, XML, Groovy, PHP et HTML.

jour quatre documents (cf. annexes n°2, 3, 4 et 5). Un document de **spécification abstraite** du logiciel pour y recenser toutes les actions possibles, comment les réaliser et le résultat qu'elles produisent. Un autre document de **spécification concrète** de tests au niveau du projet ATM[8], à travers lequel j'ai annoté les actions concrètes et les résultats attendus en retour. Ce document est en fait le support que j'ai utilisé pour la programmation des tests. Un autre document est consacré aux **problèmes** rencontrés durant l'utilisation de Hamsters et aux **suggestions** qui pouvaient être faites pour améliorer cet outil. Enfin, le dernier document contient un **tableau de versionnage** me permettant de recenser toutes les versions de tous les documents avec la date et l'auteur des **modifications**. En effet, les documents de spécification ont été mis à jour tout au long de ma mission. Ces trois documents ont été réalisés dans l'optique de pouvoir programmer les tests les plus adaptés possibles pour mettre en avant les **disfonctionnements** et les possibilités **d'évolution** du logiciel. Pour l'équipe iCS, ce travail est utile dans le sens où ils n'ont pas le temps de tester régulièrement le logiciel Hamsters et donc ils ne peuvent pas l'améliorer. Mon rôle a donc été de porter un **œil neuf** sur l'outil de manière à faire ressortir les **incohérences** et donc les **solutions** à apporter à l'ensemble de l'équipe.

Après ces deux étapes primordiales, j'ai pu me lancer dans le **codage des tests.** La programmation de ceux-ci a été effectuée à l'aide de la bibliothèque **JellyTools** de NetBeans qui permet l'automatisation de **tests d'interface**. Ainsi, après m'être familiarisée avec toutes les possibilités qu'offre JellyTools, avoir eu une bonne maîtrise et avoir fait plusieurs tests, **l'automatisation** a pu commencer.

B. Présentation de l'organisation, de la planification et de l'environnement de travail

Cette partie consiste à vous faire part des conditions dans lesquelles mon travail s'est déroulé.

Tout d'abord, au niveau de **l'organisation** du travail et de son suivi il a fallu trouver une manière de **communiquer** et de **partager** les documents et autres fichiers nécessaires pour mon avancement. Tout d'abord, pour ce qui est du partage de documents, de projets, de fichiers d'installation, de documentation et autres nous avons utilisé Drop Box[9], le Cloud[10] de l'IRIT et la messagerie étudiante. Ensuite, pour ce qui est de la communication, cela n'a pas été difficile car j'ai eu la chance de partager le bureau de Célia Martinie, Camille Fayollas et Racim Fahssi. Ainsi, nous avons échangé de manière **orale** en cas de demande de précisions, d'interrogations, de difficultés, etc.

Ensuite, au niveau de la **planification** je n'ai pas reçu d'exigence particulière et j'ai pu avancer à mon rythme. Pour vous donner un ordre d'idée j'ai passé deux semaines sur la **prise en main** du logiciel et la **spécification des tests**. Célia Martinie m'a beaucoup guidé durant cette période, surtout lors de la

[8] Projet existant et fonctionnel créé au sein de l'équipe ICS sur lequel j'ai pu m'exercer lors de la période de prise en main du logiciel Hamsters. Ce projet ATM consiste en la modélisation d'un retrait d'argent à un distributeur automatique de billets (de l'identification de l'utilisateur au retrait de l'argent par l'utilisateur).

[9] Service de stockage et de partage de copies de fichiers locaux en ligne.

[10] Exploitation de la puissance de stockage de serveurs informatiques distants par l'intermédiaire d'internet (le plus souvent). C'est en fait l'accès via un réseau de télécommunication à des ressources partagées.

précision de certains concepts vus plus haut dans ce rapport, comme l'utilisation des composants et des subroutines. Elle m'a aussi donné des **conseils** concernant la rédaction de la spécification ce qui m'a permis d'améliorer la présentation mais surtout la **précision** de mes documents. Après et pendant ces deux semaines nous avons fait quelques points d'avancement et des relectures pour me permettre, par la suite, de me lancer rigoureusement dans l'étape de **codage des tests** de non-régression. Cette dernière a duré jusqu'à la fin de mon stage entrecoupée par des **points d'avancement** et, comme dans tout projet, des **imprévus** qu'il a fallu traiter pour finaliser ma mission dans les temps.

Enfin, au niveau de **l'environnement de travail** j'ai disposé d'un bureau, d'un ordinateur et d'un écran supplémentaire pour plus d'efficacité et de confort. Comme dit précédemment, j'ai partagé mon bureau avec Célia Martinie, Camille Fayollas et Racim Fahssi, ce qui a également contribué à mon **intégration** au sein de l'équipe de recherche. Je considère donc que cet environnement de travail a été des plus favorables et je peux affirmer qu'il m'a permis de **me sentir bien** tout au long de ces deux mois et de **m'impliquer** rigoureusement dans ma mission. En effet, la **proximité** avec les membres de l'équipe m'a aidé à faire face aux situations compliquées et m'a permis de voir que ma contribution avait un intérêt certain pour l'équipe iCS.

4. Travail réalisé

A. Réalisation

a) Prise en main de l'existant

Après l'étape de spécification j'ai pu me lancer dans l'automatisation des tests. Avant cela, il fallait que je prenne en main le **code source** de l'ensemble de l'équipe dans l'environnement **NetBeans** à l'aide d'un outil de **gestion d'automatisation** de production de projets logiciel : **Maven**. Cet outil utilise un **POM** (Project Object Model) afin de décrire un projet logiciel, ses **dépendances** avec des **modules** externes et l'ordre à suivre pour sa production. En plus de cet outil j'ai dû me familiariser avec un **gestionnaire de versions** distribué, nommé **Mercurial**. C'est ce dernier qui permet à chaque membre de l'équipe de **partager** en temps voulu les **modifications** qu'il a effectué pour les soumettre aux autres membres qui peuvent alors **synchroniser** leur code source **local** avec ceux des autres. C'est pourquoi il était nécessaire que je comprenne bien le fonctionnement de ces outils afin de pouvoir **soumettre mon avancement** à l'ensemble de l'iCS et que chacun puisse **valider ou non** les tests que j'avais programmés.

b) Analyse et apprentissage de l'outil JellyTools et des concepts de tests d'interface

Comme je l'ai précisé plus haut dans ce rapport, j'ai dû apprendre en **autonomie** l'ensemble des concepts de tests d'interface avec l'outil **JellyTools**. J'ai effectué beaucoup de recherches sur Internet pour en savoir un peu plus sur cette **bibliothèque** permettant de rendre automatique différentes

actions, telles qu'un clic souris, à travers du code en langage **Java**[11]. J'ai dû acquérir une maitrise de cet outil pour pouvoir coder les différents **scénarios** que j'avais décrits dans mon support de **spécification** de tests. Ce fut une étape relativement complexe dû au manque de précision de la documentation sur Internet et dû à la multitude de possibilités qu'offre **JellyTools**, ce qui était un avantage mais qui exigeait une compréhension parfaite de chacune d'entre elles.

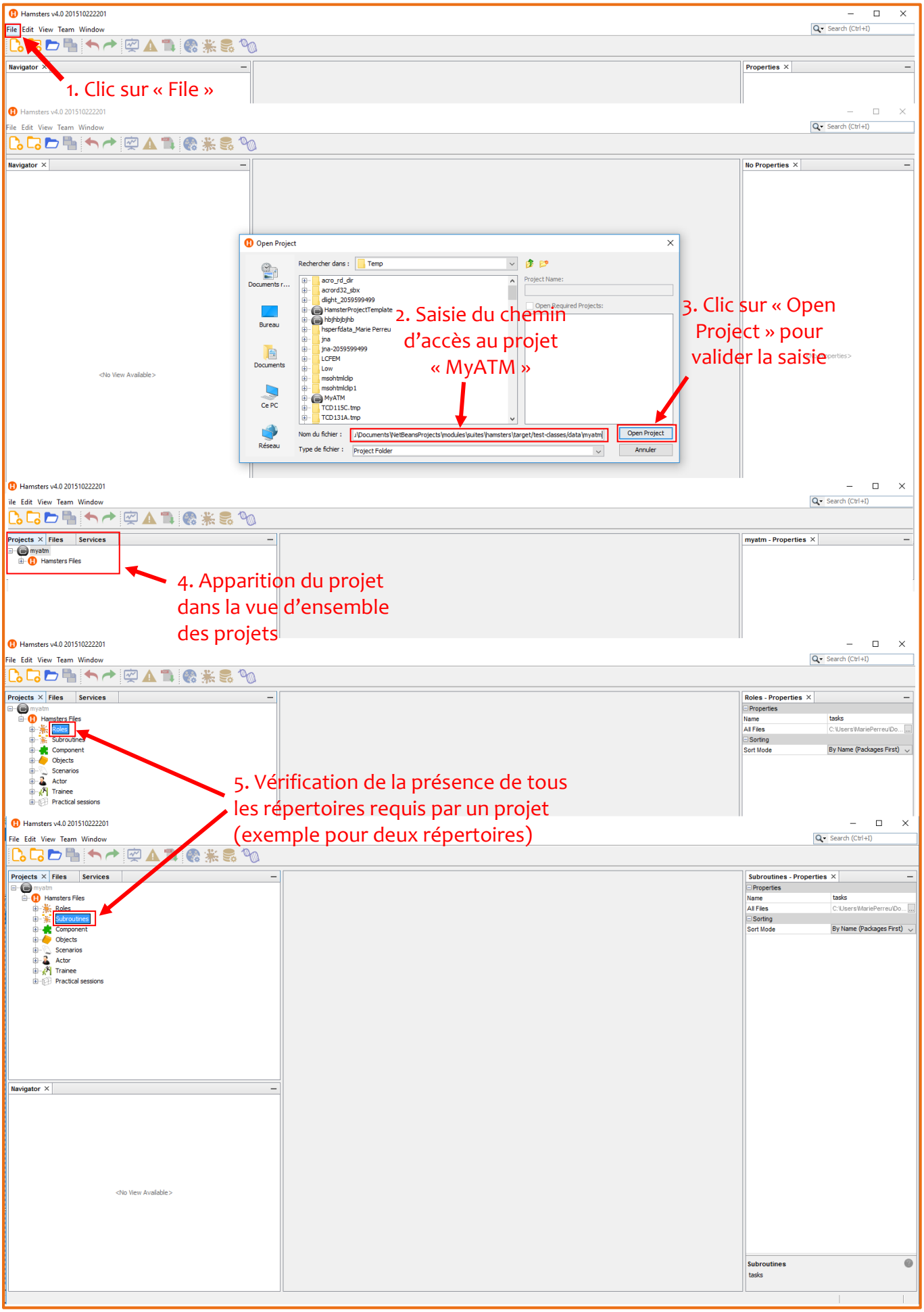
c) Spécification et codage des tests de non-régression

La réalisation de cette mission comprend notamment la spécification des tests de non-régression. Cette partie recense en détail les différentes **fonctionnalités** du logiciel, comment on les **utilise** et quels **résultats** elles produisent précisément. Les documents produits m'ont servis de **support** pour la programmation des tests. Pour la suite je vais vous expliquer un peu mieux en quoi consiste réellement un test d'interface codé à l'aide de l'outil JellyTools, que **fait-il** vraiment, **visuellement** que voit-on à l'exécution, que **démontre-t-il**, etc. Pour cela je prendrai pour exemple un des tests que j'ai codé, soit le test **d'ouverture du projet** « **MyATM** », projet sur lequel j'ai réalisé tous mes tests. Ce dernier modélise les tâches nécessaires au retrait d'argent grâce à un **distributeur automatique**.

Ce qui est important de comprendre c'est que le **concept** de réalisation de tests d'interface repose sur le principe de **remplacer la personne** qui teste le logiciel manuellement par du **code** qui fait automatiquement ce que ferait cette personne. Ainsi une **instruction** de code va permettre alors réaliser un **clic souris** sur un bouton, **déplacer la souris** à un endroit sur l'écran, **taper au clavier** pour saisir du texte, etc. Le test concret d'ouverture du projet « MyATM » consiste en **plusieurs actions**. Dans un premier temps il faut cliquer sur **« File »** puis sur « **Open Project** », après quoi une fenêtre de dialogue s'ouvre. Dans cette dernière il faut saisir le **chemin d'accès** au projet « MyATM » et enfin cliquer sur le bouton « **Open Project** » pour **valider** l'opération. Après cette suite séquentielle d'événements, le projet souhaité va apparaitre dans la vue d'ensemble des projets, ce qui signifie que le projet « MyATM » s'est bien **ouvert** et peut être consulté et édité. Vous pourrez consulter le code ce test au niveau de l'annexe n°6.

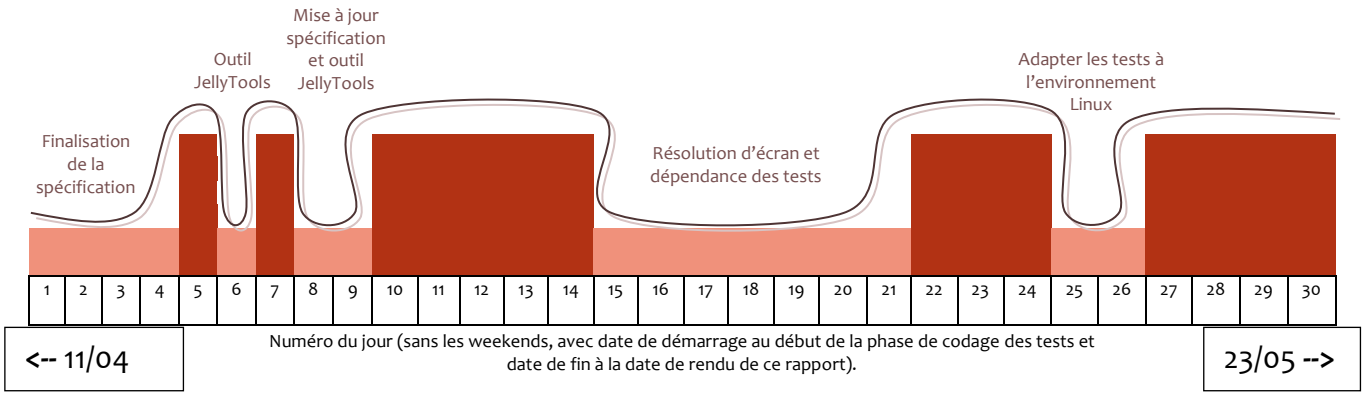
[11] Langage de programmation informatique orienté objet qui a la particularité d'être très facilement portable sur plusieurs systèmes d'exploitation, tels que Windows, Linux, MacOS, etc.

Voici ci-dessous les captures d'écran montrant les étapes principales d'exécution du test :



B. La gestion des difficultés rencontrées

Pour cette partie j'ai pris du recul sur mon expérience et j'ai explicité les périodes durant lesquelles j'ai dû faire face à des **contretemps**. Ainsi, j'ai retenu plusieurs **phases** de complications que vous pourrez identifier sur le diagramme ci-dessous. Ce diagramme représente en **rouge foncé** les jours durant lesquels j'ai **avancé** dans mon travail, c'est-à-dire les jours où j'ai codé des tests en plus et où j'ai mis à jour les documents de spécification. En revanche, en rouge clair ce sont les jours où j'ai rencontré différents types de **problèmes** et donc où je n'ai pas pu automatiser de nouveaux tests. Notons que j'ai fait le choix de ne recenser que la période du **11 avril au 23 mai** sans compter les weekends, le 11 mai correspondant à la date de commencement du codage des tests et le 23 mai étant la date de rendu de ce document.



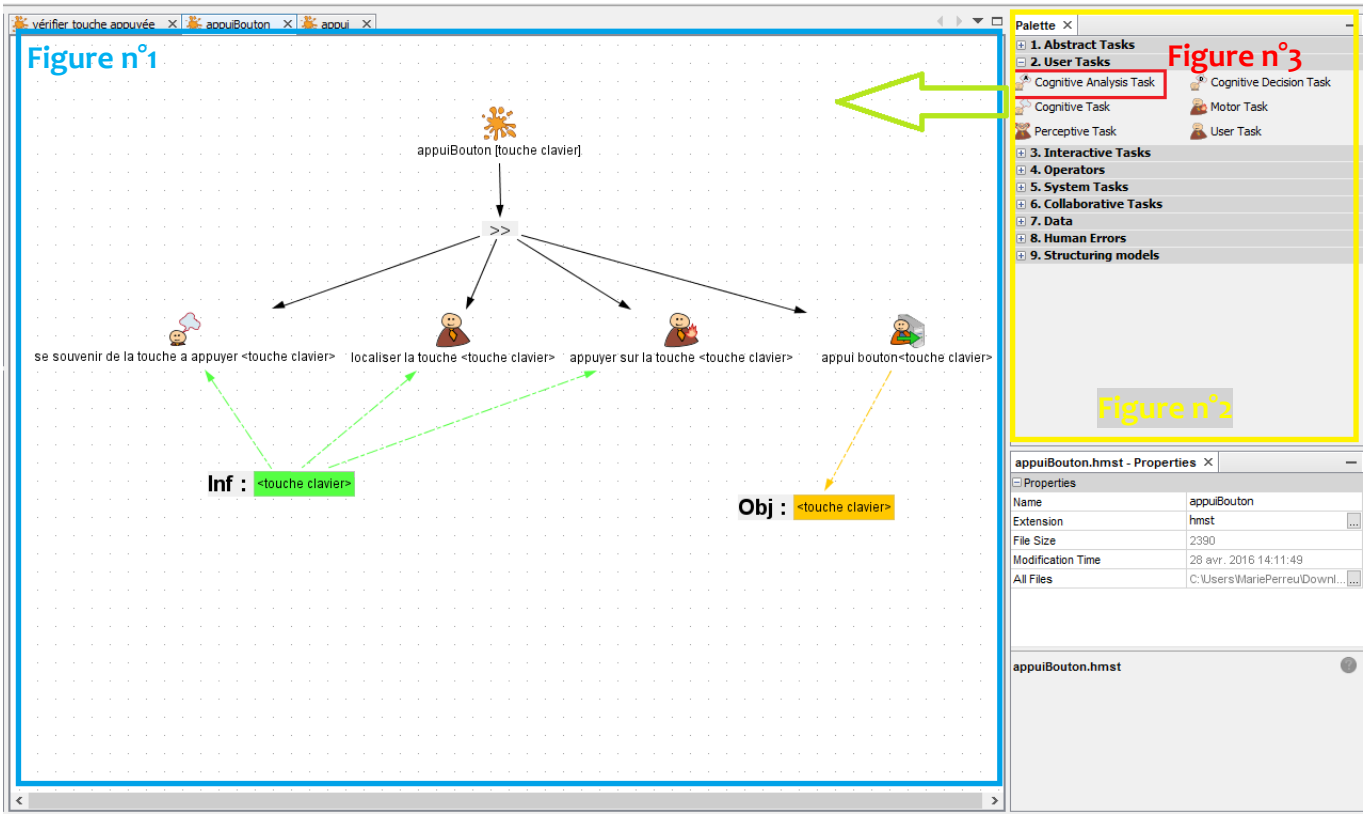
Au total 31 jours se sont écoulés pour la troisième étape de ma mission, soit l'automatisation de test de non-régression. Au cours de cette réalisation j'ai rencontré **5 principaux problèmes** qui m'ont ralenti dans mon travail.

Dans un premier temps, il a fallu que je **finalise** l'étape de la spécification des tests pour pouvoir entamer **rigoureusement** la partie programmation. Durant ces 4 jours j'ai pris en compte les **retours** de Célia Martinie sur mon travail et j'ai corrigé les erreurs présentes dans les différents documents. Il est important que cette étape ne soit pas négligée car c'est elle qui va servir de **support** pour l'automatisation des tests et sa mise à jour régulière permet de garder une **traçabilité** des tests en tout temps. Cette étape n'a pas été un réel problème mais il était **essentiel** de la finaliser pour me permettre de passer à l'étape **concrète** de ma mission.

Ensuite, une fois l'étape du codage amorcée j'ai tout de suite compris que l'utilisation de la librairie **JellyTools** allait être la difficulté majeure. Je n'ai jamais été confrontée à cet outil et celui-ci possède peu de documentation, j'ai donc dû approfondir mes recherches ce qui m'a pris plus de temps que prévu.

De plus, au fur et à mesure de mon avancement il a fallu que je me concentre sur de **nouvelles notions** de JellyTools qui étaient de plus en plus **spécifiques**. Ainsi, ces nouvelles connaissances m'ont amené à modifier certains déroulements de tests et j'ai dû prendre du temps pour **adapter** le code et pour bien **mettre à jour** les documents de spécification, ce qui était essentiel pour garantir un **suivi fiable** de l'automatisation des tests.

Une fois **l'outil** et les **mécanismes** bien pris en main j'ai pu avancer raisonnablement sur une période de 5 jours. Et après avoir soumis les tests effectués aux autres membres de l'équipe nous nous sommes rendu compte d'un problème concernant une partie des plus **délicates** de mon travail. Comme mentionné plus haut, ma mission consistait à rendre automatiques des tests d'interface, comme par exemple simuler **un glisser déposer** d'un composant (« Cognitive Analysis Task », Figure n°3) sur la **scène** (Figure n°1).



Après beaucoup d'essais j'ai réussi à simuler correctement cette action en utilisant une méthode consistant à déterminer une position (X, Y) du composant dans la « **Palette** » (Figure n°2) ainsi sa position future sur la scène. Ainsi, à l'aide d'un **mécanisme de simulation** de clic souris et de déplacement de la souris je pouvais effectuer le glisser déposer d'un point fixe de **départ** vers un point fixe d'**arrivée**. Cela marchait correctement sur ma machine mais pas sur les machines des autres membres de l'équipe. En effet, par manque de recul et de pratique je n'ai pas pensé que la **résolution de l'écran** ferait varier les **valeurs fixes** que j'avais déterminées pour chaque composant à faire glisser sur la scène. Il a donc fallu que je trouve des **référentiels de base communs** pour la scène et pour la

« Palette ». Pour cela un des membres de l'équipe, **Eric Barboni**, m'a beaucoup aidé à trouver la solution et j'ai repris l'avancement de ma mission.

Ainsi, après un stade d'avancement raisonnable il était intéressant de vérifier la **qualité** des tests programmés. Pour cela il faut regarder à quel point les tests couvrent le **code source** [12]: on dit que l'on réalise la **couverture des tests**[13]. Une bonne couverture signifie que la majeure partie du code est testée et que ce dernier garantit d'identifier un nombre **maximum** de comportements problématiques du logiciel. L'environnement **Linux** [14] permet cette évaluation qualitative, c'est pourquoi j'ai dû entreprendre **l'adaptation** des tests à ce système d'exploitation. Ainsi, pour assurer cette **portabilité**[15] j'ai travaillé 2 jours durant lesquels j'ai rencontrés des problèmes de **compatibilité** de code. J'ai réalisé des recherches et j'ai pu régler ce contretemps assez rapidement ce qui m'a permis de continuer à avancer dans l'automatisation de nouveaux tests.

En somme, le **diagramme** vu plus haut me permet d'illustrer les contrastes liés à mon avancement en différenciant les jours où j'ai avancé réellement dans mon travail et les jours où j'ai dû gérer des complications. Je considère cette représentation comme étant un **scénario classique**, car pour moi les contretemps font partie intégrante d'un développement quel qu'il soit et ne signifient pas vraiment que je n'ai pas avancé dans ma mission. En effet, c'est lors de ces imprévus que j'ai **enrichi** mes connaissances, notamment sur le **plan technique** car, forcée de devoir résoudre ces problèmes, j'ai acquis une meilleure **maîtrise** des mécanismes et des outils. Mais aussi sur le **plan organisationnel**, car après chaque contretemps j'ai dû redoubler d'efforts pour respecter les **objectifs** fixés.

5. Bilan des résultats pour l'équipe iCS

A. Niveau d'atteinte des objectifs

A cette fin de deuxième mois de stage je pense avoir rempli pleinement mes objectifs. En effet, tous les **scénarios** de tests fixés au départ, lors de la rédaction des documents de **spécification**, ont été programmés et fonctionnent sur les environnements **Windows** et **Linux**. Cela va permettre à l'équipe de recherche de pouvoir partir sur une bonne **base** qui leur servira pour améliorer le logiciel en évitant au mieux la **régression**.

B. Apport du travail réalisé

Mon intervention au sein de l'équipe a permis d'apporter une **nouvelle vision** pour le logiciel Hamsters et de **nouvelles connaissances** concernant les tests d'interface à l'aide de la librairie

[12] Liste d'instructions d'un programme exprimées dans un langage que l'Homme est capable de manipuler aisément.
[13] Mesure qui permet d'identifier la proportion du code testé.
[14] C'est un système d'exploitation, au même titre que Windows ou Mac OS.
[15] Capacité à fonctionner dans différents environnements d'exécution.

JellyTools. Mon travail leur assure alors une perspective **d'évolution solide** grâce à des tests **automatisés** qui éviteront la **régression** de **Hamsters** et garantiront son **amélioration**.

C. Perspectives futures pour le dernier mois de stage

Pour la suite des événements il est prévu que je fournisse à l'ensemble de l'équipe un **document complet** présentant les **concepts** de la programmation de tests d'interface à l'aide de JellyTools. En effet, cette **documentation** leur permettra par la suite de pouvoir aisément continuer mon travail en automatisant de nouveaux tests au fur et mesure de **l'évolution** du logiciel **Hamsters**. De plus, il est envisagé une **présentation orale** de tout ce que j'ai appris sous forme de **tutoriel** et des tests d'interface réalisés. Cette soutenance va permettre de **découvrir** les **outils** nécessaires à l'automatisation des tests de non-régression et donc rendre **possibles** et **fiables** de futures évolutions de **Hamsters**.

6. Bilan personnel

A. Enseignements reçus

Durant ce stage de fin de formation de Licence 3 en Informatique plusieurs des **enseignements** reçus m'ont aidé à remplir mes **objectifs**. Tout d'abord les cours **Test et Maintenance de Logiciels** m'ont fournis les connaissances nécessaires pour la programmation de tests, comprenant les **bonnes pratiques** de codage. Ensuite, l'unité d'enseignement **Programmation Evénementielle** durant laquelle j'ai appris à manipuler l'environnement de développement **NetBeans**, ce qui m'a beaucoup aidé dans la prise en main de l'existant. De plus, les cours concernant la **Programmation Orientée Objet** m'ont apporté les notions nécessaires en langage **Java** donnant une base solide pour le codage des tests dans ce langage. Enfin, la majorité de la documentation en ligne étant rédigée en **Anglais**, mes **compétences linguistiques** m'ont aidé dans la compréhension rapide des fonctionnements et autres informations nécessaires à mon avancement.

B. Projet professionnel

Ce stage s'intègre parfaitement dans mes **projets d'études** futurs. En effet, j'ai pour ambition de candidater pour le **Master IHM** proposé par l'Université Paul Sabatier et l'équipe des systèmes interactifs critiques évolue dans ce secteur. Ainsi, j'ai pu acquérir quelques **notions** et m'enrichir de l'**état d'esprit** relatif à ce domaine et au **monde de la recherche** auquel je porte un intérêt certains. Concernant mes intentions **professionnelles** celles-ci ne sont pas encore bien définies mais le métier **d'enseignante-chercheuse** me parait correspondre à mes attentes. En somme, cette expérience professionnelle est réellement **adaptée** à mes désirs d'évolution future et me permet de concrétiser fortement mes **motivations** pour continuer dans le secteur de **l'IHM**.

CONCLUSION

Finalement, cette expérience **professionnelle** se révèle être encore plus enrichissante que ce que j'aurais espéré. Tant sur le plan **technique** que sur le plan **humain** j'ai découvert un domaine qui m'attirait et j'ai confirmé mon vœu de poursuivre dans le secteur des **Interfaces Homme-Machine**.

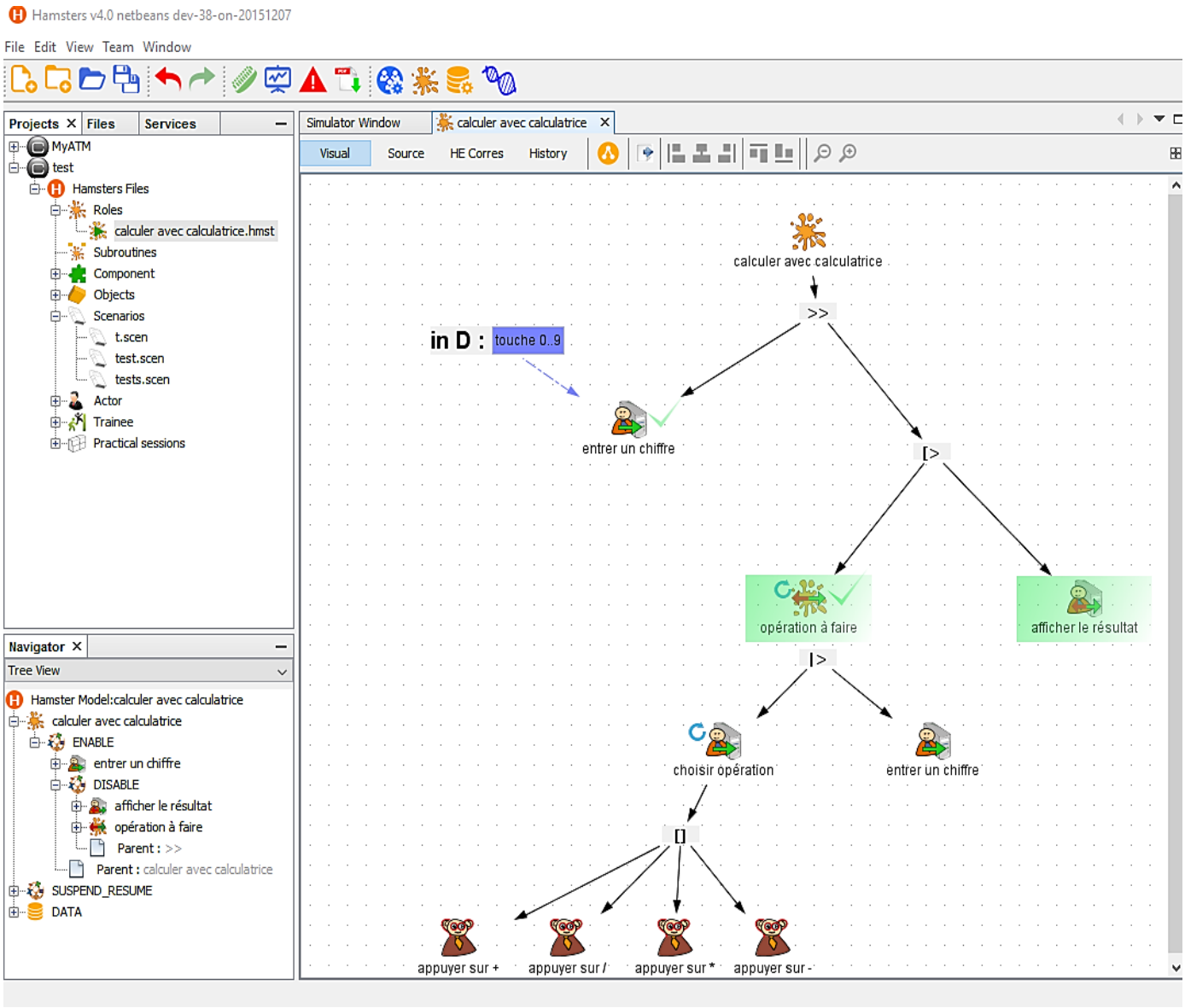
Sur le plan **technique**, l'apprentissage **autonome** des tests d'interface m'a montré que cette facette de l'Informatique était **essentielle**, car un logiciel testé garantit sa **fiabilité** et sa **survie**. J'ai trouvé ma mission particulièrement **formatrice** et **agréable** car, malgré la complexité d'utilisation des différents outils auxquels j'ai été confronté, j'ai apprécié l'aspect **dynamique** et **non-monotone** de cette expérience. En effet, à chaque test programmé je rencontrais de **nouvelles notions** ce qui m'a permis d'acquérir une certaine **maîtrise** des concepts de test d'interface. Je suis également particulièrement fière d'avoir pu apporter de **nouvelles possibilités d'évolution** à l'ensemble de l'équipe iCS. Grâce à un **œil neuf** et aux **nouvelles connaissances** que j'ai pu leur fournir, les membres de l'équipe de recherche vont pouvoir alors entreprendre **l'amélioration** du logiciel **Hamsters** de manière **fiable** et **robuste**. Enfin, les **perspectives** pour le dernier mois de stage vont me permettre d'expérimenter le métier de **formatrice** par le biais d'un **tutoriel** prévu pour l'ensemble de l'équipe. Je vais donc avoir l'occasion de **préparer** un **cours**, de faire **partager** mes connaissances et d'exposer le travail que j'ai effectué.

Sur le plan **humain**, cette immersion dans le **monde de la recherche** m'a permis de **découvrir** une manière de contribuer au large domaine qu'est **l'Informatique**. J'ai apprécié **l'état d'esprit** qui régnait au sein de l'équipe de recherche, **l'ambition** toujours plus grande et **la rigueur** dans le travail à effectuer qui m'ont convaincu. De plus, l'accueil que j'ai reçu a fortement contribué à mon **intégration** au sein de l'iCS et donc à mon **investissement** certain dans ma mission de stage.

En somme, cette **riche expérience** a répondu à mes attentes et me permet aujourd'hui d'affirmer mes **motivations** pour concrétiser mon **projet futur** d'étude.

ANNEXES

Annexe n°1 : *Projet personnel lors de l'apprentissage des fonctionnalités du logiciel Hamsters*



🛠 Annexe n°2 : *Extrait du document de spécification abstraite du logiciel Hamsters*

1.CONFIGURATION PROJET			
Identifiants des tests	**Actions possibles**	**Comment ?**	**Résultat attendu**
SAH_01CP	Créer un projet	Depuis l'onglet « File » : 1.Clic sur File > New project … 2.Catégorie -> « Hamsters-Petshop » Projets -> « Hamsters Petshop » 3.Clic sur « Next » 4.Remplir le nom du nouveau projet 5.Clic sur « Finish » Depuis la barre des tâches : 1.Clic sur l'icône de création d'un nouveau projet dans la barre des tâches 2.Catégorie -> « Hamsters-Petshop » Projets -> « Hamsters Petshop » 3.Clic sur « Next » 4.Remplir le nom du projet 5.Clic sur « Finish »	✓ Le nouveau projet apparaît dans la section « Projects » de vue d'ensemble du projet en cours de configuration ✓ Le projet présente différents répertoires : ☒ Roles ☒ Subroutine ☒ Component ☒ Objects ☒ Scenarios ☒ Actor ☒ Trainee ☒ Practical sessions
SAH_02OP	Ouvrir un projet	Depuis l'onglet « File » : 1.Clic sur File > Open project … 2.Sélectionner le Project Folder souhaité 3.Clic sur « Open Project » Depuis la barre des tâches : 1.Clic sur l'icône d'ouverture d'un fichier de la barre des tâches 2.Sélectionner le Project Folder souhaité 3.Clic sur « Open Project »	✓ Le projet voulu apparaît dans la section « Projects » de vue d'ensemble du projet en cours de configuration ✓ Le projet présente différents répertoires : ☒ Roles ☒ Subroutine ☒ Component ☒ Objects ☒ Scenarios ☒ Actor ☒ Trainee ☒ Practical sessions
SAH_03CR	Créer un rôle	1. Clic droit sur le répertoire « Roles » dans la section « Projects » 2. Clic sur « Add Role » 3. Entrer le nom du nouveau rôle 4. Clic sur « Add »	✓ Le nouveau rôle s'affiche dans le répertoire « Roles » de la vue d'ensemble du projet en cours de configuration (section « Projects »)
SAH_04CMT	Créer un modèle de tâche	1. Clic droit sur le répertoire « Roles » dans la section « Projects » 2. Clic sur New > Other… 3. Catégorie -> « Modeling » > « Hamsters » Type de fichier -> « TaskModel.hmst »	✓ Le nouveau modèle de tâches s'affiche dans le répertoire « Roles » de la vue d'ensemble du projet en cours de configuration (section « Projects »)
SAH_05DMR	Déplacer un modèle de tâche dans un répertoire, pour l'affilier à un rôle précis	1. Maintenir un clic gauche sur un modèle de tâche et faire glisser sur la cible, un rôle dans le répertoire « Roles »	✓ Déplacement du modèle de tâche dans le répertoire cible ✓ Modèle de tâches affilié à un rôle
SAH_06AMPR	Assigner un modèle de tâches principal pour un rôle	1. Clic droit sur un modèle de tâche 2. Clic sur « Set as a Main Role Model »	✓ Un logo vert se rajoute sur l'icône du modèle de tâche pour indiquer que le modèle est le modèle principal du rôle

2.EDITION

Identifiants des tests	Actions possibles	Comment ?	Résultat attendu
SAH_07AT	Ajouter une tâche à un modèle de tâches	1. Sélectionner la tâche voulue parmi les tâches possibles dans la section « Palette » à droite de l'écran 2. Faire glisser la tâche à l'endroit souhaité sur la scène (zone graphique d'édition de modèle)	✓ La tâche apparaît sur la scène avec un nom prédéfinit selon son type ✓ La tâche apparaît dans la section « Navigator » en bas à gauche de l'écran
SAH_08AO	Ajouter un opérateur	1. Sélectionner l'opérateur voulu parmi les opérateurs proposés dans la section « Palette », partie 4 2. Faire glisser l'opérateur à l'endroit souhaité sur la scène	✓ Le nouvel opérateur apparaît sur la scène ✓ L'opérateur apparaît dans la section « Navigator » en bas de l'écran à gauche
SAH_09CS	Créer une subroutine	↳ A partir de la « Palette » : 1.Aller dans la partie 9 de la section « Palette » 2.Faire glisser une subroutine à l'endroit souhaité sur la scène ↳ A partir d'un modèle de tâche : 1.Clic droit sur une tâche du modèle de tâche 2.Clic sur « Create subroutine »	✓ La subroutine apparaît dans le répertoire « Subroutine » de la section « Projects » ✓ La tâche prend l'aspect d'une subroutine (icône), affichage de la subroutine sur la scène ✓ La nouvelle subroutine indique ses entrées et sorties ✓ La subroutine apparaît dans la section « Navigator » en bas à gauche de l'écran ✓ Une fenêtre s'ouvre pour l'édition de la nouvelle subroutine
SAH_10LS	Choix d'une subroutine à lier depuis un autre modèle de tâche (subroutine créée au préalable)	1. Créer une subroutine à partir de la section « Palette » comme vue précédemment 2. La relier à un opérateur 3. Dans la section « Properties » en bas à droite de l'écran, chercher la ligne « Subroutine » 4. Clic sur la cellule en face 5. Sélectionner la subroutine déjà créée dans un autre modèle	✓ La nouvelle subroutine prend le nom de la subroutine liée ✓ Ses entrées et sorties sont affichées
SAH_11AIES	Ajouter une information en entrée d'une subroutine	1. Aller dans la partie 7 « Data », de la section « Palette » 2. Faire glisser sur la scène la donnée choisie 3. Maintenir appuyer sur la touche Ctrl 4. Maintenir un clic gauche sur la nouvelle donnée en se déplaçant sur la subroutine (maintenir la touche Ctrl) 5. Relâcher	✓ Les entrées de la subroutine sont mises à jour ✓ Affichage de la subroutine sur le modèle de tâches modifié
SAH_12AISS	Ajouter des informations en sortie d'une subroutine	1. Aller dans la partie 7 « Data » de la section « Palette » 2. Faire glisser la donnée choisie sur la scène 3. Maintenir appuyer sur la touche Ctrl 4. Maintenir un clic gauche sur la subroutine en se déplaçant sur la nouvelle donnée (maintenir la touche Ctrl) 5. Relâcher	✓ Les sorties de la subroutine sont mises à jour ✓ Affichage de la subroutine sur le modèle de tâches modifié

🛠 ***Annexe n°3 :*** *Extrait du document de spécification concrète de tests à travers le projet « MyATM »*

1.CONFIGURATION PROJET : 4/4					
Identifiants des tests	**Actions possibles**	**Comment ?**	**Résultat attendu**	**Windows**	**Linux**
SCH_01OP	Ouvrir le projet ATM	1. Clic sur « File » > « Open Project… » 2. Entrer le chemin d'accès au projet « MyATM » 3. Clic sur « Open Project »	✓ Ajout du projet dans la section « Projects », avec les différents répertoires de base : + Roles + Subroutine + Component + Objects + Scenarios + Actor + Trainee + Practical sessions		
SCH_02ART	Ajouter le rôle « Testeur »	Prérequis : ouverture du projet « MyATM », suppression du rôle « Testeur » s'il existe (ici pour les tests l'opération est faite depuis le système de fichiers). 1. Clic droit sur le répertoire « Roles » de la section « Projects » 2. Clic sur « Add role » 3. Entrer le nom « Testeur » 4. Clic sur « Add »	✓ Ajout du rôle « Testeur » dans le répertoire « Roles » de la section « Projects »		
SCH_03AMT	Ajouter un modèle de tâches au rôle « Testeur », appelé « TesteurTaskModel »	Prérequis : ouverture du projet « MyATM », suppression du modèle de tâches « TesteurTaskModel.hmst » s'il existe (ici pour les tests l'opération est faite depuis le système de fichiers). 1. Clic sur « File » > « New File… » 2. Choisir la catégorie « Modeling » > « Hamsters » 3. Choisir le type de fichier « TaskModel.hmst » 5. Clic sur « Next » 6. Remplir le nom avec « TesteurTaskModel » 7. Clic sur « Finish » 8. Dans la section « Projects » répertoire « Roles », faire glisser le nouveau modèle de tâches créé dans le rôle « Testeur »	✓ Ajout du modèle « TesteurTaskModel » dans le répertoire du rôle « Testeur » de la section « Projects »		
SCH_04MTP	Mettre le modèle de tâches de « Testeur » en modèle principal du rôle	Prérequis : ouverture du projet « MyATM », ouverture du fichier « TesteurTaskModel.hmst » (« File » > « Open File », entrer le chemin d'accès au fichier, clic sur « Ouvrir »). 1. Clic droit sur le modèle de tâche « TesteurTaskModel.hmst » 2. Clic sur « Set as Main Role model »	✓ Ajout d'une icône sur le modèle de tâche dans la section « Projects » dans l'onglet du rôle « Testeur »		

2. EDITION : 9/9

Identifiants des tests	Actions possibles	Comment ?	Résultat attendu	Windows	Linux
SCH_05ATA	Ajouter une tâche abstraite au modèle de tâche « UserTaskModel.hmst », du rôle « User »	Prérequis : ouverture du projet « MyATM », ouverture du fichier « UserTaskModel.hmst » » (« File » > « Open File », entrer le chemin d'accès au fichier, clic sur « Ouvrir »). 1. Faire glisser une tâche abstraite (Abstract Task) de la section Palette partie 1, sur la scène	✓ La tâche a le nom du modèle de tâche en cours de réalisation, soit « UserTaskModel » ✓ Ajout de la tâche dans la section en bas à gauche, « Navigator »		
SCH_06AOE	Ajouter l'opérateur « Enable »	Prérequis : ouverture du projet « MyATM », ouverture du fichier « UserTaskModelVersionTestOperatorEnable.hmst » » (« File » > « Open File », entrer le chemin d'accès au fichier, clic sur « Ouvrir »). 1. Dans la section « Palette », partie 4 « Operators », faire glisser un opérateur « Enable » sur la scène 2. Relier le nouvel opérateur avec la tâche abstraite créée précédemment	✓ L'opérateur a été ajouté dans la section « Navigator », rangé dans le répertoire de la tâche abstraite « UserTaskModel » (hiérarchie)		
SCH_07AIT	Ajouter d'une « Input task »	Prérequis : ouverture du projet « MyATM », ouverture du fichier « UserTaskModelVersionTestInputTask.hmst » » (« File » > « Open File », entrer le chemin d'accès au fichier, clic sur « Ouvrir »). 1. Dans la section « Palette », partie 3 « Interactive Task », faire glisser sur la scène une « Input Task » 2. Relier la tâche créée avec l'opérateur « Enable »	✓ Ajout de la tâche dans la section "Navigator", rangée dans le répertoire de l'opérateur créé précédemment ✓ Cette nouvelle tâche est nommée de manière automatique avec "Input Task1"		
SCH_08RIT	Renommer l'Input task en la nommant « appui bouton »	Prérequis : ouverture du projet « MyATM », ouverture du fichier « UserTaskModelVersionTestRenameInputTask.hmst » » (« File » > « Open File », entrer le chemin d'accès au fichier, clic sur « Ouvrir »). ➢ Dans la section « Properties », à la ligne « description », en face faire un clic et puis entrer le nom "appui bouton"	✓ Modification du nom de la tâche dans la section « Navigator » ✓ Modification du nom de la tâche dans la section « Properties »		
SCH_09ATS	Ajouter une tâche système	Prérequis : ouverture du projet « MyATM », ouverture du fichier « UserTaskModelVersionTestSystemTask.hmst » » (« File » > « Open File », entrer le chemin d'accès au fichier, clic sur « Ouvrir »). 1. Dans la section « Palette », partie 5 « System tasks », faire glisser une « System task » sur la scène 2. Relier la tâche créée avec l'opérateur « Enable »	✓ Cette tâche se nomme « System Task1 » ✓ Ajout de la tâche dans le répertoire de l'opérateur « Enable » dans la section « Navigator »		
SCH_10RTS	Renommer la tâche système avec « vérifier touche appuyée »	Prérequis : ouverture du projet « MyATM », ouverture du fichier « UserTaskModelVersionTestRenameSystemTask.hmst » » (« File » > « Open File », entrer le chemin d'accès au fichier, clic sur « Ouvrir »).	✓ Modification du nom de la tâche dans la section « Navigator » ✓ Modification du nom de la tâche dans la section « Properties »		

🛠 **Annexe n°4 :** *Extrait du document consacré aux problèmes rencontrés et aux suggestions*

Suggestion	Disfonctionnement	Description du disfonctionnement	Description de la suggestion éventuelle
		Quand on créé un composant, on veut l'utiliser dans un modèle de tâches en lui donnant un paramètre spécifique. Si on fait glisser un nouveau composant depuis la palette vers la scène, pour lui spécifier un paramètre relatif au modèle en cours. Au moment de lier le nouveau composant à un composant on a besoin de lui donner aussi le paramètre nécessaire. Dans le système actuel on va faire clic droit sur le nouveau composant et clic gauche sur « Instancied component ». Cela va ouvrir une fenêtre de dialogue permettant de spécifier le composant à lier et le nom paramètre.	On pourrait avoir cette possibilité à partir de la zone de modification des paramètres de fichier ou d'éléments (section des « Properties » en bas à droite de l'écran). On pourrait alors ajouter une ligne « Instancied component » et choisir le composant à lier. Cela ouvrirait la fenêtre de dialogue pour choisir le paramètre à spécifier. On pourrait même ne pas ouvrir de fenêtre et rajouter une ligne dans l'onglet « Component » appelée « Parameters » et en face on entrerait le nom du paramètre.
	(au sens de la compréhension) 	Lors de l'ajout d'un nouveau composant depuis la section « Projects » dans le répertoire « Component ». Cela nous ouvre une fenêtre permettant de choisir la nature du fichier. Ici l'icône du composant est la même que celle du modèle de tâches.	Il faudrait une icône différente pour faciliter la compréhension de l'utilisateur.
	(au sens ou l'utilisateur va chercher à faire une opération inexistante qui serait « mieux ») 	Actuellement, le système ne permet pas d'effectuer un clic droit sur un rôle particulier dans le répertoire « Roles » de la section « Projects ».	Cela serait plus pratique de pouvoir, par exemple, créer un modèle de tâche principal pour un rôle choisi.
		Le déplacement sur la scène à travers le modèle de tâche se fait à l'aide des barres de défilement sur le côté.	Peut-être qu'il plus pratique de pouvoir également se déplacer à l'aide d'une main, en maintenant un clic sur l'arrière-plan de la scène. Ainsi, le déplacement de la souris engendrerait un déplacement sur le modèle de tâches.
	(au sens où on perd l'utilisateur) 	Lors de l'ajout d'un nouveau composant celui-ci se met dans le répertoire « Roles » de manière « aléatoire ».	Peut-être que l'on pourrait créer un répertoire dédié aux composants d'un modèle de tâches d'un rôle.
		Quand on fait glisser un composant sur la scène celui-ci prend un nom définit par le système.	Il faudrait que ce nom soit cohérent comme pour l'utilisation des subroutines.
		Quand on fait glisser un composant sur la scène celui-ci ne s'ajoute pas dans le répertoire « Component » de la section « Projects ».	Pour une meilleure utilisation cela serait plus judicieux de le retrouver à cet endroit-là.
		Quand on fait clic droit sur le répertoire « Component » de la section « Projects » on a la possibilité de créer un modèle de tâche. C'est une action qui n'a pas de rapport direct avec l'utilisation des composants.	

🛠 ***Annexe n°5 :*** *Extrait du document de versionnage de tous les documents*

Auteur : Marie Perreu

Mis à jour le : 19/05/16

Tableau des versions

Nom du document	Nom de l'auteur	Date	Version
Version abstraite de spécification des tests pour le logiciel HAMSTERS	Marie Perreu	08/04/16	1
Présentation des suggestions et des disfonctionnement (bugs) rencontrés lors de l'utilisation de Hamsters	Marie Perreu	08/04/16	1
Version concrète (Projet ATM) de spécification des tests pour le logiciel HAMSTERS	Marie Perreu	15/04/16	1
Version concrète (Projet ATM) de spécification des tests pour le logiciel HAMSTERS	Marie Perreu	19/04/16	2
Version concrète (Projet ATM) de spécification des tests pour le logiciel HAMSTERS	Marie Perreu	22/04/16	3
Version concrète (Projet ATM) de spécification des tests pour le logiciel HAMSTERS	Marie Perreu	25/04/16	4
Version concrète (Projet ATM) de spécification des tests pour le logiciel HAMSTERS	Marie Perreu	26/04/16	5
Version concrète (Projet ATM) de spécification des tests pour le logiciel HAMSTERS	Marie Perreu	27/04/16	6

🛠 Annexe n°6 : Code du test d'ouverture du projet « MyATM »

```
/**
 * ****************************** TEST CODE : ***********************************
 ******************************** SCH_01OP ************************************
 ******************************************************************************
 */
package testInterfaceHamsters;
import java.awt.AWTException;
import java.io.File;
import java.io.IOException;
import org.netbeans.jellytools.JellyTestCase;
import org.netbeans.jellytools.NbDialogOperator;
import org.netbeans.jellytools.ProjectsTabOperator;
import org.netbeans.jellytools.actions.ActionNoBlock;
import org.netbeans.jellytools.nodes.Node;
import org.netbeans.jemmy.operators.JButtonOperator;
import org.netbeans.jellytools.MainWindowOperator;
import static junit.framework.TestCase.assertTrue;

/**
 *
 * @author Marie Perreu
 */
public class T01_OpenProjectATM extends JellyTestCase {

    public T01_OpenProjectATM(String name) {
        super(name);
    }

    @Override
    protected int timeOut() {
        return 6000;
    }

    public void setUp() throws Exception {
        super.setUp();
        System.out.println("##############################################################################");
        System.out.println("   * * * * * *   " + getName());
        System.out.println("##############################################################################\n");
    }

    /**
     * Test d'ouverture du projet ATM
     *
     * @throws InterruptedException
     * @throws AWTException
     */
    public void testOuvertureProjetATM() throws InterruptedException, AWTException, IOException {
        clearTestStatus();
        startTest();
        MainWindowOperator aDefault = MainWindowOperator.getDefault();
        aDefault.maximize();
        aDefault.activate();

        /*************************************
         ********** OPEN PROJECT ATM ************
         *************************************/

        //simule l'action utilisateur : clic sur File puis Open Project.. dans la barre de menu de
    l'interface Hamsters qui vient de s'ouvrir
        new ActionNoBlock("File|Open Project...", null).performMenu();
        //fait une pause le temps que la fenetre de dialogue d'ouverture de projet s'ouvre

        //creation d'un robot qui va effectuer l'appui sur les touches du clavier (celles des caracteres du
chemin d'acces au projet a ouvrir)
        Actions action = new Actions();
        String property = System.getProperty("xtest.data");
        String filePath = property + File.separator + "myatm";
        //rentre le texte dans le champ de saisi
        action.ecrit(filePath);
        //simulation d'un clic sur le bouton "Open Project"
           if(System.getProperty("os.name").contains("nux") ||
       System.getProperty("os.name").contains("mac")){
         JButtonOperator jbo = new JButtonOperator(new NbDialogOperator("Open Project"), "Open Project");
            jbo.push();
            jbo.push();
        } else {
            new JButtonOperator(new NbDialogOperator("Open Project"), "Open Project").push();
        }
        //ouverture du projet
        Thread.sleep(1000);
```

```
        /*****************************************
         ********** PROJECT ATM OPENED ************
         *****************************************/

        ProjectsTabOperator pto = new ProjectsTabOperator();
        Node nodeRoles = new Node(pto.tree(), "MyATM|Hamsters Files|Roles");
        nodeRoles.select();
        //si le programme ne plante pas sur une assertion failed alors c'est ok le test est passe
        //ici on teste que le test amene bien a la creation d'un noeud "Roles" dans la section "Projects"
        assertTrue(nodeRoles.isPresent());
        Thread.sleep(500);
        System.out.println("-- T01_OpenProjectATMTest -- ######### TEST NOEUD ROLES VALIDATE #########");

        pto = new ProjectsTabOperator();
        Node nodeSubroutines = new Node(pto.tree(), "MyATM|Hamsters Files|Subroutines");
        nodeSubroutines.select();
        //si le programme ne plante pas sur une assertion failed alors c'est ok le test est passe
        //ici on teste que le test amene bien a la creation d'un noeud "Subroutines" dans la section
"Projects"
        assertTrue(nodeSubroutines.isPresent());
        Thread.sleep(500);
        System.out.println("-- T01_OpenProjectATMTest -- ###### TEST NOEUD SUBROUTINES VALIDATE ######");

        pto = new ProjectsTabOperator();
        Node nodeComponent = new Node(pto.tree(), "MyATM|Hamsters Files|Component");
        nodeComponent.select();
        //si le programme ne plante pas sur une assertion failed alors c'est ok le test est passe
        //ici on teste que le test amene bien a la creation d'un noeud "Component" dans la section
"Projects"
        assertTrue(nodeComponent.isPresent());
        Thread.sleep(500);
        System.out.println("-- T01_OpenProjectATMTest -- ####### TEST NOEUD COMPONENT VALIDATE ######");

        pto = new ProjectsTabOperator();
        Node nodeObjects = new Node(pto.tree(), "MyATM|Hamsters Files|Objects");
        nodeObjects.select();
        //si le programme ne plante pas sur une assertion failed alors c'est ok le test est passe
        //ici on teste que le test amene bien a la creation d'un noeud "Objects" dans la section "Projects"
        assertTrue(nodeObjects.isPresent());
        Thread.sleep(500);
        System.out.println("-- T01_OpenProjectATMTest -- ######## TEST NOEUD OBJECTS VALIDATE ########");

        pto = new ProjectsTabOperator();
        Node nodeScenarios = new Node(pto.tree(), "MyATM|Hamsters Files|Scenarios");
        nodeScenarios.select();
        //si le programme ne plante pas sur une assertion failed alors c'est ok le test est passe
        //ici on teste que le test amene bien a la creation d'un noeud "Scenarios" dans la section
"Projects"
        assertTrue(nodeScenarios.isPresent());
        Thread.sleep(500);
        System.out.println("-- T01_OpenProjectATMTest -- ####### TEST NOEUD SCENARIOS VALIDATE #######");

        pto = new ProjectsTabOperator();
        Node nodeActor = new Node(pto.tree(), "MyATM|Hamsters Files|Actor");
        nodeActor.select();
        //si le programme ne plante pas sur une assertion failed alors c'est ok le test est passe
        //ici on teste que le test amene bien a la creation d'un noeud "Actor" dans la section "Projects"
        assertTrue(nodeActor.isPresent());
        Thread.sleep(500);
        System.out.println("-- T01_OpenProjectATMTest -- ######## TEST NOEUD ACTOR VALIDATE ########");

        pto = new ProjectsTabOperator();
        Node nodeTrainee = new Node(pto.tree(), "MyATM|Hamsters Files|Trainee");
        nodeTrainee.select();
        //si le programme ne plante pas sur une assertion failed alors c'est ok le test est passe
        //ici on teste que le test amene bien à la creation d'un noeud "Trainee" dans la section "Projects"
        assertTrue(nodeTrainee.isPresent());
        Thread.sleep(500);
        System.out.println("-- T01_OpenProjectATMTest -- ######## TEST NOEUD TRAINEE VALIDATE ########");

        pto = new ProjectsTabOperator();
        Node nodePracticalSessions = new Node(pto.tree(), "MyATM|Hamsters Files|Practical sessions");
        nodePracticalSessions.select();
        //si le programme ne plante pas sur une assertion failed alors c'est ok le test est passe
        //ici on teste que le test amene bien a la creation d'un noeud "Pratical sessions" dans la section
"Projects"
        assertTrue(nodePracticalSessions.isPresent());
        Thread.sleep(500);
        System.out.println("-- T01_OpenProjectATMTest -- ## TEST NOEUD PRACTICAL SESSIONS VALIDATE ##");
        endTest();
    }
}
```