uOttawa

L'Université canadienne
Canada's university

Amélioration de l'outil de génération de rapports de jUCMNav

Rapport présenté à : Professeur Daniel Amyot

CSI 4900 – Honours Projects/Projets de recherche

Par :

Alexandre Hamel (numéro d'étudiant : 5238030)

Jean-François Séguin (numéro d'étudiant : 5188520)

École de science informatique et de génie électrique (SIGE)

Université d'Ottawa

Le 14 décembre 2012

Résumé

Au cours de la session d'automne 2012, nous avons travaillé sur une nouvelle version de l'outil de génération de rapports de jUCMNav (*Java Use Case Map Navigator*). Ce plugin, pouvant être intégré à l'environnement de développement intégré Eclipse, permet de créer et de modifier des modèles URN (ou *User Requirements Notation*), comprenant la définition de diagrammes de cas-types, de diagrammes GRL et de scénarios (entre autres). Dès le début du mois de septembre 2012, nous avons élaboré une liste d'exigences dans le but de bien définir les nouvelles fonctionnalités qui devaient être ajoutées au générateur de rapports, et bien cerner les bogues que nous devions réparer.

Après nous être familiarisés avec jUCMNav, nous avons pu commencer à effectuer des changements à cet outil. Pendant trois mois, nous avons effectué plusieurs changements au générateur de rapports existant. Ces changements comprennent l'ajout d'informations sur les scénarios et les tendances des stratégies GRL au contenu des rapports, l'internationalisation des rapports ainsi que la création d'une nouvelle page de préférences pour le générateur de rapports (afin de de rendre ce dernier beaucoup plus configurable). Par ailleurs, nous avons aussi corrigé de nombreux bogues faisant partie du générateur de rapports, tels que les comportements inattendus lors de la navigation dans un rapport en format HTML, la gestion des pages dans les rapports générés en format PDF/RTF et les figures mal redimensionnées dans les rapports RTF. Pour bien documenter nos changements, le site Foswiki du projet jUCMNav et le site de suivi de bogues de l'équipe de développement de jUCMNav étaient constamment mis à jour.

Ce projet nous a non seulement permis d'en apprendre davantage sur la norme URN et l'ingénierie des exigences en général, mais nous a aussi permis de voir à quel point l'élaboration d'exigences était importante dans le processus de développement logiciel. Sans cette liste d'exigences, il aurait été facile d'effectuer du développement qui aurait été hors de la portée de notre projet. Cette liste nous a aussi permis de définir quelques éléments à considérer lors du développement des prochaines versions de jUCMNav. De plus, ce projet nous a permis de voir à quel point une bonne documentation était importante lors du processus de développement d'un logiciel. Puisque jUCMNav est un logiciel complexe, la documentation qui était présente sur le site Foswiki du projet jUCMNav nous a grandement aidés tout au long de la session.

Table des matières

Liste des figures

1. Introduction

Le projet sur lequel nous avons travaillé tout au long de la session d'automne 2012 consistait en la construction d'une version améliorée, plus complète et beaucoup plus configurable du générateur de rapports faisant partie de l'outil jUCMNav. jUCMNav (*Java Use Case Map Navigator*) est un plug-in pouvant être installé avec l'environnement de développement intégré Eclipse, et permettant de créer des modèles qui sont conformes avec la norme internationale URN (*User Requirements Notation*, ou notation d'exigences utilisateurs). Cette norme, largement utilisée dans le domaine du génie logiciel (plus précisément dans la discipline de l'ingénierie des exigences), permet de clarifier, d'analyser et de valider des exigences lors du processus de développement de logiciels [1], et comprend deux types de « langages ». Le premier de ces langages est la notation UCM, qui consiste en des diagrammes de cas-types (ou *Use Case Maps*, en anglais). Ces diagrammes démontrent les comportements typiques d'un logiciel ainsi que divers scénarios expliquant les interactions entre ce logiciel et ses utilisateurs [2]. Le deuxième langage est le langage des exigences orienté but (GRL, ou *Goal-oriented Requirement Language*, en anglais), qui décrit de façon explicite des objectifs d'entreprises, des buts à atteindre ainsi que l'élaboration et l'évaluation de diverses alternatives permettant de réaliser ces buts [2]. jUCMNav fournit des éditeurs permettant de créer et de modifier des diagrammes GRL et UCM, et fournit des outils permettant d'évaluer des stratégies GRL et de définir plusieurs scénarios basés sur un/des diagramme(s) UCM.

Le générateur de rapports qui faisait partie de jUCMNav était en fait un prototype qui avait été construit en 2008 pour donner aux utilisateurs n'utilisant pas jUCMNav la possibilité de voir certains éléments des modèles URN créés à l'aide de ce logiciel. Ce prototype, qui pouvait générer des rapports en format PDF, RTF et HTML, ne prenait en compte que certains éléments des modèles URN (telles que des images des diagrammes UCM, des images des diagrammes GRL ainsi qu'une table d'évaluations de stratégies GRL). De plus, puisque cette première version du générateur de rapports avait une structure très rigide, celle-ci comprenait très peu d'éléments configurables et générait les rapports en un seul langage (en anglais). Enfin, de nombreuses exceptions pouvaient être lancées par Eclipse lorsqu'un utilisateur peu familier avec jUCMNav utilisait l'outil de génération de rapports, et certaines erreurs de formatage pouvaient

se glisser dans les rapports-mêmes lorsque les préférences du générateur de rapports n'étaient pas fixées aux valeurs par défaut.

Dans ce rapport, nous discuterons du processus de développement du générateur de rapports pour la nouvelle version de jUCMNav, ainsi que des défis auxquels nous avons dû faire face durant ce processus. La première section de ce rapport discutera des nouvelles fonctionnalités qui ont été ajoutées au générateur de rapports de jUCMNav. Cette section parlera, entre autres, de la personnalisation des rapports et des éléments URN qui ont été ajoutées au contenu des rapports. La deuxième section, quant à elle, parlera des bogues qui ont dû être corrigés, et de l'approche que nous avons utilisée pour résoudre ces défauts. La troisième section, de son côté, discutera de toute la documentation reliée à la nouvelle version de jUCMNav. Finalement, la dernière section de ce rapport discutera des connaissances que nous avons acquises tout au long du développement de la nouvelle version de l'outil de génération de rapports ainsi que des éléments à considérer lors du développement des prochaines versions de cet outil.

2. Nouveaux éléments ajoutés au générateur de rapports

Le prototype de l'outil de génération de rapports qui était disponible au moment où nous avions sélectionné notre projet était très peu configurable et les rapports générés par cet outil ne pouvaient contenir qu'une partie des informations des modèles URN créés à l'aide de jUCMNav. Nous devions alors travailler sur ces aspects du générateur de rapports avant la sortie de la nouvelle version de ce logiciel (la version 5.3). Certains ajouts à apporter à l'outil avaient déjà été assignés à l'équipe de développement de jUCMNav avant même que le processus de développement ne commence (ces changements étaient documentés à l'aide du logiciel de suivi de bogues Bugzilla de l'équipe de développement). Après avoir élicité la liste d'exigences pour notre projet (vers la deuxième semaine de septembre), d'autres nouvelles fonctionnalités (de même que certains bogues) ont étés ajoutés à Bugzilla, et nous avons ensuite pu commencer le développement de la nouvelle version du générateur de rapports pour jUCMNav.

2.1 Éléments ajoutés au contenu des rapports

Tel que mentionné dans l'introduction, les scénarios basés sur des diagrammes UCM sont partie intégrante des modèles URN, et sont présents dans l'outil jUCMNav depuis ses tout premiers débuts. Par contre, aucune documentation sur ces scénarios n'était présente dans les rapports générés par cet outil, puisque les concepteurs du prototype du générateur de rapports étaient très limités par le temps lors de l'implémentation de cette fonctionnalité de jUCMNav. Une de nos tâches consistait donc à ajouter de la documentation sur les différents éléments des scénarios UCM (points de départs/d'arrivées, initialisations de variables, etc.), et d'ajouter un sommaire de l'exécution de ces scénarios.

Nous avons donc commencé par créer une nouvelle section « Informations sur les scénarios » à la toute fin des rapports PDF et RTF générés par jUCMNav. Cette nouvelle section est générée par la classe *ReportScenarios*. La première partie de cette nouvelle section contient le sommaire d'exécution des scénarios UCM sous forme tabulaire. Pour récupérer l'information sur l'exécution de ces scénarios, nous avons utilisé une méthode semblable à la méthode *traverse(…)* de la classe *ScenarioUtils*. Cette méthode prend en paramètre un scénario UCM et l'exécute en traversant les éléments du diagramme UCM à partir duquel ce scénario a été créé. Notre méthode (nommée *traverseWarn(…))*, en plus de faire l'exécution du scénario en question, retourne la liste d'avertissements qui résulte de cette exécution. La méthode a une visibilité publique afin que notre classe (*ReportScenarios*) puisse l'utiliser. Le sommaire d'exécution des scénarios UCM est ensuite ajouté au rapport via la méthode *writeSummary()* de cette classe. Cette méthode crée une table pour chaque groupe de scénarios. Chacune de ces tables contient trois colonnes : le nom du scénario, le résultat de l'exécution de ce scénario (réussite ou échec) et les messages d'avertissements qui ont été lancés lors de l'exécution. Pour chaque scénario du groupe, une rangée est ajoutée à la table. Nous utilisons ensuite la méthode *traverseWarn(...)* pour obtenir la liste d'avertissements, tel qu'indiqué plus haut. Si aucun avertissement n'est lancé après l'exécution d'un scénario, cela signifie que l'exécution de ce dernier a été réussie; la couleur de fond de cette entrée dans la table est donc changée à vert et la colonne « Messages » demeure vide. Par contre, si l'exécution d'un scénario retourne un ou plusieurs message(s) d'avertissement(s), cela veut dire que l'exécution de ce scénario a échouée. La couleur de fond

de l'entrée dans la table est donc changée à rouge, et les messages d'avertissements obtenus lors de l'exécution sont inscrits dans la colonne « Messages ». Nous avions initialement l'intention de créer deux différentes catégories de messages, c'est-à-dire une pour les erreurs et une pour les avertissements. Cependant, tous les messages retournés par jUCMNav lors de l'exécution de scénarios ont le même niveau de priorité; il n'y a donc pas de distinction entre une erreur qui empêche l'exécution d'un scénario et un simple avertissement.

Avant de discuter de l'approche que nous avons utilisée afin d'incorporer les informations sur les scénarios dans les rapports HTML, il est important de préciser les éléments faisant partie de ce type de rapport. Premièrement, la page principale de chaque rapport HTML (*index.html*) consiste en deux cadres (ou *frames*) : un contenant la page principale du rapport, et un qui consiste en une liste de menus, regroupant chaque élément du modèle URN par catégorie. Le premier menu comprend des liens vers les diagrammes UCM du modèle, le deuxième comprend des liens vers les diagrammes GRL, et la dernière section comprend un lien vers la page titre du rapport, un lien vers la page des définitions UCM (variables, types d'énumérations, etc.) ainsi qu'un lien vers les définitions GRL (évaluations de stratégies). La page HTML correspondante à la liste de menus récupère tous les éléments de menus à l'aide d'un fichier XML représentant une structure d'arbre, où chacun des trois menus mentionnés précédemment est une « branche », et chacun des liens regroupés sous ces menus est une « feuille ». Si un digramme UCM donné possède plusieurs niveaux, la branche correspondante au menu « Diagrammes UCM » contiendra une branche correspondante au diagramme de premier niveau, et cette branche contiendra à son tour d'autres branches correspondantes aux autres niveaux, jusqu'à ce que le dernier niveau de diagrammes UCM soit atteint. Dans ce cas, les liens vers

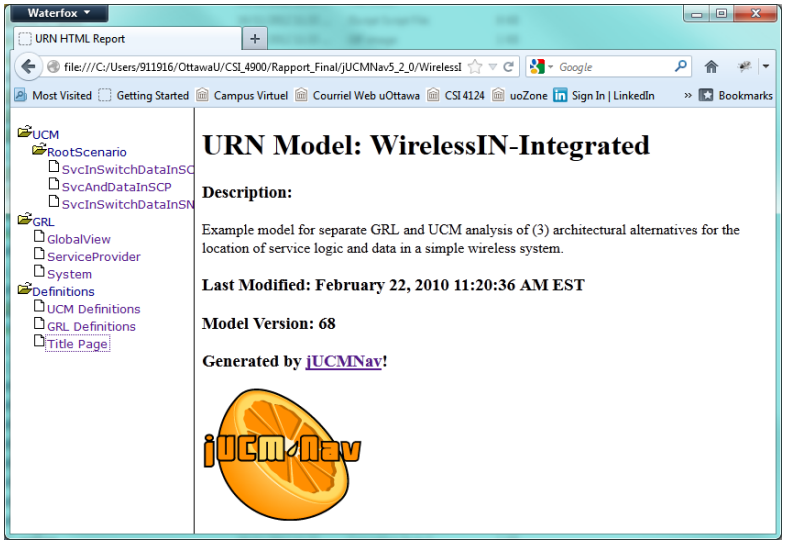

Figure 1 : Rapport HTML généré à l'aide du prototype de générateur de rapports de jUCMNav

ces diagrammes seront des feuilles. Pour afficher la page correspondante à un diagramme (ou une définition) dans le cadre principal du rapport, des fichiers Javascript sont incorporés aux fichiers du rapport, et contiennent des fonctions qui définissent le comportement du rapport lorsque l'utilisateur clique sur un lien donné via un des menus. Un fichier XSLT est aussi incorporé à cette liste de fichiers, et son rôle est d'afficher les icônes appropriées à côté de chaque nom de menu. Enfin, une page HTML est générée pour chaque diagramme faisant partie du modèle URN, et une page est générée pour chacun des autres éléments mentionnés plus tôt (page titre et pages de définitions du modèle).

Après avoir étudié la structure des rapports HTML, nous avons ajouté un nouveau menu à ceux-ci (sous la branche « Définitions ») pour accéder à une page HTML contenant l'information sur les éléments et l'exécution des scénarios. Ensuite, nous avons créé une nouvelle méthode dans la classe *HTMLReport* (la classe générant les rapports HTML), et cette méthode crée une nouvelle page Web représentant la documentation sur l'exécution des scénarios. L'information affichée sur cette page est très similaire à celle affichée dans les rapports PDF/RTF, puisque les nouvelles méthodes créées dans *HTMLReport* utilisent la même logique que les méthodes qui ont été créées dans la classe *ReportScenarios*. De plus, nous avons ajouté des hyperliens sur les noms des scénarios dans la table « sommaire d'exécution », ce qui fait en sorte que l'utilisateur est redirigé vers une ancre dans la section « Informations sur les scénarios », lui permettant ainsi de voir les détails du scénario sur lequel il a cliqué.

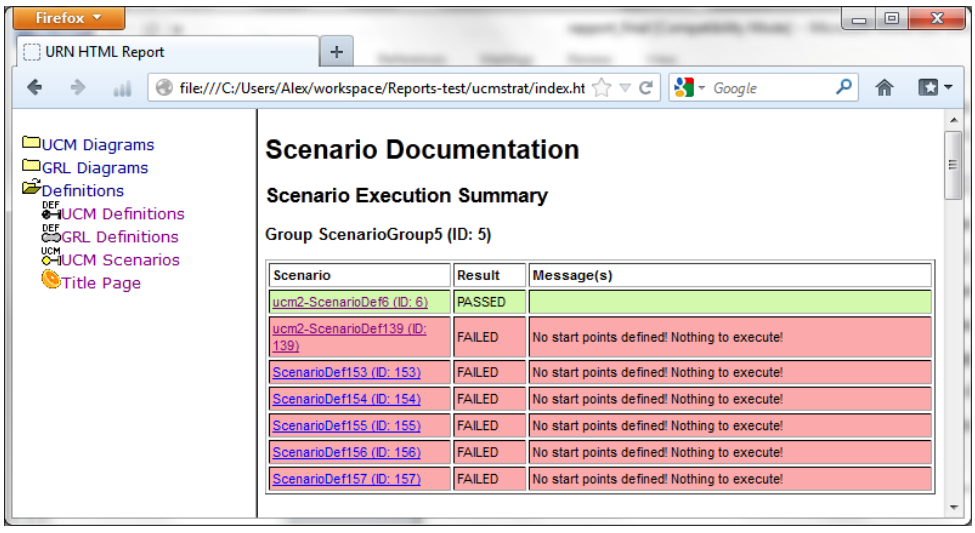

Figure 2 : Exemple d'un sommaire d'exécution de scénarios UCM dans un rapport HTML

Pour savoir où trouver l'information dont nous avions besoin pour accéder aux éléments des scénarios UCM, nous devions consulter le métamodèle URN utilisé pour construire l'outil jUCMNav. Ce métamodèle est en fait une collection de diagrammes de classes UML démontrant les attributs de chaque élément URN, ainsi que la relation entre ces éléments [3]. L'information sur les scénarios UCM était facilement accessible, puisque des méthodes *getters* avaient été implémentées pour chacun des éléments de ces scénarios. L'approche que nous avons adoptée pour inscrire les informations sur les scénarios est alors la suivante : pour chaque groupe de scénarios, boucler à travers chacun des scénarios contenus dans ce groupe et, pour chacun de ces scénarios, écrire toute l'information sur le scénario UCM en question. Cette information couvre les scénarios inclus dans ce scénario, les points de départs, les points d'arrivées, l'initialisation des variables, les préconditions ainsi que les postconditions de ce scénario. Nous avons donc implémenté une méthode dans la classe *ReportScenarios*, servant de point d'entrée, pour récupérer la liste des groupes de scénarios à l'aide de l'objet *UCMSpec* passé en paramètre. Ensuite, pour chacun de ces groupes, une autre méthode est appelée, et celle-ci ne fait que récupérer une liste de scénarios faisant partie du groupe de scénarios passé en paramètre. Cette méthode boucle ensuite à travers tous les scénarios de cette liste, et, pour chacun de ces scénarios, appelle une série de méthodes qui récupèrent toute l'information sur ce dernier. Enfin, l'information est inscrite dans le rapport généré (PDF ou RTF) au fur et à mesure que celle-ci est trouvée. Puisque la génération de rapports HTML est très différente de la génération de rapports PDF ou RTF, nous devions créer une série de méthodes dans la classe *HTMLReport*, afin d'effectuer les mêmes tâches que celles mentionnées ci-haut pour les rapports PDF/RTF. Nous avons donc gardé la même logique lors de l'implémentation de ces méthodes pour les rapports HTML, mais l'information était inscrite en utilisant HTML plutôt que l'éditeur de texte inclus dans le cadre d'application utilisé pour générer les rapports PDF/RTF.

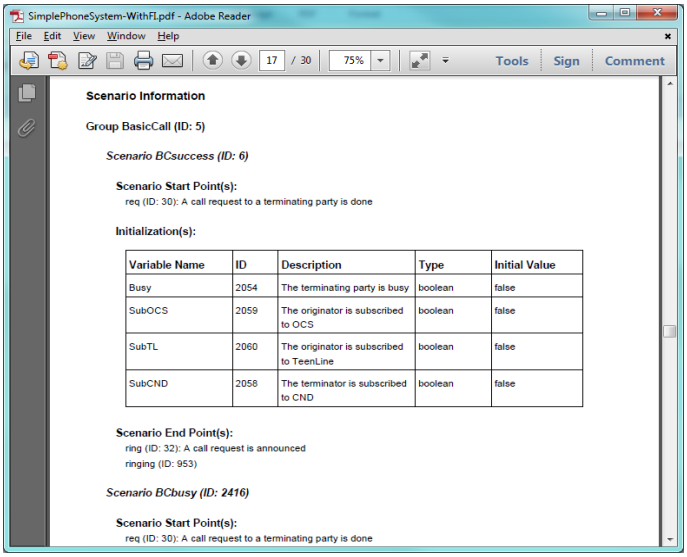

Figure 3 : Section « Informations sur les scénarios » dans un rapport en format PDF

Une autre nouvelle fonctionnalité que nous avons ajoutée aux rapports concerne la possibilité de visualiser des tendances dans le tableau de stratégies d'évaluations GRL. Pour chaque élément intentionnel (ou acteur) dans un modèle GRL, une tendance est calculée d'après les *x* dernières stratégies (*x* étant défini dans les préférences). Une flèche apparaît ensuite dans la dernière colonne du tableau, indiquant la tendance de l'acteur ou de l'élément intentionnel (cette tendance peut être soit positive, négative, neutre ou variable). S'il n'y a pas assez d'information disponible pour calculer la tendance, soit parce qu'il y a moins de stratégies que le nombre indiqué dans les préférences ou que ce nombre est plus petit que 2, un point d'interrogation est affiché. Pour calculer ces tendances, il a fallu premièrement trier la liste de stratégies en ordre alphabétique, puisque l'ordre a un effet sur la valeur de la tendance. Pour ce faire, nous avons créé une méthode *sortStrategies()* pour la classe *StrategiesGroup*. Cette méthode appelle la méthode *sort()* de l'objet *EList* qui contient les stratégies, afin de trier les éléments de la liste selon leur ordonnance naturelle. Il a donc aussi fallu définir l'ordonnance naturelle de l'objet *EvaluationStrategy*, en surchargeant sa méthode *compareTo(...)* pour comparer le nom d'une stratégie donnée avec le nom d'une autre stratégie. Une fois la liste triée, la méthode *calculateTrend(...)* (définie dans la classe *ReportStrategies*) calcule la tendance d'un élément (ou acteur) et retourne un nombre entier. Avec cette valeur, une autre méthode, *writeTrend(...)*, récupère l'image appropriée et ajoute la cellule de la tendance (contenant cette image) au tableau. La partie la plus difficile de l'implémentation de cette fonctionnalité a été l'utilisation d'un tableau avec des colonnes qui sont générées dynamiquement (pour un document PDF/RTF). Lorsqu'un tableau est plus petit que la largeur d'une page, le reste de la page est rempli par des cellules vides et sans bordures. Ainsi, après qu'une colonne pour chaque stratégie est ajoutée au tableau, une méthode est invoquée pour ajouter un certain nombre de cellules vides. En ajoutant une colonne de plus pour la tendance, le nombre de cellules vides doit naturellement changer. La logique existante a donc dû être modifiée pour prendre ce fait en considération. La logique pour le rapport HTML, bien que similaire, est beaucoup plus simple puisqu'on évite les difficultés reliées à la création d'un tableau dynamique en PDF/RTF.

	Strategy Evaluations			
	1	2	3	Trend
DBAdmin (A)	0	0	0	→
Prevent Unauthorized Disclosure	25	43	50	↗
Ensure Accountability of Data User	18	12	13	↗
Check Compliance to Legislation	100	100	100	→
Get Signed Agreement	0	75	100	↗
Verify Access Logs	75	-25	-50	↘

Figure 4 : Tableau d'évaluations des stratégies GRL

D’autres ajouts mineurs ont été faits à l’outil de génération de rapports au cours de la phase de développement du projet afin d’assurer une documentation plus complète des modèles URN dans les rapports générés par cet outil. L’un des éléments ayant été ajoutés aux rapports est la description des stratégies GRL dans la légende du tableau d’évaluations de stratégies. Auparavant, seul le nom des stratégies était affiché, et aucune autre documentation sur ces stratégies n’était visible dans les rapports générés par jUCMNav. En particulier, la valeur de la propriété « Description » des stratégies GRL (pouvant contenir des informations utiles sur le modèle URN en question) n’était affichée nulle part dans ces rapports. Nous avons donc inclus la description en plus du nom des stratégies dans la légende du tableau d’évaluation de stratégies GRL, lorsque cette information était disponible. Par ailleurs, le gabarit du fichier servant de page titre aux rapports HTML a été modifié afin d’inclure des informations importantes sur le modèle URN. Ces informations comprennent le nom et la description du modèle à partir duquel un rapport a été généré, le nom de l’auteur de ce modèle URN, les dates de création/modification de ce modèle ainsi que la date de génération du rapport. Afin de créer une page titre pour un rapport HTML donné, jUCMNav ne fait que copier le fichier gabarit dans le répertoire où le rapport HTML sera situé, pour ensuite ouvrir ce fichier. Ensuite, à l’aide d’expressions régulières, l’outil de génération de rapports trouve les champs où une information du modèle doit être inscrite, et remplace cette valeur par l’information en question. Les pages titre générées pour les rapports HTML contiennent dorénavant les mêmes informations que les pages titre des rapports en formats PDF et RTF. Finalement, un autre ajout que nous avons effectué aux rapports HTML fut l’ajout d’icônes plus représentatifs sur les menus latéraux. Ce nouvel ajout a non seulement permis aux rapports HTML d’être plus expressifs et consistants avec l’outil jUCMNav, mais a aussi grandement amélioré l’apparence générale de ces rapports. Nous avons remplacé les icônes qui étaient présentes sous le menu « Diagrammes UCM » par l’icône représentant un diagramme UCM dans jUCMNav. D’une façon similaire, nous avons remplacé les icônes présentes sous la section « Diagrammes GRL », de même que les icônes situées à côté des menus représentant la page titre du rapport, les définitions UCM/GRL ainsi que les scénarios UCM. Ceci a pu être fait à l’aide d’expressions XSLT, en définissant, pour chaque type de menu, un XPath valide qui conduit vers un nœud correspondant à ce type de menu dans le fichier XML du rapport. Une image (icône) est ensuite assignée au lien correspondant à ce menu. Les figures 5 et 6 montrent le résultat des deux derniers ajouts mentionnés ci-haut.

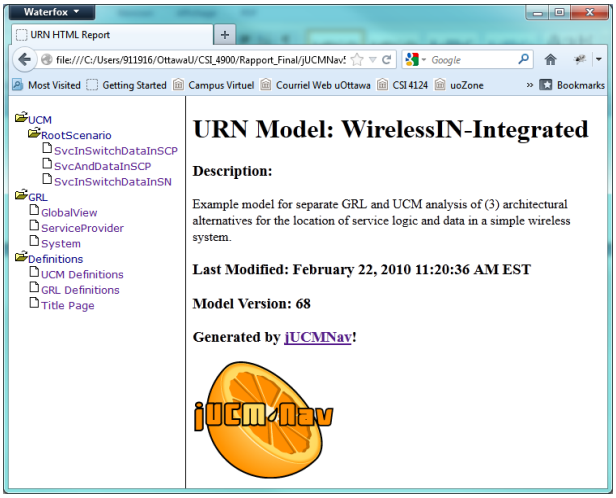

Figure 5 (ci-haut) : Rapport HTML généré avec jUCMNav 5.2

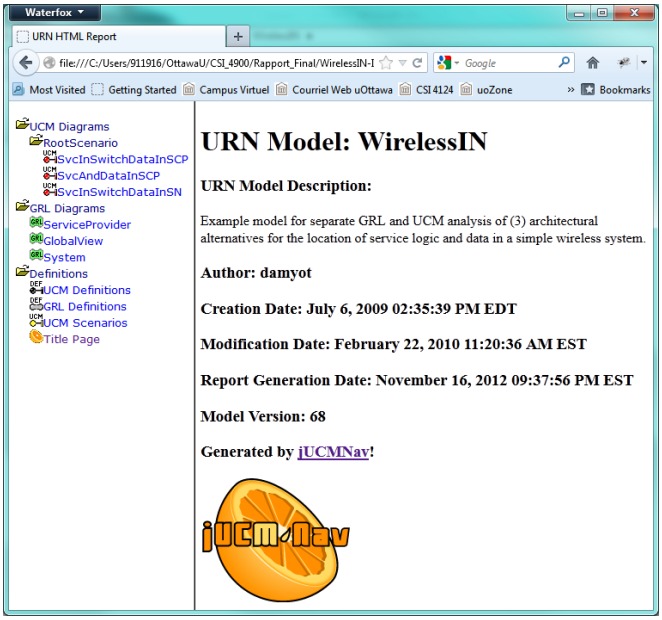

Figure 6 (à droite) : Rapport HTML généré avec jUCMNav 5.3

2.2 Personnalisation des rapports

L'un des aspects les plus importants du générateur de rapports de jUCMNav 5.3 est sans aucun doute la capacité de personnaliser les rapports. En effet, une bonne partie de ce projet a été consacrée à l'élaboration d'une nouvelle liste de préférences pour l'outil de génération de rapports ainsi qu'à la modification de l'outil-même pour rendre la structure de celui-ci moins rigide. Cela a été effectué dans l'unique but de rendre le générateur de rapports plus configurable.

L'élément le plus notable du nouvel outil de génération de rapports est sans doute le fait que ces rapports peuvent maintenant être générés dans plus d'une langue. En effet, les rapports peuvent dorénavant être généré dans n'importe quelle langue dans laquelle jUCMNav peut être exécuté. Pour ce faire, nous devions « externaliser » toutes les chaînes de caractères des packages *importexport.html* et *importexport.reports*. Externaliser une chaîne de caractères inscrite dans une classe signifie tout simplement d'ajouter cette chaîne à un fichier de propriétés faisant partie d'un projet Java, et de lui assigner une clé, afin de rendre la chaîne accessible de l'extérieur (chaque entrée de ce fichier de propriétés est une combinaison clé-valeur, où valeur est la chaîne en question). Puisque le fichier de propriétés est indépendant des classes Java, il est possible de créer un fichier de propriétés pour chaque langue dans laquelle nous voulons traduire

un programme Java. Il suffit simplement de créer ces fichiers, de copier le contenu du fichier de propriétés initial dans les nouveaux fichiers, et de traduire chaque chaîne de caractères présente dans ces nouveaux fichiers en la langue voulue. Les classes du programme Java en question peuvent ensuite récupérer ces chaînes de caractères à l'aide de l'appel <nom_du_fichier_de_propriétés>.*getString(*<nom_de_la_clé>*)*. La langue d'exécution du programme Java en question détermine le fichier de propriétés duquel nous devons récupérer les chaînes de caractères. Puisqu'un fichier de propriétés en français existait déjà dans le projet jUCMNav (le reste des fonctionnalités de l'outil était disponible en anglais et en français), nous devions simplement ajouter des nouvelles entrées dans les fichiers de propriétés en français et en anglais. Ceci a pu être effectué assez aisément à l'aide de l'outil d'externalisation des chaînes fournit avec Eclipse (accessible via le menu *Source -> Externalize Strings…*). En plus de spécifier toutes les chaînes à externaliser, nous devions aussi spécifier, à l'aide de l'outil mentionné précédemment, les chaînes à ignorer durant le processus d'externalisation des chaînes de caractères (telles que les balises HTML dans la classe *HTMLReport*). À la fin de la session, nous avions externalisé quelques 200 chaînes de caractères, et spécifié quelques centaines de chaînes à ignorer. Le générateur de rapports peut dorénavant générer des rapports en français ou en anglais, dépendamment de la langue dans laquelle jUCMNav est exécuté. De plus, puisqu'il est possible de créer plusieurs fichiers de propriétés au sein d'un même projet Java, il sera aussi possible à l'avenir de facilement traduire l'outil de génération de rapports (de même que tout le reste de jUCMNav) en plusieurs autres langues.

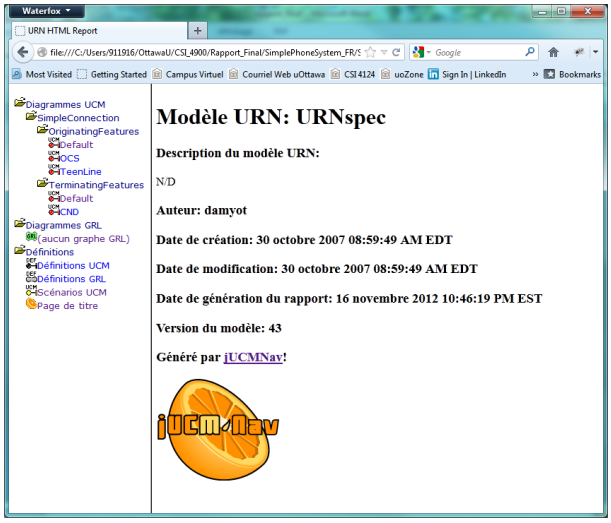

Figure 7 : Exemple d'un rapport HTML généré en français

Par ailleurs, nous avons aussi ajouté quelques préférences à la liste de préférences du générateur de rapports dans le but de donner l'option aux utilisateurs de jUCMNav de générer des rapports uniquement sur l'un des deux types de diagrammes d'un modèle URN (diagrammes UCM ou GRL). Il a été démontré par le passé que de nombreuses personnes qui utilisent

jUCMNav l'utilisent pour créer uniquement l'un des deux types de diagrammes (par exemple, une classe d'introduction au génie logiciel l'utilise uniquement pour créer des diagrammes UCM) [4]. Afin que les rapports ne contiennent aucune information non nécessaire à de tels utilisateurs, nous avons ajouté deux préférences à liste de préférences du générateur de rapports : « Afficher les diagrammes UCM » et « Afficher les diagrammes GRL ». Lorsque l'une de ces deux préférences n'est pas choisie, l'outil de génération de rapports ne tiendra pas compte de toute l'information reliée aux diagrammes dont le type correspond à celui de la préférence qui a été désactivée. Les valeurs de ces deux nouvelles préférences ont donc été importées dans les classes *Report* (pour les rapports PDF/RTF) et *HTMLReport* (pour les rapports HTML), et une instruction *if* a simplement été ajoutée à l'extérieur des appels aux fonctions dont la tâche consiste à inscrire les caractéristiques de l'un des deux types de diagrammes. Par exemple, si la préférence « Afficher les diagrammes GRL » est désactivée et qu'un rapport PDF est généré, alors la méthode *PDFReportDiagram.createPDFReportDiagramsAndDescription(…)* vérifiera si cette préférence est activée avant d'appeler les méthodes *insertDiagram(…)* et *createGRLDiagramDescription(…)* pour les diagrammes GRL faisant partie du modèle à partir duquel le rapport est généré. Donc, si cette préférence est désactivée, le rapport PDF résultant ne contiendra pas les sections sur les diagrammes GRL. De plus, la liste des diagrammes qui est présente dans l'assistant de génération de rapports (tel que montré à la Figure 13) a dû être filtrée afin de tenir compte de ces nouvelles préférences. Pour ce faire, la méthode *filterMapsToExport()* a été créée au sein de la classe *ReportWizardMapSelectionPage*, et cette méthode ne fait que retirer des diagrammes de la liste initiale si l'une des deux nouvelles préférences a été désactivée (et ne fait rien autrement). Puisque cette méthode est exécutée avant même que la fenêtre de l'assistant n'apparaisse, ce processus est transparent aux yeux des utilisateurs.

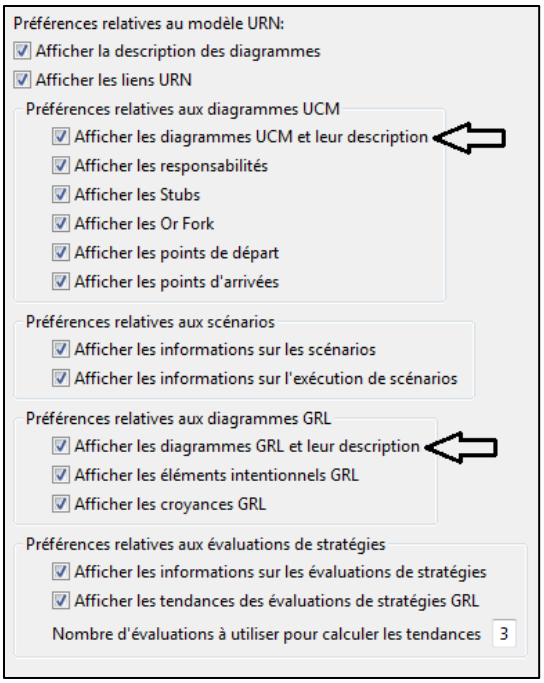

Figure 8 : Préférences du générateur de rapports relatives aux diagrammes UCM et GRL

Nous avons aussi ajouté des préférences spécifiques à d'autres éléments des modèles URN afin d'inclure ou d'éliminer certains de ces éléments lorsqu'un rapport est généré. Premièrement, nous avons ajouté deux préférences reliées aux scénarios UCM. Lorsque la préférence « Afficher les informations sur l'exécution de scénarios » est cochée, la méthode *writeSummary(…)* de la classe *ReportScenarios* (ou *HTMLReport*) est appelée, ce qui génère la table qui contient le sommaire d'exécution des scénarios. Par la suite, si la préférence « Afficher les informations sur les scénarios » est cochée, la méthode *writeUCMScenarioInformation(…)* de la classe *ReportScenarios*/*HTMLReport*, qui génère l'information détaillée des scénarios, est appelée. Cependant, tel qu'expliqué plus tôt, le tableau du sommaire d'exécution présent dans un rapport de type HTML comporte des hyperliens pour chacun des scénarios inscrit dans ce tableau, et ces hyperliens redirigent l'utilisateur vers les informations détaillées du scénario sur lequel il/elle clique. Afin d'éviter de voir ces liens lorsque la section sur les éléments des scénarios n'est pas incluse dans le rapport, les hyperliens sont ajoutés dans le sommaire seulement si la préférence « Afficher les informations sur les scénarios » est cochée. Par ailleurs, il y a aussi trois préférences spécifiques à l'évaluation des stratégies GRL. Ces préférences seront considérées seulement si le rapport comporte des diagrammes GRL. Premièrement, la table qui comprend l'évaluation des éléments intentionnels et des acteurs des stratégies GRL n'apparaîtra que si la préférence « Afficher les informations sur les évaluations de stratégies » est cochée. Par la suite, la colonne qui affiche la tendance des éléments n'apparaîtra que si la préférence « Afficher les tendances des évaluations de stratégies GRL » est cochée. Finalement, le champ « Nombre d'évaluations à utiliser pour calculer les tendances », aussi situé sur la page de préférences du générateur de rapports, change le calcul des tendances (pour prendre en compte plus ou moins d'évaluations) et peut donc modifier la tendance d'un élément GRL. La valeur spécifiée dans les préférences est donc incluse dans la légende des évaluations de stratégies, pour indiquer au lecteur du rapport le nombre d'évaluations sur lequel l'outil s'est basé pour calculer les tendances des stratégies GRL.

3. Bogues corrigés et problèmes rencontrés

En plus de l'ajout de plusieurs nouvelles fonctionnalités et de l'élaboration d'une nouvelle page de préférences pour l'outil de génération de rapports de jUCMNav, plusieurs défauts présents au sein de cet outil ont dû être corrigés. La grande majorité de ces bogues étaient souvent dus à la structure trop rigide du générateur de rapports. Les prochaines sous-sections discuteront des bogues les plus importants que nous avons trouvés, ainsi que de l'approche que nous avons adoptée afin de régler ces bogues.

3.1 Bogues reliés au contenu des rapports

Un des bogues les plus subtils de l'outil de génération de rapports était relié à la sauvegarde des rapports. Le générateur de rapport n'avertissait pas l'utilisateur lorsque celui-ci tentait de sauvegarder un rapport dans un répertoire contenant un fichier portant le même nom que ce rapport. jUCMNav ne faisait que remplacer le fichier existant, ou lançait un message d'erreur générique si ce dernier était ouvert par une autre application. Nous avons donc décidé de faire en sorte que le générateur de rapports avertisse l'utilisateur avant qu'il ne remplace un fichier existant, afin d'éviter d'effacer un fichier important. Pour ce faire, nous avons modifié la classe *ReportWizard* afin de lui ajouter deux variables booléennes : *fileExists* et *userChoice*. Avant d'exporter un modèle URN dans un rapport, le générateur de rapports vérifie si un fichier portant le même nom que le rapport existe déjà dans le répertoire spécifié. Si le type du rapport est HTML, on utilise plutôt « index.html » comme nom du fichier (la raison de ce choix sera expliquée en détails à la section 3.2). Si un tel fichier existe déjà, une boite de dialogue apparaît, avertissant l'utilisateur et lui demandant s'il veut remplacer le fichier existant. S'il choisit de le remplacer, *userChoice* est mis à vrai. Pour que l'outil de génération de rapports sache s'il doit exporter un modèle URN, l'outil vérifie les valeurs des variables *fileExists* et *userChoice*. L'outil va générer un rapport seulement lorsque *fileExists* est faux, ou que *userChoice* est vrai. De plus, afin d'éviter de fermer l'assistant de génération de rapport lorsque l'utilisateur choisi de ne pas remplacer le fichier existant, la méthode *performFinish()* de la classe *ReportWizard* retournera « *false* » si *fileExists* est vrai et *userChoice* est faux. Ce changement mineur rend le générateur de rapport beaucoup plus pratique à utiliser, et évite de perdre des fichiers importants.

Par ailleurs, nous avons remarqué que lorsqu'un rapport de dimension 0x0 était généré, une exception était lancée. Pourtant, les méthodes *setters* des préférences dans la classe *ReportGeneratorPreferences* semblaient traiter les valeurs invalides. Après avoir examiné le code et retracé l'exécution lorsqu'une préférence était modifiée, nous avons découvert que ces *setters* n'étaient en fait jamais utilisés. Nous avons observé que l'objet *StringFieldEditor* (la boîte de texte qui contient la valeur d'une préférence) possède une méthode *doStore(…)*, qui est automatiquement appelée lorsque l'utilisateur appuie sur le bouton *OK* de la page de préférences d'Eclipse. Cette méthode sauvegarde le contenu de la boite de texte dans la préférence correspondante. Les *setters* qui contiennent le code pour s'assurer de sauvegarder des valeurs valides ne sont donc jamais utilisés. Pour assurer la validité des préférences qui utilisent une boîte de texte, nous avons donc déplacé le code en charge de traiter les exceptions dans les méthodes *getters* de la classe *ReportGeneratorPreferences*. Ainsi, même si une valeur invalide est enregistrée dans une préférence, la méthode *getter* de cette préférence retournera une valeur valide lorsque cette préférence sera utilisée, ce qui évite des erreurs d'exécution.

Mis à part les bogues d'ordre technique, certains bogues reliés à la validité du contenu des rapports ont aussi dû être corrigés. Un des buts premiers de la nouvelle version du générateur de rapports était, bien entendu, la conformité des éléments inscrits dans les rapports avec la norme URN. En plus d'avoir des normes très strictes sur les éléments des modèles-mêmes, la norme internationale URN dicte que toutes les dates présentes dans un modèle URN doivent être inscrites dans le format AM/PM (par exemple, *December 1, 2012 9:00:00 PM EDT* en anglais, ou 1 décembre 2012 9:00:00 PM EDT en français) [5]. Par contre, nous avons remarqué que lorsque la langue de l'environnement Eclipse n'était pas en_XX (English Canada, English US ou English UK), toutes les dates du modèle étaient générées selon la langue dans laquelle Eclipse était exécuté. Puisque le format date/heure du générateur de rapports était fixe (celui-ci était défini dans les fichiers de propriétés) et qu'Eclipse utilisait souvent un système 24 heures pour enregistrer les dates, alors il arrivait quelquefois de voir des dates du genre « 5 November 2012 21:30:00 AM EDT » inscrites sur la page titre des rapports (dans les champs date de création/modification du modèle et date de génération du rapport). L'approche que nous avons utilisée pour réparer ce bogue a été la suivante : vérifier le format de la date analysée et convertir cette date en une chaîne de caractères, récupérer le format dans lequel nous voulons sauvegarder

la date dans le rapport (défini au préalable), et reconvertir la chaîne de caractères en une date en utilisant ce nouveau format. Après avoir effectué quelques tests en exécutant Eclipse en anglais et en français, nous avons pu générer des rapports contenant des dates conformes à la norme internationale URN.

3.2 Problèmes rencontrés lors de la génération des rapports en format HTML

De nombreux changements ont dû être apportés au prototype de l'outil de génération de rapports HTML, puisque ce dernier renfermait de nombreux bogues. Ces bogues étaient surtout causés par la structure plutôt rigide des fichiers de sortie générés par cet outil.

Le premier problème auquel nous avons fait face était un problème de compatibilité avec certains navigateurs Web. Afin de vérifier si les rapports HTML pouvaient être ouverts à l'aide des navigateurs Web les plus populaires, nous avons tenté de générer quelques rapports HTML et de les ouvrir à l'aide de divers navigateurs (Internet Explorer 8-10, Mozilla Firefox, Waterfox et Google Chrome). Les rapports s'affichaient tel que prévu sur tous ces navigateurs sauf Chrome. Lorsqu'un rapport était ouvert avec ce navigateur Web, seul le contenu du cadre principal était affiché. Cela faisait alors en sorte qu'il était impossible pour les utilisateurs de naviguer à travers des pages des rapports. Après avoir effectué quelques recherches sur ce problème, nous avons appris que ce problème était dû à un élément de sécurité de Chrome. Lorsqu'une page Web est ouverte localement sur Chrome (i.e. en utilisant le protocole file://), ce navigateur ne permet pas l'ouverture de d'autres fichiers locaux via cette page Web [6]. Afin de vérifier si cela était bel et bien le problème, nous avons créé un projet « Site Web » sur l'environnement de développement intégré Microsoft Visual Studio, et nous avons exécuté ce projet. Puisque MS Visual Studio démarre un serveur

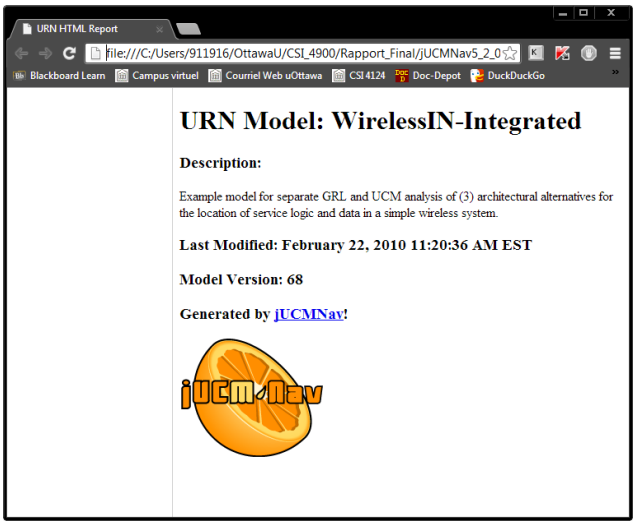

Figure 9 : Rapport HTML ouvert avec Chrome en utilisant le protocole file://

lorsqu'un tel projet est exécuté, alors le rapport s'ouvrait en utilisant le protocole http://. Tel que montré à la Figure 10, cette approche a fonctionné et nous avons pu confirmer ce que nous avions trouvé en effectuant nos recherches. Puisque la très grande majorité des rapports HTML générés par jUCMNav seront localisés sur des serveurs Web, et que Google Chrome ne détient qu'une très faible part du marché des navigateurs Web, nous en avons convenu qu'il ne valait pas la peine de changer la structure des rapports uniquement pour que ceux-ci puissent être ouverts localement à l'aide du navigateur Google Chrome [4]. Nous avons ensuite pris note de cette situation pour l'ajouter à la documentation de la nouvelle version de jUCMNav (présente sur le site Foswiki de jUCMNav).

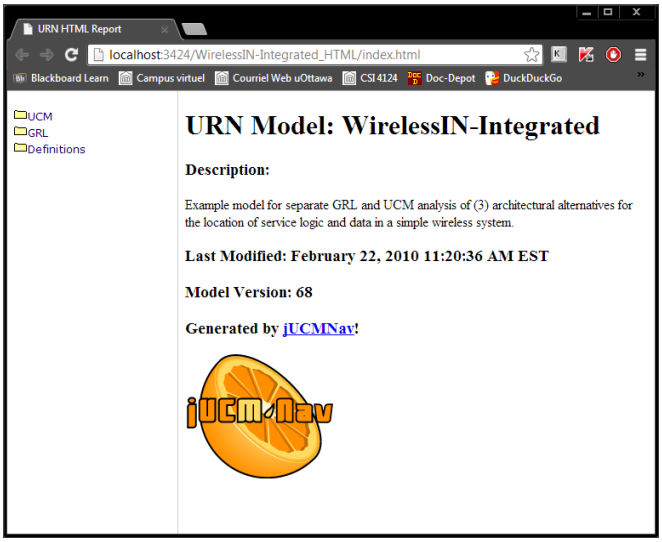

Figure 10 : Rapport HTML ouvert avec Chrome en utilisant le protocole http://

Après s'être familiarisé avec les rapports, nous avons remarqué quelques problèmes avec le comportement du menu latéral des rapports HTML. Lorsque les rapports étaient ouvert avec le navigateur Firefox et qu'on cliquait sur l'une des branches du menu (tel que montré à la Figure 11), le menu latéral apparaissait dans le cadre principal du rapport. Par ailleurs, lorsque ce rapport était ouvert avec Internet Explorer (tel que montré à la Figure 12), c'était le contenu du répertoire qui contient le rapport qui était affiché dans le cadre principal. Après avoir examiné les documents HTML, XML et Javascript du rapport, nous avons pu identifier le problème. Lorsque l'on appuyait sur un élément du menu latéral, la fonction Javascript

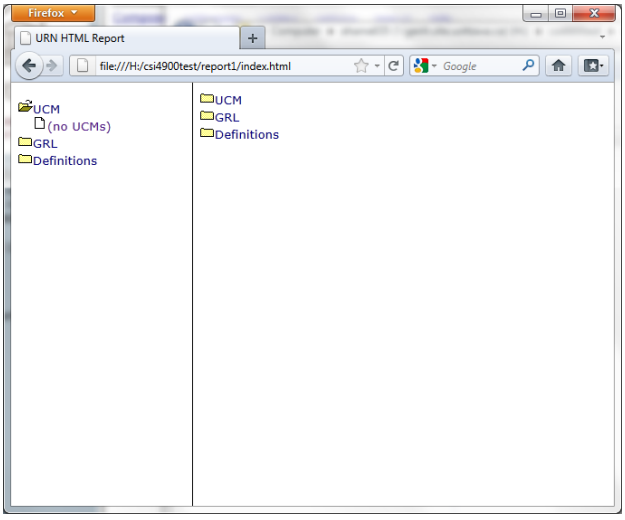

Figure 11 : Ouverture d'un rapport HTML avec Mozilla Firefox

redirect(…) était appelée. Cette fonction prend en paramètre la valeur de la variable *branchLink*, et cette variable contient la valeur spéciale « *notRedirect* » lorsque l'on clique sur une branche. Lorsque la fonction reçoit cette valeur, elle ne fait rien. Si elle obtient toute autre valeur, elle tente d'ouvrir le lien correspondant à la valeur de ce paramètre dans le cadre principal du rapport. Le problème avait lieu dans le fichier *xmlTree.js*. Alors que le fichier *tree.xml* fixait l'attribut *branchLink* des nœuds XML, le fichier Javascript envoyait la valeur *branchlink* (avec un « l » minuscule) à la fonction *redirect(…)*. Cette faute de frappe signifiait qu'un paramètre différent était utilisé (un paramètre qui n'avait aucune valeur). Une valeur vide était donc envoyée à la fonction et, puisque la valeur vide n'était pas égale à « *notRedirect* », cette fonction tentait d'ouvrir le lien. Le problème que nous voyions au départ était simplement les différentes façons que les navigateurs Web utilisaient pour gérer des liens vides. Après avoir corrigé cette faute de frappe, nous pouvions voir le comportement attendu du rapport HTML lorsqu'on sélectionnait l'une des branches du menu latéral (c'est-à-dire que le contenu du cadre principal ne changeait pas).

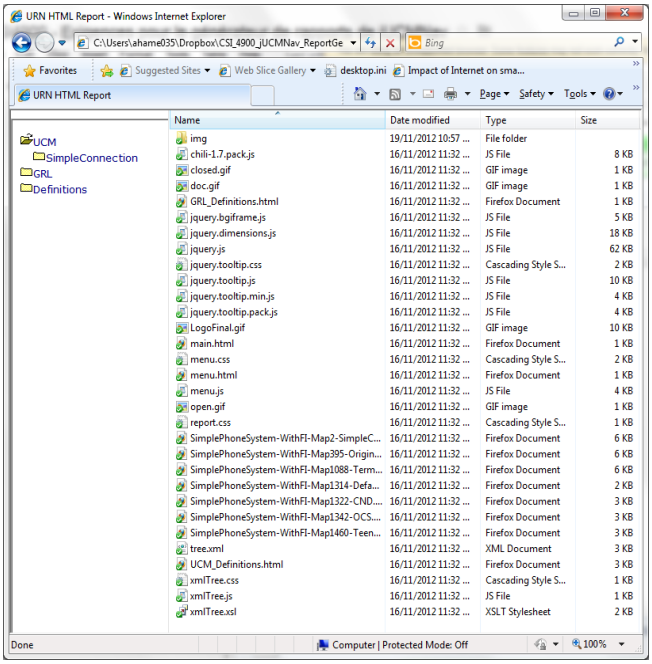

Figure 12 : Ouverture d'un rapport HTML avec Internet Explorer

Par ailleurs, nous devions aussi modifier l'un des paramètres dans l'assistant de génération de rapports de jUCMNav lorsque le type de rapport sélectionné était HTML. Contrairement aux deux autres types de rapports (PDF et RTF), le générateur de rapports ne considère pas le nom de rapport (appelé « préfixe de rapport » dans l'assistant) entré par l'utilisateur lorsqu'un rapport est généré en format HTML. Ceci est dû à deux facteurs : le nombre de fichiers générés et le nom de fichier par défaut du fichier principal. Puisqu'un rapport HTML typique contient quelques dizaines de fichiers (tel que mentionné à la section 2), et que plusieurs de ces fichiers sont interdépendants et générés à partir de gabarits (*templates*) de fichiers, le fait de renommer tous ces fichiers ferait en sorte que plusieurs références seraient

perdues lors de la génération des rapports. Aussi, nous ne voulions pas changer le nom du fichier principal (*index.html*), puisqu'il s'agit de la page HTML par défaut affichée par un serveur Web lorsqu'un utilisateur accède au répertoire où se trouve le rapport. Le fait d'assigner un autre nom à ce fichier ferait alors en sorte que cette page par défaut serait introuvable et que l'utilisateur voulant accéder au rapport via un site Web devra spécifier le nom exact du fichier HTML principal du rapport. Ce bogue peut sembler banal, mais si cette option était disponible pour les rapports HTML, cela porterait à confusion (les utilisateurs pourraient penser que le fichier principal du rapport généré porte le nom qu'ils avaient spécifié). Pour remédier à cela, nous avons dû modifier le gestionnaire de l'événement *SelectionEvent* de la classe représentant l'assistant de génération de rapports, afin de détecter le moment où le type « Rapport HTML » est choisi par l'utilisateur (le choix du type de rapport est effectué via une liste déroulante). Dans la définition de ce gestionnaire d'événement, nous avons simplement ajouté une instruction *if* vérifiant si le type de rapport choisi est HTML. Si tel est le cas, alors la boîte de texte dans laquelle le nom du rapport peut être spécifié est désactivée.

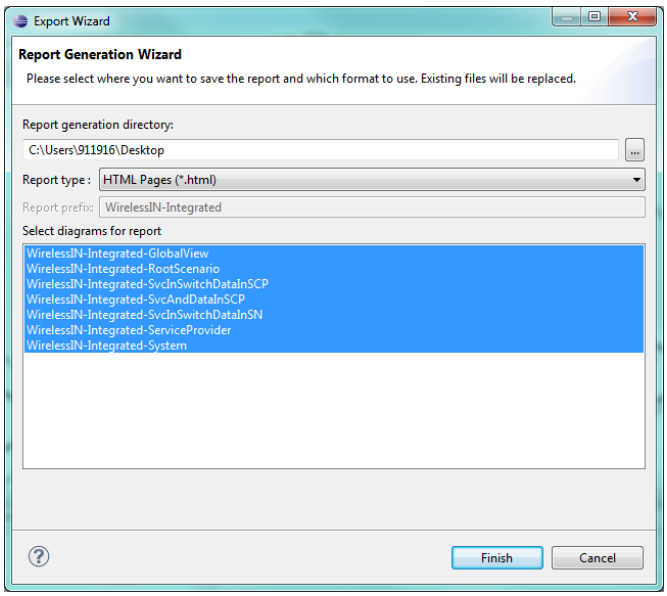

Figure 13 : Assistant de génération de rapports, lorsque le type de rapport choisi est HTML

Enfin, d'autres bogues mineurs présents dans l'outil de génération de rapports HTML ont aussi pu être corrigés lors du processus de développement. Entre autres, nous avons dû modifier la classe *HTMLReport*, afin que celle-ci génère une page HTML même si le modèle à partir duquel un rapport est généré ne contient seulement qu'un des deux types de diagrammes. Jusqu'à la version 5.2, l'outil ne générait pas de pages HTML pour le type de diagramme manquant, ce qui faisait en sorte que le navigateur Web sur lequel un rapport HTML était consulté lançait des erreurs de type 404 : page non trouvée. Par exemple, si aucun diagramme GRL n'était présent dans un modèle URN donné et qu'un rapport HTML était généré, le cadre contenant les différents menus affichait quand même une branche pour les diagrammes GRL, et cette branche

contenait une feuille avec le nom par défaut « *no GRLs* ». Il était possible pour les utilisateurs de cliquer sur ce lien, mais, puisqu'il n'y avait aucune page qui était générée pour les diagrammes GRL, une erreur de type 404 était lancée. Puisque la structure du fichier XML utilisé pour construire le menu avait une structure très rigide (ce fichier était déjà présent dans le projet jUCMNav et était simplement copié dans le dossier où le rapport HTML était généré), nous ne pouvions pas enlever ou modifier certaines branches des menus. Pour remédier à la situation, nous avons décidé de créer une page HTML contenant un message indiquant qu'aucun diagramme UCM/GRL n'est présent dans le modèle, et d'ajouter cette page au rapport lorsqu'aucune information sur l'un des deux types de diagrammes n'est trouvée. Donc, lorsqu'un utilisateur clique sur une « feuille » contenant le message par défaut « *no GRLs* » (ou « *no UCMs* »), cet utilisateur était tout simplement redirigé vers cette page HTML. Après avoir réglé ce problème, nous avons aussi dû modifier les liens vers la page Foswiki du projet jUCMNav. Les liens vers cette page étaient brisés depuis quelques mois déjà, car le site Web du projet jUCMNav avait migré vers une nouvelle plateforme, et les liens vers ce site Web n'avaient pas été mis à jour (ceux-ci étaient inscrits dans les gabarits de fichiers utilisés pour générer les rapports HTML). Nous avons alors effectué les changements nécessaires à ces fichiers *templates*, et nous avons ensuite pu appliquer ces changements au serveur SVN du projet jUCMNav.

3.3 Bogues corrigés lors de la génération des rapports en format PDF/RTF

Après avoir résolu un grand nombre de bogues reliés au générateur de rapports HTML, nous devions corriger quelques défectuosités de l'outil de génération de rapports PDF/RTF. Avant de discuter de ces bogues et de la façon dont nous les avons résolus, il est important de préciser que les rapports RTF ont été beaucoup moins utilisés que les rapports en format PDF, et donc certains bogues dans le passé ont été corrigés pour le générateur de rapports PDF, mais ces changements n'ont souvent pas été appliqués aux portions de code spécifiques à la génération de rapports RTF (même si les classes correspondantes aux deux outils utilisent la même librairie). Nous avions donc un peu plus de résolution de bogues à faire du côté de l'outil de génération de rapports RTF que du côté de l'outil de génération de rapports PDF.

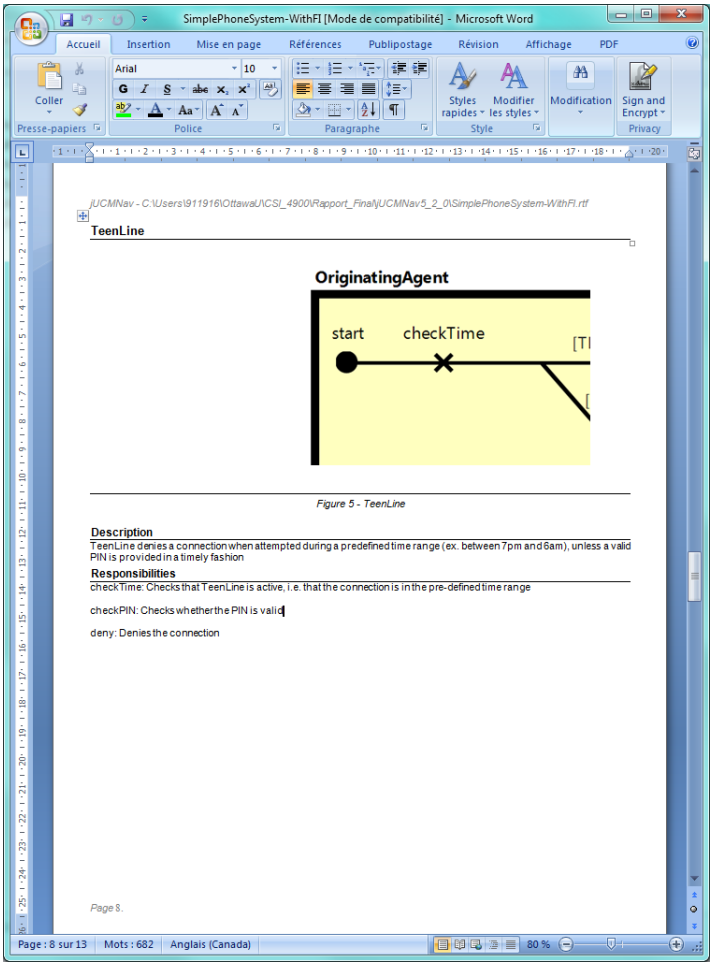

Figure 14 : Exemple de figure dans un rapport RTF

Premièrement, un des bogues les plus évidents du générateur de rapports RTF était la grandeur des figures dans les rapports générés. Presque toutes les figures étaient soient beaucoup trop grandes, soient beaucoup trop petites, et donc il n'était souvent pas possible de voir un diagramme UCM/GRL dans son ensemble. Nous avons donc jeté un coup d'œil à la classe *ReportUtils*, une classe qui, comme son nom l'indique, fournit un ensemble de fonctions utilisées lors de la génération de rapports. Nous avons remarqué que cette classe renfermait plusieurs méthodes permettant de bien redimensionner une image pour un rapport, ainsi que deux méthodes utilisées pour insérer une figure dans un rapport (une pour les rapports PDF, et l'autre pour les rapports RTF). Après avoir suivi l'exécution de la génération d'un rapport en format RTF, nous avons constaté que la méthode insérant une image dans un rapport RTF (*insertRTFImage(…)*) effectuait essentiellement les mêmes tâches que celle insérant une image dans un rapport PDF (*insertImage(…)*), à deux exceptions près : *insertRTFImage(…)* appliquait deux transformations de plus que *insertImage(…)* sur une figure (la figure était agrandie d'un facteur de deux, et était translatée, avant de subir les mêmes transformations que celles effectuées par *insertImage(…)*). Nous avons donc retiré de la méthode *insertRTFImage(…)* ces transformations non nécessaires, et nous avons ajusté le facteur d'agrandissement des images à 1.0 (100%). Après avoir appliqué ces quelques changements à la classe *ReportUtils*, les images insérées dans les rapports RTF n'étaient pas plus larges que l'espace entre les marges de ces rapports (définie à l'aide des préférences du générateur de rapports).

Un autre bogue assez important relié à la génération de rapports PDF et RTF fut la gestion des pages lorsque l'information était inscrite dans des rapports générés dans l'un ou l'autre de ces formats. Contrairement à une page HTML, où la longueur d'une page est infinie, la longueur d'une page de rapport en PDF/RTF est fixe. Donc, lors de la génération d'un rapport, l'application écrivant dans celui-ci doit s'assurer que certaines informations (une figure, par exemple) ne soient pas réparties sur plus d'une page. Il est donc important de faire un saut de page après chaque section du rapport. Par contre, le fait d'ajouter trop de sauts de page peut causer un effet visuel tout aussi désagréable. Cette dernière situation arrivait très souvent dans l'outil de génération de rapports de jUCMNav, car un saut de page était inséré avant d'inscrire et après avoir inscrit les informations reliées à un diagramme. Ceci faisait alors en sorte qu'une page blanche s'insérait dans le rapport entre la fin de la description d'un diagramme du modèle et le début de la description du prochain élément du modèle URN en question. Cela avait alors comme conséquence d'augmenter considérablement la taille du rapport. Afin d'insérer un nombre raisonnable de sauts de page dans un rapport en format PDF ou RTF, nous avons déplacé l'instruction ajoutant ce saut de page à des endroits spécifiques dans les méthodes générant ces types de rapports. Des sauts de page ont ainsi été ajoutés à la fin de la page de titre des rapports, à la fin de la section « dictionnaire de données », ainsi qu'au début de chaque section correspondante à la description d'un diagramme.

4. Mise à jour de la documentation reliée à jUCMNav

Vu la complexité du plugin jUCMNav et le nombre élevé de développeurs qui participent à son développement, la documentation des nouvelles fonctionnalités et des bogues est importante pour les développeurs futurs. En plus de la documentation et des commentaires ajoutés au code afin d'expliquer la logique derrière les nouvelles classes/méthodes implémentées ainsi que tous les autres changements effectués, nous avons mis à jour l'aide en ligne sur le site Foswiki du projet jUCMNav. La section du générateur de rapports étant désuète, nous l'avons modifiée pour décrire brièvement les nouvelles fonctionnalités et sections des rapports, et nous avons mis à jour les images qui étaient insérées dans le texte. Par la suite, nous avons exécuté le script qui synchronise l'aide intégrée au plugin avec la page d'aide en ligne. Nous avons aussi créé quelques courts vidéos avec l'application Wink, dans le but de mettre en valeur les

nouvelles fonctionnalités de l'outil de génération de rapports. Ces vidéos servent aussi de tutoriels sur la génération de rapports, le changement des préférences du générateur de rapports et la navigation à travers un rapport. Finalement, chaque ajout/bogue sur lequel nous avons travaillé tout au long de la session a été ajouté au site Bugzilla de l'équipe de développement de jUCMNav (s'il n'y était pas déjà), et ce dans le but de pouvoir retracer les changements au code. Lorsque nous modifions le code pour ajouter une nouvelle fonctionnalité ou réparer un bogue, nous faisions référence au numéro de l'item Bugzilla correspondant à ce bogue (ou fonctionnalité) dans le commentaire du commit sur le serveur SVN du projet afin de bien identifier les changements qui ont été faits.

5. Connaissances acquises et développement futur

Le projet sur lequel nous avons travaillé tout au long de la session d'automne 2012 nous a permis d'appliquer des connaissances que nous avions acquises durant notre cheminement à l'Université d'Ottawa, et nous a permis d'en acquérir d'autres. Tel que décrit plus loin, l'élaboration d'une nouvelle version de l'outil de génération de rapports de jUCMNav a été un projet qui nous a donné l'opportunité d'en connaître davantage sur le processus de développement et de maintenance d'un logiciel.

5.1 Connaissances acquises et leçons tirées du développement

Ce projet de développement nous a permis de mettre en pratique les connaissances théoriques que nous avions acquises en classe. Par contre, en tant qu'étudiants en informatique, nous n'avions vu le processus de développement logiciel que brièvement. Ce projet nous a permis d'acquérir de l'expérience reliée au processus de développement logiciel, et nous a aidés à comprendre l'importance de chacune des étapes de ce processus, en commençant par l'élaboration d'un document d'exigences. Même si le projet jUCMNav était déjà mis en place, la définition d'exigences en lien avec le générateur de rapports (élaborées au tout début de notre projet) a été essentielle afin de clairement identifier les nouvelles fonctionnalités qui devaient être ajoutées à cet outil. Cette première étape du processus de développement nous a aussi permis de choisir un nombre raisonnable de nouveaux éléments à ajouter au générateur de rapports pour le temps qui nous était alloué. Même si nous savions que notre liste d'exigences changerait en

cours de route, nous avons été surpris du nombre d'exigences qui ont été ajoutées, révisées ou enlevées au cours du processus de développement (voir les annexes I et II).

Après avoir élaboré une première version de la liste d'exigences pour notre projet, nous devions nous familiariser avec le code et les standards existants, et ce, avant même de commencer à ajouter et modifier du code. Ceci a souligné l'importance d'une documentation claire et concise, et nous a encouragés à bien documenter tous les changements que nous avons apportés à l'outil de génération de rapports. La documentation de toutes les nouvelles fonctionnalités de cet outil facilitera la maintenance du code lors du développement des prochaines versions de jUCMNav. Une fois implémentées, ces nouvelles fonctionnalités étaient tout de suite testées afin de s'assurer de leur bon fonctionnement et afin de vérifier que ces changements à l'outil n'affectaient pas les fonctionnalités existantes. Nous avons consacré une bonne partie du temps au débogage des nouvelles fonctionnalités ainsi qu'au débogage du code existant, où nous avions trouvé quelques erreurs. Ceci souligne encore une fois l'importance d'une bonne documentation. Le fait d'avoir trouvé et documenté les erreurs trouvées dans le code fera en sorte que les développeurs qui trouveront des comportements inattendus en lien avec les nouvelles fonctionnalités que nous avons ajoutées puissent les corriger plus facilement. Ce projet nous a donc permis de voir à quel point l'implémentation n'est qu'une petite partie du processus de développement logiciel.

Par ailleurs, puisque jUCMNav est un outil permettant de modeler les exigences des utilisateurs, nous avons aussi acquis des connaissances reliées à un autre aspect du processus de développement logiciel : la modélisation des exigences en utilisant la norme URN. Encore une fois, puisque nous sommes étudiants en informatique, nous n'avons pas étudié ce sujet à fond. En effet, le seul aspect de la modélisation des exigences que nous avions étudié dans le passé est l'élaboration de diagrammes de séquences permettant de définir le comportement d'un logiciel. Nous avons donc dû étudier la norme URN ainsi que tous les éléments des modèles basés sur cette norme (à savoir, les diagrammes de cas-types, les scénarios UCM, les contraintes OCL, les diagrammes GRL, les évaluations de stratégies GRL et les indicateurs de performance). Nous avons eu la chance de consulter divers documents sur la norme internationale URN avant et pendant le développement de l'outil de génération de rapports pour jUCMNav (nous avons pu

étudier ces documents à partir de la fin du mois de juillet 2012). De plus, le métamodèle qui a été élaboré à partir de cette norme nous a grandement aidés à visualiser les attributs des éléments URN, et nous a aussi aidés à mieux comprendre les relations entre ces éléments (par exemple, la relation entre un objet *Actor* et un objet *GRLSpec*, ou encore la relation entre un objet *StartPoint* et un objet *UCMSpec*).

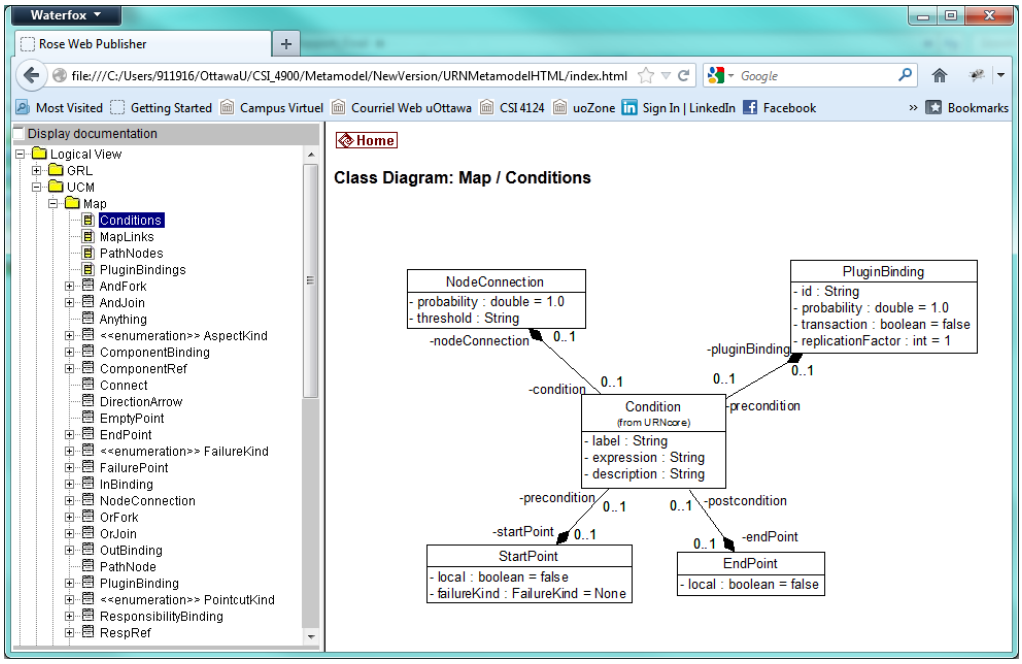

Figure 15 : Extrait du métamodèle URN utilisé lors de l'implémentation de jUCMNav

5.2 Éléments à considérer lors du développement futur

Puisque nous n'avions que trois mois pour implémenter la nouvelle version de l'outil de génération de rapports de jUCMNav, certains éléments des modèles URN n'ont pas pu être ajoutés au contenu des rapports. Entre autres, les contraintes OCL ainsi que les indicateurs de performance (ou *Key Performance Indicators*, en anglais) ne sont pas inclus dans les rapports générés par jUCMNav 5.3. Les contraintes OCL (ou *Object Constraint Language*) permettent à l'utilisateur de spécifier des règles pour les diagrammes UCM, et ces règles complémentent les règles syntaxiques existantes dans ces diagrammes [4], [7]. Une section sur ces contraintes devrait être ajoutée aux rapports, et cette section devrait montrer les résultats de l'évaluation des règles OCL, afin que les lecteurs des rapports puissent voir si les diagrammes UCM définis au sein d'un modèle respectent ces règles. De plus, comme pour la plupart des nouvelles sections

que nous avons ajoutées aux rapports, les utilisateurs de jUCMNav devraient pouvoir décider si les rapports générés doivent inclure une section sur les contraintes OCL ou non (via les préférences du générateur de rapports). Par ailleurs, les indicateurs de performance sont une extension récemment standardisée des diagrammes GRL, et permettent l'évaluation des performances dans un processus opérationnel [2], [4]. Une autre section dans les rapports pourrait être ajoutée afin d'inclure ces indicateurs KPI. Encore une fois, les utilisateurs de jUCMNav devraient être en mesure de décider si oui ou non cette section doit être ajoutée au contenu des rapports. Afin que ces deux additions à l'outil de génération de rapports puissent être considérées lors du développement des prochaines versions de jUCMNav, deux items ont été ajoutés à Bugzilla.

Certains éléments que nous voulions ajouter au départ à l'outil de génération de rapports se sont avérés très complexes à implémenter, ce qui nous a poussés à changer nos plans. Lors de l'implémentation de la section sur l'exécution des scénarios UCM, nous avons exploré la possibilité d'inclure des diagrammes de séquence. Ces éléments visuels offriraient une vue plus linéaire de l'exécution des scénarios, et seraient une addition intéressante aux rapports. Cependant, nous avons constaté que l'ajout de tels diagrammes n'est pas trivial. Les diagrammes de séquence qui sont affichés dans jUCMNav sont en fait générés par un autre plugin d'Eclipse; il faudrait donc inclure tout le code de cet autre projet dans le projet jUCMNav. Vu la complexité de cette modification, nous avons opté pour l'affichage sous forme tabulaire des messages qui résultent de l'exécution des scénarios UCM. Par contre, l'affichage des diagrammes de séquence dans les rapports est un élément à considérer dans le cadre d'un projet futur.

Un autre élément à considérer pour les versions futures de jUCMNav concerne l'implémentation de tests automatisés pour l'outil de génération de rapports. Comme nous l'avons vu dans certains cours d'informatique et de génie logiciel, l'utilisation de tests automatisés est très utile lors du développement d'un logiciel, car ces tests permettent de s'assurer que les modifications qui sont apportées à ce logiciel n'affectent pas les anciennes fonctionnalités de ce dernier. Présentement, aucun test n'existe pour le générateur de rapports; les seuls tests automatisés qui existent au sein du projet jUCMNav sont ceux qui testent le bon fonctionnement des diverses commandes des éditeurs de diagrammes. Cependant, dans le cadre

de notre projet, nous avons jugés que l'implémentation de tests automatisés pour l'outil de génération de rapports n'aurait pas été très utile pour l'effort requis. Il aurait été tout à fait possible d'écrire un test en utilisant JUnit pour vérifier qu'un rapport est bel et bien créé. Pour ce faire, il aurait fallu inclure un fichier .jucm dans le répertoire où les tests sont situés, en retirer l'instance de l'objet *URNspec* avec toute la collection de diagrammes qu'il contient (qui sont des objets créés à l'aide d'une classe implémentant l'interface *IURNDiagram*), puis exécuter la méthode *export(...)* de la classe *PDFReport*, *RTFReport* ou *HTMLReport*, avec un nom de rapport et un nom de répertoire. Il aurait ensuite fallu vérifier que le document existe dans le répertoire spécifié. Cependant, le (ou les) document(s) de sortie est/sont toujours créé(s), même lorsqu'une erreur a lieu lors de l'exportation du rapport. Ceci est dû au fait que la création du (ou des) fichier(s) de sortie est la toute première étape de la génération d'un rapport avec jUCMNav. Donc, même si le fichier résultant de l'exécution du test est corrompu (et donc illisible), ce fichier aurait été quand même présent dans le répertoire spécifié. Une telle situation rendrait donc le test valide et, dû à cela, nous avons conclu qu'un tel test n'aurait pas été très utile. D'un autre côté, créer un test qui vérifierait le contenu d'un rapport aurait été beaucoup plus difficile (et coûteux) à implémenter. Pour analyser un rapport en format HTML, il aurait fallu reconstruire l'arbre des éléments du rapport et valider le contenu de chaque section de celui-ci. En plus de cela, il aurait fallu s'assurer que le test soit écrit d'une telle façon que la validation ne serait pas affectée par l'ajout de nouveaux éléments dans le contenu des rapports. En ce qui concerne la validité du contenu des rapports en format PDF et en format RTF, il aurait fallu utiliser un outil supplémentaire pour pouvoir lire le document, ce qui aurait ajouté une autre dépendance externe. De plus, puisque la classe *HTMLReport* est une classe indépendante des deux autres (*PDFReport* et *RTFReport*), le fait de valider uniquement le contenu d'un rapport HTML n'aurait pas été suffisant. Vu la difficulté et le temps qui nous était alloué, il était plus simple et plus sûr de faire les tests manuellement lors de l'implémentation de la nouvelle version de l'outil de génération de rapports, mais la mise en place de tests automatisés pour cet outil pourrait être un projet à considérer lors de la prochaine mise à jour de jUCMNav.

6. Conclusion

Au cours de la session d'automne 2012, nous avons grandement amélioré le générateur de rapports de jUCMNav en identifiant et en corrigeant des bogues, en ajoutant des éléments URN au contenu des rapports et en rendant ces derniers beaucoup plus configurables. La version 5.3 de jUCMNav permettra aux utilisateurs d'inclure les éléments dont ils ont besoin dans les rapports, et d'enlever les éléments non nécessaires qui encombreraient ces rapports. Cette nouvelle version inclut aussi des nouveaux éléments, tels que l'exécution des scénarios UCM et la visualisation des tendances GRL, qui offrent une vue plus complète des diagrammes créés avec l'application. De plus, l'internationalisation des rapports augmente la portée des rapports et rend l'outil plus intéressant pour certains utilisateurs, tels que les agences gouvernementales et certaines personnes vivant à l'extérieur du Canada. Plusieurs changements esthétiques ont aussi été appliqués aux rapports afin de les rendre plus agréable à lire et à utiliser. En raison du temps limité pour le projet, nous n'avons pas pu ajouter tous les éléments que nous aurions voulus. Toutefois, le générateur de rapport de la version 5.3 est beaucoup plus raffiné qu'auparavant. De plus, ce projet nous a permis d'acquérir des connaissances pratiques et de l'expérience dans le domaine du développement logiciel, ce qui complémentera les connaissances que nous avons acquises en classe durant notre séjour à l'Université d'Ottawa. Ces connaissances seront sans aucun doute très utiles lorsque nous arriverons sur le marché du travail.

Références

[1] ProjetSEG Web (sur Foswiki), « jUCMNav Online Help », novembre 2012 [En ligne]. Disponible : http://jucmnav.softwareengineering.ca/ucm/bin/view/ProjetSEG/HelpOnLine. [Accédé : 17 novembre 2012].

[2] D. Amyot et G. Mussbacher, « User Requirements Notation: The First Ten Years, The Next Ten Years », *Journal of Software*, vol. 6, no. 5, pp. 747-768, mai 2011.

[3] ProjetSEG Web (sur Foswiki), « URN Metamodel », 22 septembre 2012 [En ligne]. Disponible : http://jucmnav.softwareengineering.ca/ucm/bin/view/ProjetSEG/URNMetaModel. [Accédé : 23 novembre 2012].

[4] Daniel Amyot, Ph.D., ing. (professeur à l'école de science informatique et de génie électrique de l'Université d'Ottawa), communication personnelle (septembre à décembre 2012).

[5] International Communication Union, « User Requirements Notation (URN) – Language Definition », Recommendation Z.151, dans *Series Z: Language and General Software Aspects for Telecommunication Systems*, Genève, 2012, p. 13.

[6] HelpNDoc, « Google Chrome shows an error when searching HTML documentation », 2012 [En ligne]. Disponible : http://www.helpndoc.com/sites/default/files/documentation/html/index.html?GoogleChromeshowsanerrorwhensear.html. [Accédé : 22 novembre 2012].

[7] T. Lethbridge et R. Laganière, Chapitre 5 : « The basics of Object Constraint Language (OCL) » dans *Object-Oriented Software Engineering – Practical Software Development Using UML and Java*, 2e édition, London, McGraw-Hill, 2005, pp. 193-195.

ANNEXE I : Exigences pour le projet DA-1 - Extension du générateur de rapports pour jUCMNav (version initiale)

1. Le générateur de rapports doit pouvoir évaluer les tendances générales (si possible) de chaque élément présent dans un diagramme GRL, à partir d'une série d'évaluations de stratégies (*Strategy Evaluations*) (référence : bug # 777).

    i. Le générateur de rapports devrait donner à l'utilisateur l'option (booléenne) de calculer les tendances générales des éléments faisant partie d'un groupe de stratégies donné (via les préférences de jUCMNav).
    ii. Le générateur de rapports doit être en mesure d'ajouter une colonne à chaque tableau d'évaluations de stratégies (un tableau par groupe), représentant la tendance générale de chacun des éléments faisant partie de ce groupe de stratégies. La valeur inscrite dans la colonne sera un symbole qui représente cette tendance (ex. : « + » pour une tendance positive, « - » pour une tendance négative, « = » lorsqu'il n'y a aucune variation et « ? » lorsqu'il n'y a pas assez d'information ou qu'aucune tendance ne peut être identifiée).
    iii. Le générateur de rapports doit pouvoir évaluer les tendances générales de chaque élément présent dans un diagramme GRL à partir d'un nombre fixe d'évaluations, spécifié au préalable dans les préférences de jUCMNav.
    iv. Le générateur de rapports doit faire en sorte que toutes les évaluations de stratégies soient triées en ordre alphabétique de nom avant qu'un rapport ne soir généré.

2. Le générateur de rapports doit faire en sorte que le nom du modèle URN soit affiché sur la page couverture des rapports générés.

3. Le générateur de rapports doit pouvoir afficher les détails du rapport dans la ou les langue(s) choisie(s) au préalable dans les préférences de jUCMNav (i.e. un rapport pour chaque langue choisie doit être généré) (référence : bug # 813).

4. Le générateur de rapports devrait donner l'option à l'utilisateur de créer un rapport incluant uniquement un des deux types de diagrammes (UCM ou GRL, avec les informations associées), ou bien les deux types.

    i. Le générateur de rapports devrait pouvoir détecter la présence de diagrammes UCM et GRL, afin de générer un rapport incluant uniquement un/des diagramme(s) UCM si aucun diagramme GRL n'est trouvé (ou bien de générer un rapport incluant uniquement des diagrammes GRL si aucun diagramme UCM n'est trouvé).

5. Le générateur de rapports doit pouvoir afficher la validité d'un scénario (exécuté à l'aide d'un diagramme UCM), à l'aide d'une description (sous forme textuelle) de l'exécution.

    i. **Optionnel :** Le générateur de rapports devrait pouvoir inclure une représentation graphique (*Message Sequence Chart*) de l'exécution des scénarios.

Bogues trouvés dans le générateur de rapports:

1. Le générateur de rapports ne devrait pas donner l'option de parcourir le répertoire du rapport HTML (il est possible de faire cela à l'aide du menu latéral lorsque le rapport est ouvert avec IE ou Mozilla Firefox).

2. Le générateur de rapports devrait avertir l'utilisateur lorsque le nom de fichier défini par cet utilisateur est déjà existant dans le répertoire spécifié.

3. Le générateur de rapports doit être en mesure de bien encadrer les diagrammes sur les pages du rapport, lorsque celui-ci est généré en format RTF (référence : bug # 563).

4. Le générateur de rapports doit désactiver la boîte de texte « Report Prefix» lorsque le « Report Type » choisi est HTML.

5. **Très optionnel :** Le générateur de rapports devrait éviter de générer des pages blanches lors de la génération d'un rapport (formats PDF et RTF).

Autres:

1. Le code produit doit respecter les conventions établies dans le projet jUCMNav (structure de packages, noms des classes/variables, documentation minimale mais pertinente).

2. Le code produit doit être testé, idéalement (mais optionnellement) à l'aide de tests automatisés.

3. La documentation du générateur de rapports doit être mise à jour sur le site FosWiki du projet.

ANNEXE II : Exigences pour le projet DA-1 - Extension du générateur de rapports pour jUCMNav (version finale)

Nouvelles fonctionnalités à ajouter :

1. Le générateur de rapports doit pouvoir évaluer les tendances générales (si possible) de chaque élément présent dans un diagramme GRL, à partir d'une série d'évaluations de stratégies (*Strategy Evaluations*) (référence : bug # 777).
    i. Le générateur de rapports devrait donner à l'utilisateur l'option (booléenne) de calculer les tendances générales des éléments faisant partie d'un groupe de stratégies donné (via les préférences de jUCMNav).
    ii. Le générateur de rapports doit être en mesure d'ajouter une colonne à chaque tableau d'évaluations de stratégies (un tableau par groupe), représentant la tendance générale de chacun des éléments faisant partie de ce groupe de stratégies. La valeur inscrite dans la colonne sera un symbole qui représente cette tendance (ex: « + » pour une tendance positive, « - » pour une tendance négative, « = » lorsqu'il n'y a aucune variation et « ? » lorsqu'il n'y a pas assez d'information ou qu'aucune tendance ne peut être identifiée).
    iii. Le générateur de rapports doit pouvoir évaluer les tendances générales de chaque élément présent dans un diagramme GRL à partir d'un nombre fixe d'évaluations, spécifié au préalable dans les préférences de jUCMNav.

2. Le générateur de rapports doit faire en sorte que toutes les évaluations de stratégies soient triées en ordre alphabétique de nom avant qu'un rapport ne soit généré (référence : bug # 883).

3. Le générateur de rapports doit faire en sorte que le nom du modèle URN soit affiché sur la page couverture des rapports générés (référence : bug # 875).
    i. Lors de la génération d'un rapport HTML, le générateur de rapports devrait afficher les mêmes informations sur la page titre que sur les rapports en formats PDF et RTF.

4. Le générateur devrait utiliser des icônes plus représentatives dans le menu de gauche lorsqu'un rapport est généré en format HTML.

5. Le générateur de rapports doit pouvoir afficher les détails du rapport dans la langue dans laquelle jUCMNav est exécuté (référence : bug # 813).

6. Le générateur de rapports devrait donner l'option à l'utilisateur de créer un rapport incluant uniquement un des deux types de diagrammes (UCM ou GRL, avec les informations associées), ou bien les deux types (référence : bug # 876).
    i. Le générateur de rapports devrait pouvoir détecter la présence de diagrammes UCM et GRL, afin de générer un rapport incluant uniquement un/des diagramme(s) UCM si aucun diagramme GRL n'est trouvé (ou bien de générer un rapport incluant uniquement des diagrammes GRL si aucun diagramme UCM n'est trouvé).

7. Le générateur de rapports doit afficher toutes les dates de manière consistante sur les pages de titre des rapports, en utilisant le système de 12 heures AM/PM. Ces dates devraient aussi prendre en compte la langue dans laquelle jUCMNav est exécuté (référence : bug # 881).

8. Le générateur de rapports devrait avertir l'utilisateur avec un message d'avertissement lorsque la préférence "Show UCM diagrams and descriptions" (ou "Show GRL diagrams and descriptions") est désactivée. Puisque le API utilisé pour créer les préférences ne permet pas de désactiver les contrôles d'une page directement, cette fonctionnalité devrait être remplacée par des messages d'avertissements (référence : bug # 882).

9. Le générateur de rapports doit afficher la description des stratégies GRL en plus du nom. (référence : bug # 887).

10. Le générateur de rapport doit pouvoir afficher des informations sur les scénarios définis dans le modèle URN (Préconditions, initialisations de variables, postconditions, etc.), sous forme textuelle/tabulaire (référence : bug # 889).
    i. Les sous-sections ne doivent pas apparaître si aucune information n'est trouvée pour ces sous-sections.

11. Le générateur de rapports doit pouvoir afficher la validité d'un scénario (exécuté à l'aide d'un diagramme UCM), à l'aide d'une description (sous forme textuelle) de l'exécution. (référence : bug # 892).
    i. **Optionnel** : le générateur de rapports devrait pouvoir inclure une représentation graphique (*Message Sequence Chart*) de l'exécution des scénarios.
    ii. Le rapport HTML doit avoir des ancres qui redirigent l'utilisateur vers la section contenant la description des scénarios.

12. Le générateur de rapport doit afficher les évaluations des stratégies GRL et l'exécution des scénarios UCM indépendamment des diagrammes selon deux préférences spécifiées par l'utilisateur (référence : bug # 891).

Bogues trouvés dans le générateur de rapports :

1. Le générateur de rapports ne devrait pas donner l'option de parcourir le répertoire du rapport HTML (il est possible de faire cela à l'aide du menu latéral lorsque le rapport est ouvert avec IE ou Mozilla Firefox) (référence : bug # 877).

2. Le générateur de rapports devrait avertir l'utilisateur lorsque le nom de fichier défini par cet utilisateur est déjà existant dans le répertoire spécifié (référence : bug # 890).

3. Le générateur de rapports doit être en mesure de bien encadrer les diagrammes sur les pages du rapport, lorsque celui-ci est généré en format RTF (référence : bug # 563).

4. Le générateur de rapports doit désactiver la boîte de texte « Report Prefix» lorsque le type de rapport choisi est HTML (référence : bug # 879).

5. Le lien vers le site du projet jUCMNav (sur la page titre des rapports générés en format HTML) est brisé. Celui-ci doit être mis à jour (référence : bug # 878).

6. Le générateur de rapports ne devrait pas donner à l'utilisateur l'option de choisir une couleur de fond pour les rapports PDF. Cette fonctionnalité devrait être enlevée des préférences de jUCMNav.

7. Le générateur de rapports devrait gérer le cas où une boîte de texte ne contient aucune valeur dans les préférences. Eclipse lance une exception lorsqu'une de ces boîtes de texte est vide (référence : bug # 885).

8. **Très optionnel :** Le générateur de rapports devrait éviter de générer des pages blanches lors de la génération d'un rapport (formats PDF et RTF) (référence : bug # 888).

9. Le générateur de rapport doit manipuler les valeurs des préférences textuelles dans les *getters* afin de s'assurer qu'elle retourne une valeur valide.

Autres :

1. Le code produit doit respecter les conventions établies dans le projet jUCMNav (structure de packages, noms des classes/variables, documentation minimale mais pertinente).

2. Le code produit doit être testé, idéalement (mais optionnellement) à l'aide de tests automatisés.

3. La documentation du générateur de rapports doit être mise à jour sur le site FosWiki du projet.

Note : L'écriture en rouge dénote les exigences qui n'ont pas pu être complétées pour ce projet.