## [Editeur Xtext ET Transformation Divine](http://www-master.ufr-info-p6.jussieu.fr/2013/Editeur-Xtext-et-Transformation,744)

**RAPPORT DU PROJET**

**eNCADRe pAR :**

Yann THIERRY-MIEG

**

**Réalisé Par :**

**Mokrane KADRI**

**Tahar OUAZIB**

**Sarah DAHAB**

**SOMMAIRE :**

1. Préambule………………………………………………………………………………………………..
2. Objectifs Du project…………………………………………………………………………………..
3. Outils de développements ………………………………………………………………………..
4. Description des Langages……………………………………………………………….

4.1. **Divine** ……………………………………………………………………………………

1. Le Système Devine………………………………………………………………..…
2. Les Types de données………………………………………………………………..
3. Les Processus………………………………………………………………..…………
4. Les Channels………………………………………………………………..………………
5. Les Transitions & Assertions………………………………………………………
6. Les Expressions………………………………………………………………..…………

4.2. Le système Gal …………………………………………………………………………..

4.3. **Xtext & Xtend**…………………………………………………………………………….

1. Description Du travail ………………………………………………………………………..…………..
2. Etapes et Réalisation
   * 1. Modélisation et Meta Modélisation ……………………………………………………
     2. L’éditeur de Texte ……………………………………………………………………….
        1. Règles De validation
        2. Scoping
        3. Outline
        4. Quickfixs
     3. Divine to Gal
3. Conclusion ………………………………………………………………..………………………………
4. Annexe………………………………………………………………..………………………………………

**4. Description des Langages:**

Dans cette partie nous allons présenter les diffèrent langages vu dans le projet, nous commenceront par Divine, puis Gal et nous terminerons par Xtext et Xtand.

**4.1 Divine :**

**4.2 Le Système Gal :**

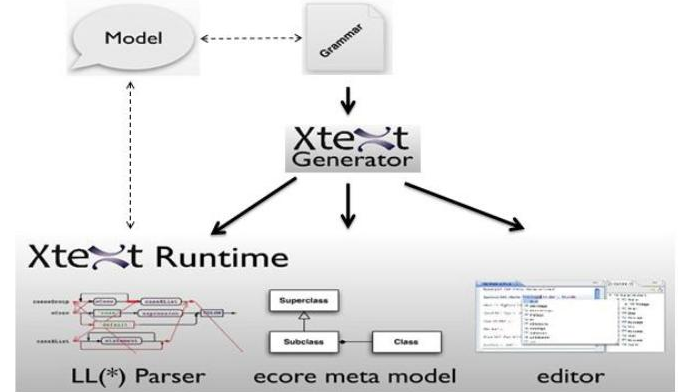
**4.3 Xtext et Xtend :**

**1. Xtext :**

Xtext permet le développement de la grammaire des langages spécifiques aux domaines (DSL: Domain Specific Languages) et d’autres langages textuels. Il est étroitement lié à l'Eclipse Modeling Framework (EMF).

Il permet de générer un méta-Modèle Ecore, un analyseur syntaxique (parser, en anglais) basé sur le générateur ANTLR ou JavaCC et un éditeur de texte sous La plate-forme Eclipse afin de fournir un environnement de développement intégré IDEspécifique au langage. La figure fournit une vue d’ensemble de l’outil Xtext. La forme textuelle du DSL constitue le point de départ. La grammaire du DSL est le point d’entrée de l’outil. Xtext produit, en sortie, un analyseur syntaxique, un éditeur et un méta-modèle pour le DSL.

Xtext permet de considérer la forme textuelle du DSL comme un modèle conforme à la grammaire d’entrée, donc au méta-modèle généré.

****

**1. Xtend:**

Xtend est un langage orienté objet de haut niveau .Syntaxiquement et sémantiquement Xtend à tire ces racines du langage Java. Il se distingue par syntaxe plus concise ainsi que quelques fonctionnalités supplémentaires telles que l'inférence de type, les méthodes d'extension, et la surcharge des opérateurs. Il intègre également des fonctionnalités connues de la programmation fonctionnelle, comme les expressions lambda. Xtend est typé statiquement et utilise le système de type de Java sans modifications. Il est compilé en code Java et s’intègre ainsi parfaitement avec toutes les bibliothèques Java.

Concernant notre projet, Xtend était indispensable pour éditer certaines classes générées par Xtext permettant ainsi d’implémenter un ensemble de fonctionnalités (Règle de validation, quickFix, etc.) . Ces fonctionnalités seront détaillées dans les sections qui suivent.

**5. Description du travail :**

La Réalisation du projet s’est dérouler en plusieurs étapes, dans cette section, on soulignera les parties majeures du travail réalisé qui seront présentés plus en détail dans le chapitre suivant.

* **La Définition du Langage :**

La première étape du projet consistait à écrire une grammaire Xtext pour le langage « Divine » tout en respectant la spécification (Syntaxe) de ses éléments comme présentés sur le site officiel  (<https://divine.fi.muni.cz/manual.html> ).

* **L’éditeur de texte :**

Une fois notre grammaire opérationnelle, on a implémenté un ensemble de fonctionnalités afin d’améliorer notre éditeurs de texte et le rendre plus riche. Parmi ces améliorations on trouve :

* + Les règles de validation :

On a défini certain nombre de règles de Validation pour éviter les problèmes d’incohérence qui peuvent survenir lors de l’écriture de programme avec notre éditeur.

* + Le scoping :

Les règles dites de « scoping » servent à gérer la visibilité et la portée des variables définies.

* + Les QuickFixs :

Les quickFixs offre la possibilité de corriger des erreurs susceptible de survenir lors de l’édition des programme « Divine ».

* + Les Outlines :

Les Outlines obtenu par défaut n’était pas assez convainquant, c’est pour quoi on a réalisé une version amélioré et complète de ce dernier.

* **La Transformation :**

Cette partie est le cœur du projet. Le concept est simple : transformer à un fichier décrivant un system donné, écrit en « Divine » (.dve), en un autre fichier « Gal » (.gal) décrivant le même système.

**6. Etapes et Réalisation :**

Dans cette section, nous aborderons plus en détails les différentes étapes de la réalisation du projet tout en les illustrons par des extrait de code ainsi que des capture d’écrans.

1. Modélisation et Meta-Modélisation :

Nous allons maintenant voir comment est construite la grammaire pour chaqu’un des éléments du langage « divine ».Nous utiliseront plutôt une approche basé sur la Meta modélisation pour appuyer les explications fournis.

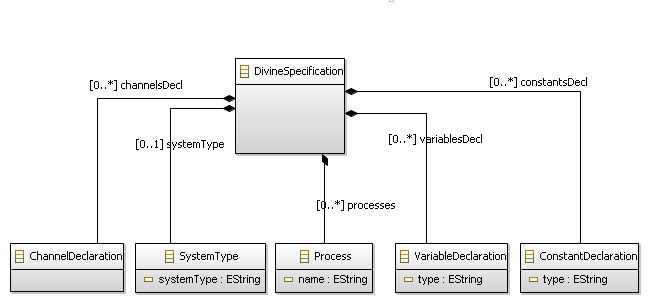
Les différents aspects des langages « divine » et « Xtext » ont été présentés dans les sections précédentes. La documentation officielle est également disponible sur leur site respectif site :

Xtext:  “<https://www.eclipse.org/Xtext/documentation.html> “

Divine: “ <https://divine.fi.muni.cz/manual.html> “

* **Le système Divine :**

Cette entité représente la structure globale d’un programme divine, tous les programme écrit en Divine devraient s’appuyer sur le Meta-modèle ci-après :

****

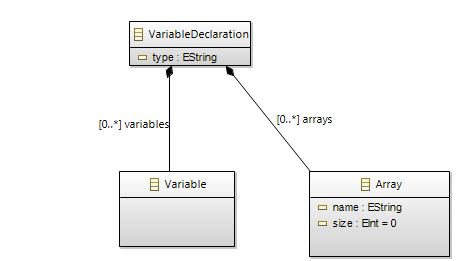
Ce Diagramme est construit à partie de la grammaire Xtext, montre bien que la méta-classe DivineSpecification englobe  les définitions des différentes type d’entité du système, tel que Les Processus (Meta-classe Process), ainsi que les différents type de données tel que les variables (Meta-classe VariableDeclaration), les Constantes (Meta-classe ConstantDeclaration), etc…

Pour plus de détaille sur la façon dont ce modèle a été générer ainsi que le code Xtext correspondant faut se référer à la règle DivineSpecification de la grammaire qui se situe dans le fichier : **fr.lip6.move.divine/src/fr/lip6/move/Divine.xtext**

* **Les Structure de donnée**

Divine permet de manipuler plusieurs types de données comme les variables, les constantes et les tableaux.

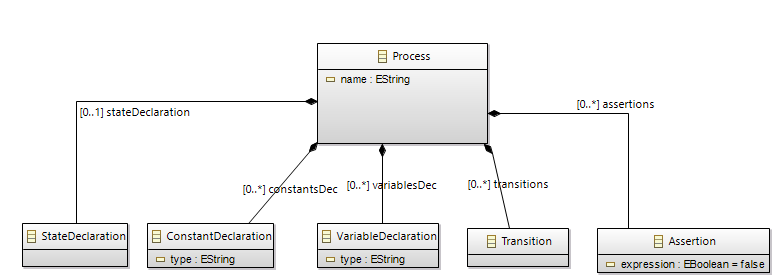
* + Les variables :



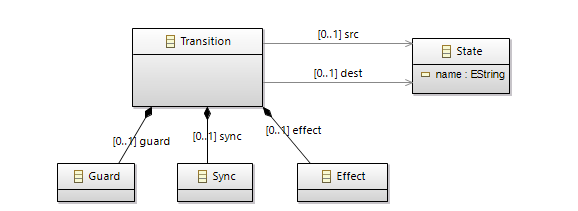
Comme l’indique le Modèle au-dessus, la déclaration de variable est en fait sois une déclaration d’un tableau celle d’une variable classique, ces deux élément ont un type (Int |Byte).

* + Les Constantes :

* + Les Process :



* Les transitions :

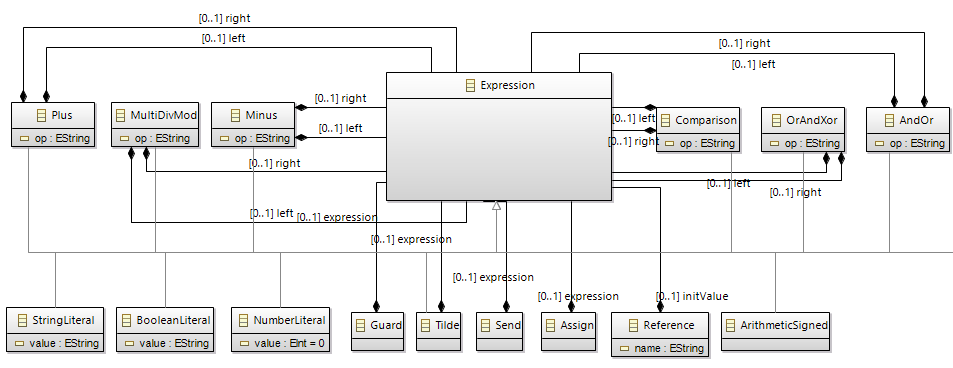


* Les Expressions :

Comme dans l’autre langage divine peut manipuler des expressions, parmi lesquelles on distingue les opérations classique qu’elle soit :

1. Binaire arithmétique (Addition, division …).
2. Binaire logiques (And, Or, comparaison …).
3. Unaire (entier négatif …).

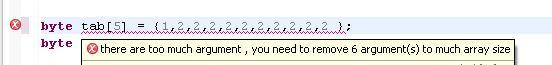
Le diagramme ci-après, expose la représentation qu’on a choisie pour définir l’expression Divine dans notre grammaire xtext. C’est un pattern composite,



1. L’éditeur Divine :
2. **Les Règles De validation**

L’ensemble des règles de validation définies vise à assurer :

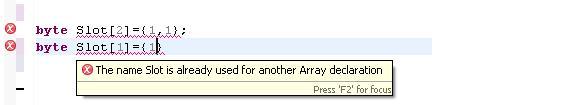
* Une cohérence entre la taille des tableaux et le cardinal de l’ensemble leurs éléments lors de la déclaration et initialisation.





* L’unicité des noms de Variables, Tableaux et Constantes.



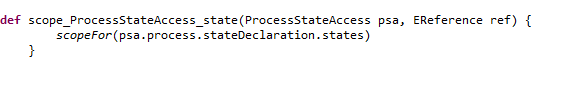


L’ensemble de ces règles est définie d’un fichier annexe « DivineValidator.xtend » fournie par Xtext. Voici un extrait de la méthode chargé de vérifier l’unicité de nom de variables. A noter que l’annotation @Check indique au Framework que la méthode qui suit est une règles de validation à vérifier.

1. **Le Scoping**

Xtext fournie un mécanisme de référencement, le problème est que les références au niveau syntaxique sont vues comme des identifiants (au même titre que les noms de variables) et sont résolues après que l’arbre syntaxique de la grammaire ait été généré.

Dans notre cas, on a eu recours à ce mécanisme pour résoudre les problèmes d’accès aux états des processus dans le corps de ces dernier .Voici un extrait du code :



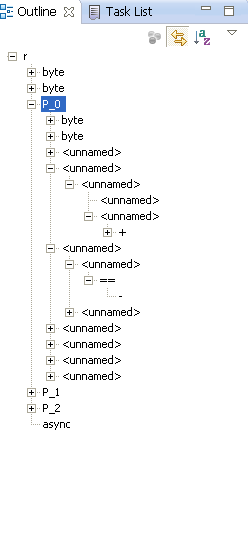
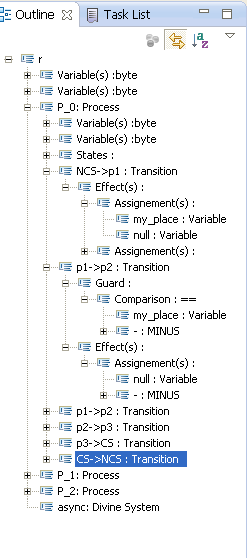
Les règles de scoping sont définies quant à elle dans un autre fichier annexe « DivineScopeProvider.xtend », généré automatiquement par Xtext aussi.

1. **Outlines :**

Les Outlines permettent de visualisé la structure des programmes, c’est comme un sommaire .il a pour vocation de faciliter navigation dans le code. Dans le code. Xtext génère par défaut des outlines qui portent le nom des labels (ID) des nœuds simples de la grammaire définie. Cependant pour les nœuds composés Xtext produit des balises « <unamed>. Dans cette optique, on définit un ensemble de méthodes qui ont pour but de guider Xtext à traiter tous les nœuds possible de notre grammaire et générer ainsi Des Outlines Complet. Le fichier à éditer est fourni par Xtext « DivineOutlineProvider.xtend » .Ce fichier regroupe un ensemble de méthodes qui se charge de renvoyer une chaine de caractère adéquate qui sera afficher dans les outlines.

Pour illustrer, voici des captures d’écran correspondant à l’aperçu des outlines avant et après la définition des règles de leur génération.

AVANT APRES

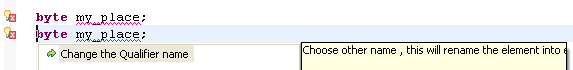
 

1. **QuickFix:**

Les deux figures suivantes illustres deux exemples de quickFixs implémentés.

Le premier permet de corriger les erreurs liés à l’absence de cohérence entre la taille d’un tableau est le cardinal des éléments qui lui sont affecter, le second permet de renommer les éléments ayant le même nom.

****

****

Le code correspondant aux quickFixes se trouve dans le fichier « DivineQuickFixProvider.xtend » du package « fr.lip6.move.divine.ui.quickfix »

1. **Divine to Gal :**

La tâche principale consistait donc réaliser une conversion du langage « divine » vers le langage « Gal » en s’appuyant à la fois sur le « Meta-Model » qu’on a obtenu grâce à notre grammaire mais aussi sur un ensemble de fonctions offertes par « Gal » via le plugin « Coloanne ».

L’ensemble des règles des transformations sont définies dans le paquage « togal », qui est composé de deux fichiers :

* « Converter.java » : ce fichier regroupe un ensemble de fonction utilitaire, capable notamment de faire des conversions de certain type d’éléments « divine » en leur équivalent en « gal », stocker les Variable et vérifier le typage des éléments.
* « DveToGalTransformer.java » : ce fichier est composé d’un certain nombre de fonction réalisant la transformation « Dve » « Gal ».

**TODO DETAILS**

1. **ANNEXE :**

* **Les documentations des langages de programmation**
* **La documentation Xtext:**

[**http://www.eclipse.org/Xtext/documentation.html**](http://www.eclipse.org/Xtext/documentation.html)

* **La documentation Xtend:**

[**http://www.eclipse.org/xtend/documentation/index.html**](http://www.eclipse.org/xtend/documentation/index.html)

* **Java Doc:**

[**http://docs.oracle.com/javase/7/docs/api/**](http://docs.oracle.com/javase/7/docs/api/)

* **GAL :**

[**http://move.lip6.fr/software/DDD/gal.php**](http://move.lip6.fr/software/DDD/gal.php)

* **Divine et sa spécification :**

[**https://divine.fi.muni.cz/manual.html#the-dve-specification-language**](https://divine.fi.muni.cz/manual.html#the-dve-specification-language)

* **Liens Utiles :**
* **Les expressions en Xtext:**

[**http://xsemantics.sourceforge.net/xsemantics-documentation/Expressions-example.html**](http://xsemantics.sourceforge.net/xsemantics-documentation/Expressions-example.html)

* **Code source pour comprendre la grammaire:**

[**http://divine.fi.muni.cz/trac/browser/divine/dve/**](http://divine.fi.muni.cz/trac/browser/divine/dve/)

* **Autre Liens Utiles**
* **Le site du Laboratoire :**

[**http://move.lip6.fr/software/DDD/**](http://move.lip6.fr/software/DDD/)

* **TeamCity :**

[**http://teamcity-systeme.lip6.fr/login.html**](http://teamcity-systeme.lip6.fr/login.html)

* **Modèles Utiliser pour les tests :**

[**https://projets-systeme.lip6.fr/svn/research/libddd/libits/trunk/tests/test\_models**](https://projets-systeme.lip6.fr/svn/research/libddd/libits/trunk/tests/test_models)

[**https://projets-systeme.lip6.fr/svn/research/libddd/libits/trunk/perfs/test\_models**](https://projets-systeme.lip6.fr/svn/research/libddd/libits/trunk/perfs/test_models)