Logo ou
nom de l'entreprise d'accueil

Sujet du stage
Département informatique – Année scolaire

Diffusion du rapport (y compris en version électronique) :

☐ Non autorisée (le rapport ne sera pas diffusé, mais archivé par obligation légale)
☐ Autorisée en interne à l'INSA
☐ Autorisée en interne et en externe

Auteur :
Maître de stage :
Correspondant INSA :

Table des matières

1 Présentation **1**

INRIA 1

Diverse 1

2 Les bases **3**

Melange 3

Eclipse 4

Langages 4

3 Travail **5**

Outline 5

Hyperlink 6

Grammaire et metamodel 6

Scope 7

Menu (W9) 8

4 Conclusion **9**

1

Présentation

INRIA

L'INRIA, ou Institut National de Recherche en Informatique et en Automatique a Rennes, très lié avec l'IRISA (Insitut de Recherche en Informatique et Systèmes Aléatoires) est un centre de recherche en informatique et automatique.

Plus de [n] chercheurs dispersé dans des centres sur toute la france (et a l'etranger?) regroupés en groueps de recherches. Chaque groupe de recherche travaille sur un domaine precis comme par exemple l'equipe "Hybrid" qui travaille sur la realité virtuelle et les interaction 3D.

Chaque equipe de recherche est composé d'un directeur de recherche, de chercheurs permanents, de post-doc, d'inge de recherches, de thesards et de stagiaires.

Fait parti de l'equipe Diverse

Diverse

Diversity-centric Software Engineering

Il s'agit d'un des seul groupe de recherche de rennes qui travaille sur le genie et la diversite logiciel. Groupe composé en moyenne de 40 personne avec n permanents, m thesard et une armee de stagiaire

4 grands mots-clés : - ingénierie des langages - variabilité des logiciels - adaptation logicielle - diversification logicielle (beaucoup trop de "logiciel" dans ces phrases)

2

Les bases

(renommer)

Melange

J'ai travaillé sur le langage Melange, le projet de thèse de Thomas Degueule. Melange est un language [melange-lang.org] de programmation de type : "Langage-based Model-oriented programming language"; c'est à dire qui manipule des languages et permet de faire des operations dessus pour produire d'autre langugaes.

Le typage de langage permet de faire pleind e trucs avec leurs modeles comme par exemple les decouper (en extraire une partie coherente) pour ensuite coller le slice sur un autre model. Il y a aussi une notion d'heritage (et du coup de sous-typage)

(essayer de citer deux trois papier la dessus pour faire credible)

Melange basé sur la platforme eclipse RCP et repose sur les plugins xtext/xtend, ecore/emf, k3, …

Beaucoup de features comme la generation d'editeurs de metamodels a la volée, la generation desdits 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 aps de client riche de developpement (lol whatajoke) il fournit uniquement une suite de framework qui permet de developper des 'client riche'. En gros c'est un SDK specifique a Eclipse. (merci wipikedia)

Plein de features a implementer (menu, hyperlink, outline, console (sorry 'bout that), …) 2 manieres de faire, soit en contribuant a un "extension point" soit en "bindant" et redefinissant les provider/classes (j'arrive pas trop a bien ecrire ca)

EMF / ECORE

EMF, ou Eclipse Modelling Framework, est un framework de modelisation et de geenration de modeles Permet de creer un model, de lui associer un fichier de generation (nommé avec l'extension .gemodel) et de generer un compilateur pour le model, un editeur, un ast et des tests.

Ecore quand a lui definit les concepts manipulables par EMF (ECLass, ETypedElement, EString, …)

Langages

XTEXT

language d'ecriture de grammaire. Permet de creer un language et ses ast pour la grammaire donnée. Si combiner avec un metamodel EMF, permet de generer le model du language et un editeur integré a eclipse.

Tres pratique et permet de s'abstraire de totue la partie "dev d'un editeur" qui peut eter tres longue.

XTEND

Xtend est un language d'expression de java. Tout programme ecris en xtend sera traduit en java. Il est ters pratique car il a une syntaxe beaucoup plus souple que java, permet d'effectuer des opertaions assez complexes de facon rapide. Avant java8 il integrait des lambdas-expressions. Et la syntaxe de ces lambdas expression est vraiment plus sympa qu'en java.

3

Travail

(penser a retravailler les noms en general…) (dire ce que mon travail a ete de faire aussi ca pourrait servir)

Outline

(Screenshot d'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 avec les principales features des languages. Le probleme c'est qu'on aime bien faire des trucs "avancés" un peu funky par ici.(ca ne restera pas) Du coup il a fallut trouverle moyend e contribuer à l'element "outline" d'Eclipse. Ensuite de creer des provider pour faire le lien entre les elements reels (classes du modele, attributs, …) et leurs icones/chaines de char et tout le bordel

il a fallut aussi trouver comment creer l'arbre syntaxique (sais pas comment ca s'apelle en vrai) a partir des packages/classes/attributs.

J'aimerai bien parler du design pattern des packages/classes/attributs mais je ne sais pas 1) s'il y en a un 2) comment il s'appele

Hyperlink

Une feature extremement pratique de la plateforme Eclipse est l'hyperlinking qui permet, au ctrl+click (ou F3) sur un nom qualifié de passer a l'endroit ou a été définie ce nom qualifié. Par exemple a l'appel d'une methode,le ctr click sur le nom permet de sauter a l'endroit où a été définie la methode.

Le probleme que l'on a est que dans la grammaire ed melange, on assicie un metamodel a un language avec la regle (sortir la vraie regle) `syntax = STRING` Le fait est que l'on donne la une uri de metamodel sous la forme d'une string. Et que eclipse ne permet pas de base de faire des lien et de suivre des string.

Le principe de cette feature est de reperer a quelle endroit de la grammaire on se trouve. Ensuite on construit un HyperlinkProvider qui contient la region du lien (pour souligner le lien lors du click) l'URI a suivre pour acceder a la definition du token et le texte a afficher dans la boite de dialogue (généré par eclipse) si jamais il y a plusiuers choix (goto implem, goto def, …)

L'hyperlink est géré en bind-ant notre classe qui etends XbaseHyperLinkHelper. pas tres interessant comme fact.

Grammaire et metamodel

Pour continuer plus loin dans la customisation de l'editeur melange, il a fallut revoire la grammaire (et du cou ple metamodel) de melange.

Au début, la grammaire gerait tout en string (lolololol) et du coup le typecheck etait present uniquement a la compilation. pas tres pratique lorsque l'on veut 1-> corriger les erreurs de scope sans avoir a faire une compil. 2-> proposer de la completion "intelligente".

Exemple le renaming: on a une regle (citer les regles de renommage) qui definie le renommage d'éléments on remarque que les elements sont qualifés par des string. Ce qui implique, pas de typecheck, pas de completion possible (bloquage eclipse), … Du coup j'ai edité ces regles de sorte a avoir des noms qualifiés (plus de strings) de cette facon (nouvelle grammaire). On sait que l'on veut renommer des packages ici, donc on limite le choix (?faute de meilleur mot) aux noms qualifiés de type EPackage (lien vers le metamodel ecore)

Pour le classbinding, on a envie de pouvoir renommer tous les elements de types EClass et DataType mais pas les ETypedElements. Donc on restreint le typecheck (pas le bon mot je sais) à des EClassifiers (toujours voir le metamodel ou la type hierarchy).

Idem pour le operationbinding (cheplu) ou l'on limite le choix aux elelements qualifiés de type ETypedElement (on veut renommer tous les parametres depuis les operations jusquau references toussa toussa)

Tous ces changements dans la grammaire sont aussi a reperuter sur le metamodel. il s'agissait la de changer l'attribut "from" du packageBinding en une reference 0..* de PackageBinding vers EPackage.

Parmis tous lse changements effectuer sur le metamodel il y a ceux pour le renaming mais aussi pour l'heritage qui etait géré de la meme maniere en string et qu'il a fallut passer en nom qualifié.

Ce passage en nom qualifié a posé un probleme assez complexe. Lorsque l'on insere un modele en utilisant le mot-clé "syntax" suivi d'un URI (sous la forme d'une string) il n'y avait pas de typecheck, du coup a la compilation ,on chargeait le model et on avait ensuite acces au elements a renommer puisque ceux-ci etait sous la forme de string. Mais avec l'ajout des noms qualifiés, la syntax du model n'est pas chargé en memoire lors de l'edition du fichier, il le sera uniquement a la compilation. Du coup l'editeur ne peux pas verifier la validité du nom qualifié et son type. ce probleme a été recupéré par fabien qui s'occupera de gere ca (blablabla)

Scope

(penser a faire une capture d'un scope)

Chose ters pratique proposé de base par xtext/emf est la geenration d'editeurs pour le metamodel specifique. EMF fournit egalement un systeme de completion basé sur les scopes XText. en gros le foncionnement standard de la completion est: si l'element de la grammaire dans laquelle je suis rendu est un mot-clé, je propose tous les mots clés dispo. Si c'est un nom qualifié je propose tous les elements du type souhaité actuellemtn accessible dans le scope du fichier. Ce fonctionnement est tres pratique dans la plupart des cas. Mais il propose trop d'information. Mon but a été alors de reduire les choix de completion.

Pour ce faire il faut contribuer a l'extension point ScopeProvider (je crois je verifierai). Cela revient a enrigistrer notre extension de scope dans le fichier plugin.xml du projet et de surclasser le scopeprovider xtext.

J'ai opté pour un dispatcher qui dispatch sur les types d'elements de la grammaire. Ensuite on filtre l'endroit dans lequel on est (mal dit) grace à des referenceID généré par xtext. Il faut ensuite, soit recuprer le scope de base, en filtrer les resultats pour ne garder que ceux souhaité et renvoyer un nouveau scope. Soit recuperer directement les objets que l'on souhaite voir proposé et construire a la main note scope Provider. Le choix se fait surtout sur ce que le scope "voit". Dans certains cas, l'information que l'on cherche est perdu dansles extension point eclipse. il faut donc construire notre provider nous meme. Ex: "exemlpe qui va bien"

Il y a des fois, comme pour l'heritage, ou xtext connait tous les choix disponibles dans le scope et ou il ne fait que fournir trop d'informtaion. Pour l'heritage, xtext va nous fournir tous les languages qui sont disponibles dans le scope du ficher. Nous par contre nous ne voulons pas tout. Nous voulons tous les langages disponibles sauf nous meme. Il s'agit donc de recuperer le scope standard et de lui retirer le languages dans lequel nous sommes.

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

Donc en vrai ce stage tait cool. J'ai appris pa smal de choses sur les langages, le mdd (lolnope) et plein d'autres trucs. voila.*