INSA Rennes

Equipe DIVERSE

Département informatique - Stage été 2015

Amélioration du support de Melange dans l'environnement de développement Eclipse

Du 06 Juillet au 28 Août 2015 (2 mois)

Auteur :
François Boschet

Maître de stage :
M. Arnaud Blouin

Correspondant INSA :
Mme. Marie Babel

remerciements

Je remercie toutes les personnes de l'équipe Diverse et plus particulièrement Mr Arnaud Blouin, pour m'avoir permis de décrocher ce stage, et Mr Thomas Degueule pour avoir passé beaucoup de temps a répondre à toute mes questions.

Table des matières

1

Présentation

INRIA

L'INRIA, ou Institut National de Recherche en Informatique et en Automatique, est un centre de recherche français composé de plus de 2700 chercheurs répartis dans des centres partout en France. Le site de Rennes est très lié avec l'IRISA (Insitut de Recherche en Informatique et Systèmes Aléatoires) et abrite près de 400 scientifiques répartis dans une douzaine d'"équipe-projet". Chaque équipe travaille sur un domaine précis, comme par exemple l'équipe Hybrid qui travaille sur le réalité virtuelle et les interactions 3D.

Diverse

Durant mon stage, j'ai fait partie de l'équipe Diverse, composé d'une quarantaine de personnes, dont 8 chercheurs permanents et dirigé par Mr Benoit Baudry. Diverse est un acronyme pour "Diversity-centric Software Engineering". Les axes de recherche de l'équipe sont l'ingénierie des langages et la variabilité, l'adaptation et la diversification des logiciels.

Ces recherches ont amenés au dévelopement de plusieurs logiciels notament Kevoree, une platforme de dévelopement logiciel distribués, Kermeta, un langage de manipulation de metamodèles, et Melange un outil de manipulation de DSL[1] sur lequel j'ai travaillé.

[1] Langage dont les specifications sont conçues pour un domaine d'application précis

2

L'environnement de travail

Melange

J'ai travaillé sur le langage Melange, le projet de thèse de Thomas Degueule. Melange est un language de programmation de type : "Langage-based Model-oriented programming language"; c'est à dire qu'il manipule des langages sous forme de métamodèles dans le but de créer un nouveau langage spécifique composé d'un ou plusieurs fragments de langages.

Melange est basé sur la platforme "client riche" d'Eclipse connue sous le nom de Eclipse RCP, cela permet . Cela permet d'utiliser les nmobreux plugins d'Eclipse déjà existant Mes plugins xtext pour la grammaire, emf pour les modèles et métamodèles, .

Beaucoup de features comme la generation d'editeurs de metamodels a la volée, la generation des-dits mm en plus des plugins eclipse de base comme l'outline, la completion auto, la consommation de memoire infinie, …

Eclipse

RCP - Rich Client Platform

Il s'agit de la plateforme de développement de plugins Eclipse. Néanmoins ce n'est pas un EDI a part entiere. RCP ne fournit pas de client riche de developpement, il fournit uniquement une suite de framework qui permet de developper des 'client riche'. Pour faire simple, Eclipse RCP est un sdk pour développement de plugins.

Le kit de développement propose tout un panel de "point d'extension" sur lesquels le développeur vient contribuer. Ce design en point d'extension permet notamment de minimiser la quantité de code à produire dans le cadre du développement de plugins Eclipse, étant donné que bon nombre de ces extensions ont déjà un comportement défini par défaut. Une bonne manièred'apprendre le foncitonnement de la plateforme client riche est de suivre les tutoriels de Lars Vogel disponible sur le site internet http://www.vogella.com/tutorials/

EMF / ECORE

EMF, ou Eclipse Modeling Framework, est un framework de modélisation et de génération de modèles. Il permet notamment de générer, à partir du modèle, un éditeur associé au modèle, une série de tests ainsi que beaucoup de petits plugins Eclipse comme un système d'hyperlien vers les définitions des objets manipulés, une gestion d'erreur et un déboggeur.

Langages

XTEXT

Xtext est un framework de développement de langages de programmation. A partir d'une grammaire sous la forme EBNF, xtext génère un compilateur pour ce langage ainsi que les fonctionnalités de bases de l'éditeur eclipse associé à ce langage, c'est à dire de la coloration syntaxique, de l'auto-complétion et une gestion des erreurs de syntaxe.

XTEND

Xtend est un langage de programmation qui, une fois compilé en java, tourne sur la *java virtual machine*. Le principal interêt est l'éclaircissement de la syntaxe de java. Par exemple les expression lambdas, dont la syntaxe est assez repoussantes en Java, devient très lisible et utilisable avec xtend. Pour faire simple, xtend est simplement du sucre syntaxique pour java. Ce qui rends le développement d'applications java plus aisé et moins verbeux.

3

Réalisation

Outline

Une des premiere fonctionnalité d'Eclipse sur laquelle j'ai travaillé été l'outline (fig 1.2). Il s'agit d'un panneau qui permet, dans notre cas, l'affichage de l'arbre syntaxique (du model?) de nos languages

Eclipse RCP propose, avec les outils de generation d'EMF de creer une outline minimale qui affiche l'arbre syntaxique des languages. Le problème c'est que l'on aimerait bien afficher des informations supplémentaires sur cet arbre, notamment les types des langages, et donc leur hiérarchie, ainsi que les "aspects" de ces langages.

Le principe est assez simple, il suffit d'étendre la classe qui créé l'arbre pour créer nos propres branches et feuilles dans l'abre qui représente les principales feature du langage. Il est intéressant de noter que la classe de base qui s'occupe de la création de l'arbre, `DefaultOutlineTreeProvider`, utilise principalement les deux méthodes: `createNode(IOutlineNode parent, org.eclipse.emf.ecore.EObject modelElement)` et `_isLeaf(org.eclipse.emf.ecore.EObject modelElement)` qui sont des dispatchers dynamiques qui sont capables de "choisir" la bonne implémentation de la méthode en fonction du type du `modelElement`. Mon travail a don été dans un premier temps de sous-typer ces fonctions dans le but de créer mes propres branches de l'arbre.

Par la suite, le travail a été de créer des provisionneurs de labels pour afficher correctement les éléments du langage dans l'abre avec leurs types associés.

Hyperlink

Une feature extrémement pratique de la plateforme Eclipse est l'hyperlinking qui permet, au ctrl+click (ou F3) sur un nom qualifié d'afficher la définition du-dit qualifié. Par exemple lorsque l'on "ctrl+click" sur un appel à une méthode, Eclipse nous affiche la définition de la méthode.

Le problème est que le mécanisme d'hyperliens ne permet de sauter qu'à la définition que de noms qualifiés, il ne permet de suivre d'URI Eclipse qui sont situés dans une chaine de caractère notamment. La grammaire de melange possède la règele de grammaire `'syntax' EcoreURI = STRING` qui permet de référencer l'URI du metamodel du langage que l'on essaye d'importer. Il peut être pratique de pouvoir naviguer facilement vers ce métamodèle vie l'URI fournie et pour cela il faut modifier le comportement du système d'hyperlien de Eclipse.

Etant donné que la feature hyperlien de melange a été généré par xtext, nous avons simplement redéfini le comportement de la méthode `createHyperlinkByOffset(XtextRessource resource, int offset, IHyperLInkAccpetor acceptor)` qui, dans le cas où la resource visé est notre élément "EcoreURI", créé un hyperlien vers le métamodèle et dans le cas contraire suit le comportement par défaut de création d'hyperlien.

Grammaire et metamodel

Pour continuer plus loin dans l'ajout de fonctionnalités à l'éditeur melange, il a fallut revoir la grammaire et le métamodèle de melange.

Le principal problème rencontré lors du développement de fonctionnalités comme la complétion automatique a été un problème de grammaire. Un exemple, la fonction de merge de melange dont la syntaxe suit la règle de grammaire `Merge returns merge: 'merge' mergedLanguage = STRING` ; ne permet pas d'effectuer de complétion automatique puisque xtext ne propose pas de mécanisme de complétion pour des objets de type STRING. Or les seuls objets qui peuvent être merge sont des objets de type `Language`. On peut donc réécrire la précédente règle de grammaire en `Merge returns merge: 'merge' mergeLanguage = [Language | QualifiedName]` qui permet, en plus d'accéder aux mécanismes de complétion integrés à Xtext, d'avoir une vérification de type directement das l'éditeur et non plus uniquement à la compilation. Cette modification de la grammaire va evidemment de pair avec la mise à jour du métamodèle du langage, en ajoutant un lien de composition entre la classe Merge et le type Language.

Le passage des STRING aux noms qualifiés dans la grammaire de melange a permis d'ajouter une véritable vérification des types qui facilite l'édition des fichiers melange,

ainsi qu'améliore l'auto-complétion par défaut d'Xtext qui era d'ailleurs la prochaine étape de développement.

Scope

Comme expliqué précedement, la modification de la grammaire melange a permis d'améliorer les possibilités d'auto-complétion et c'est donc la suite logique dans l'ajout de fonctionnalités à melange.

La complétion avec Xtext ressemble assez à l'Outline, il faut faire du polymorphisme sur la méthode `getScope(EObject object, EReference ref)` en sous-typant la classe `AbstractDeclarativeScopeProvider`. Le mécanisme de proposition de complétion est asez simple. L'appel au `CompletionProvider`, dans le cadre de mon implémentation, est un appel à un dispatcher qui appelera la bonne fonction `getScope(...)` avec comme EObject, la règle de grammaire dans laquelle se trouve l'utilisateur, et en EReference sur quel élément terminal précisement.

Par exemple, pour la règle merge expliqué précedement, on va proposé de merge tous les objets de type `Language` présent dans le scope du fichier melange. Pour d'autres fonctionnalités, nous ne voulons pas proposer n'importe quel élément du scope. Par exemple, dans le cadre de l'héritage, nous ne voulons pas proposer de sous-typer notre langage par lui-même, il faut donc restreindre le nombre de choix de coplétion possible.

Menu (W9)

This is gonna be tough to explain.

On, efin thomas :p, voulais pouvoir ouvrir un fichier d'un language avec tous les editeurs de ses languages sous types. Et ca eclipse veut pas. Mais alors pas du tout. Du coup j'ai fait le menu qui est capable d'afficher tous les editeurs dispo (petit screen du menu) mais il y a toujours un probleme a l'ouverture.

4

Conclusion

En définitive ce stage m'a permis de découvrir le monde de la recherche en travaillant sur un projet de thèse. Ce projet, Melange, m'a permis d'apprendre une facette de l'informatique que je ne connaissais pas, à savoir l'ingénièrie des modèles. Les bases du MDE m'ont été enseigné par Thomas Degueule et notamment la hiérarchie model/metamodel/metalangage.

Mais ce stage a été surtoutun défi technique pour moi; j'ai du apprendre un nouveau langage, xtend, et un nouvel environnement de développement, Eclipse RCP. Ce que j'en retiens, c'est que xtend est une manière sexy de faire du java et que ce langage permet de se détacher de certaines parties "lourdes" de la syntaxe java (notamment les expressions lambdas).

La plateforme riche d'Eclipse quant à elle est assez lourde et difficile d'accès. Le principal problème vient surtout d'un manque de documentation sur son fonctionnement et la manière d'ajouter des foncitonnalités. Cela a été le principal obstacle rencontré durant le stage où l'on a souhaité développer des fonctionnalités "avancées" pour nos plugins.

En définitive ce stage a été très enrichissant intellectuellement et m'a permis d'apprendre les bases de l'ingénièrie des modèles.

5

Conclusion

En Français

C'etait sympa,j'ai apris : - pas mal de choses sur les langages et leur representation (grammaire, models, metamodels, ast) - un nouveau langage fun pour faire du java en toute tranquilité -> xtend - les bases des plugin eclipse et le fait que la doc soit inexistante - pas mal de choses sur le monde de la recherche et du coup le master ercherche pourquoi pas -

Ce que j'ai fait: - quasiment fini - pas compris comment fonctionne log4j. Plutot comment faire ce que je veux avec. - ce magnifique rapport de stage (meta)

En Anglais

That was cool, I learned : - a lot of things on languages and them representation (grammar, model, metamodels, ast) - a new fun language that compile into java -> xtend - basics in plugin dev for the eclipse platform.the api have a really small documentation. - some thing about the MRI. Can be an option. Still need to think about it.

What I've done (Linkin Park): - almost done my job - I think I won't work with Log4j anymore. Annoying when you try to do "advanced" things. - This wonderful report (meta)