Hes·so VALAIS WALLIS
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

Domaine Sciences de l'ingénieur
Rte du Rawyl 47
CH- 1950 Sion 2
Tél. +41 27 606 85 11
Fax +41 27 606 85 75
info@hevs.ch

www.hevs.ch

Filière Systèmes industriels

Orientation Infotronics

Diplôme 2008

Hugo Nunes

Automatic Designer

Professeur Medard Rieder

Expert Thomas Seiler

Sion, le 21 novembre 2008

HES-SO Valais

SI	TV	EE	IG	EST
X	X			

Données du travail de diplôme
Daten der Diplomarbeit

FO 1.2.02.07.DB
mam/30/06/2008

Confidentiel / *Vertraulich* ☐ oui / ja ☒ non / nein

☒ FSI ☐ FTV	Année académique / *Studienjahr* **2007/2008**	No PS / *Nr. PS* **it/2008/24**
Mandant / *Auftraggeber* ☒ HES—SO Valais ☐ Industrie ☐ Ecole hôte	Etudiant / *Student* **Hugo Nunes**	Lieu d'exécution / *Ausführungsort* ☒ HES—SO Valais ☐ Industrie ☐ Ecole hôte
Professeur / *Dozent* **Medard Rieder**	Expert / *Experte* *(données complètes)* **Thomas Seiler** - Swisscom	

Titre / *Titel*

Automatic Designer

Description et Objectifs / *Beschreibung und Ziele*

The Automatic Designer semester work has shown the feasability of a meta model driven development tool for embedded systems.

The goal of the Automatic Designer diploma work will be to develop a prototype of a FSM driven tool for embedded targets. This means that a developper may draw FSMs and code for controllers / microprocessors will automatically be generated from the FSM model.

The goals of the present diploma work are:

— Refine the results of the semester work in order to derive a professional tool that can be used in daily workflow. A specification of the refinements will be elaborated at the beginning of the diploma work.

— Elaborate a complete and detailed documentation of the meta model, the design patterns and the transformation rules.

— Develop a functional prototype that is able to target a PIC ™ and an AVR ™ or ARM™ target.

Signature ou visa / *Unterschrift oder Visum*	Délais / *Termine*
Resp. de la filière *Leiter des Studieng.:*	Attribution du thème / *Ausgabe des Auftrags:* 01.09.2008
Etudiant/*Student*:	Remise du rapport / *Abgabe des Schlussberichts:* 21.11.2008, 12:00
	Exposition publique / *Ausstellung Diplomarbeiten:* 28.11.2008
	Défense orale / *Mündliche Verfechtung:* semaine/Woche 49

Hes·so VALAIS WALLIS
Rte du Rawyl 47 - 1950 Sion 2

2 1 NOV. 2008

Rapport reçu le / *Schlussbericht erhalten am* ..... Visa du secrétariat / *Visum des Sekretariats* .....

Nom, Prénom

Nunes Hugo

2008

Titre du travail de diplôme

Titel der Diplomarbeit

Objectif

Développer les composants logiciels nécessaires à l'obtention d'un outil permettant la génération de code pour des cibles embarquées à partir de FSM. Pour ce travail, utiliser le développement basé sur les méta-modèles.

Résultats

Les plugins de création de diagrammes d'état correspondent au cahier de charges et les règles de transformation génèrent le code correspondant à la spécification. Très peu de tests on été effectués.

Mots-clés

Modelling, Meta-modelling, code generation, UML 2.1

Ziel

Mit endlicher Automat zeichnen, Kode erzeugen.

Resultate

Die plugins fur graphical Editor sind fertig und die Kode is richtig erzeuge aber nicht viel Test werden gemacht.

Schlüsselwörter

Modelling, Meta-modelling, code generation, UML 2.1

Table of Contents

Index des illustrations

Index des diagrammes

Index des illustrations

Illustration 1: Editeur en arborescence....11
Illustration 2: Editeur généré par Topcased et customisé....12
Illustration 3: Vue générale de application.diagramconfigurator....13
Illustration 4: Vue générale de statemachine.diagramconfigurator....14
Illustration 5: Exemple d'architecture MVC....15
Illustration 6: Choix de figures....18
Illustration 7: Edge object dans diagramconfigurator....19
Illustration 8: Tri de child elements superflus....20
Illustration 9: Templates....32
Illustration 10: Signaux définis dans l'éditeur....35

Index des codes

Index des figures

1 Introduction

De plus en plus d'outils opensource permettent de réaliser ce que des outils onéreux nous proposent. Dans le cadre de ce travail de diplôme, des outils de développement doivent être conçus et implémentés. Ces outils permettront au développeur de dessiner des diagrammes de machines d'état et le code correpondant pour des petits systèmes embarqués sera généré automatiquement.

1.1 Code et références aux annexes

Afin de maintenir la lisibilité de ce rapport, les exemples de code donnés peuvent avoir été simplifiés, le code peut donc être une version raccourcie et incomplète, cependant, lorsque cela est possible, une référence à l'annexe comprenant l'intégralité du code est mentionnée.

1.2 Symboles

Le symbole → présent sur le code (rapport ou annexes) représente la continuation de la ligne précédente. En TGL les retours ligne au sein d'une directive n'étant pas permis, cette solution permet de rendre visible la totalité de la ligne.

2 But

Le but de mon projet est de dévellopper des plugins et de créer des régles de transformation sur la plateforme Eclipse grâce à Topcased et MDWorkbench.

Les plugins implémentent un éditeur graphique de machines d'état. Les règles de transformation reprennent un modèle de machines d'état créé par l'éditeur graphique et générent le code source C correspondant.

L'éditeur graphique devra être le plus "user friendly" tout en conservant un dégré maximum de compatibilitée avec la superstructure d'UML, cela afin de permettre une très facile évolution ou adaptation avec d'autres outils.

Le code a générer est destiné à des petits microcontrôleurs (PIC, AVR, ARM) et le code généré doit rester assez simple.

Le résultat de ce travail doit pouvoir servir à des fins éducatives au sein de l'école. Les étudiants pourront disposer de cet outil lors de laboratoires pratiques ou lors de la summer school.

A terme, cet outil doit être amélioré afin d'obtenir un outil très performant et d'une grande simplicité d'utilisation pour l'implémentation de microcontrôleurs dans l'industrie.

3 Cahier des charges

Le projet de semestre à démontré qu'il était possible de réaliser par dévelloppement dirigé par les méta-modèles, un outil de dévellopement pour systèmes embarqués.

Les spécifitées et fonctionnalitées dont l'outil devra disposer ont été définies comme suit :

- Le métamodèle doit être conforme à la superstructure UML.
- L'éditeur graphique doit évoluer afin de proposer une application professionnelle, tel que le dévellopeur puisse en faire une utilisation quotidienne.
- L'application devra être structurée tel que sur la figure suivante :
- L'outil doit contenir un champ ISR qui permet au dévelloppeur d'implémenter le code à placer dans l'Interrupt Service Routine.
- L'outil doit permettre l'implémentation de :

  multiples régions,
  Initial State
  Final State
  State
  Choice
  EntryAction
  ExitAction
  Events
  Timer
- L'application devra être structurée tel que sur la figure suivante :

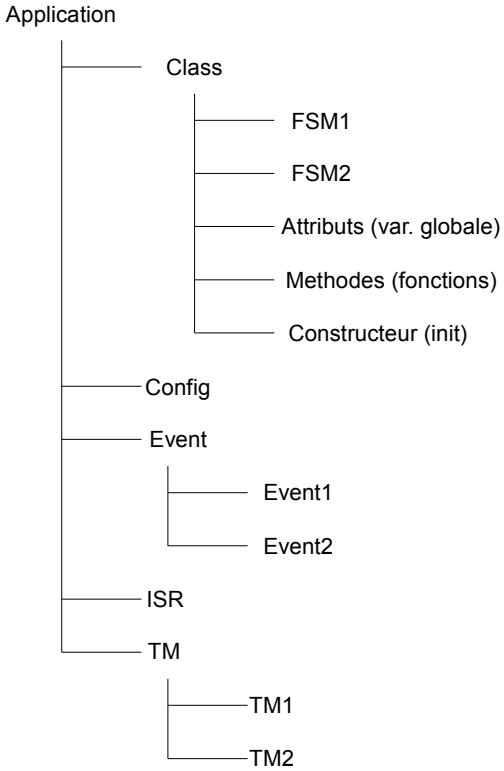


Schéma 1: Structure de l'application

4 Outils

Les principaux outils de dévelloppement employés dans ce travail sont :

4.1 Topcased *version: 2.1.0*

Topcased est un ensemble d'outils pour le dévelloppement d'applications critiques et de systèmes. Il fonctionne avec Eclipse et se présente sous forme de plugins ou d'un bundle. Dans ce travail, sa principale utilisation est la génération de code pour la création d'un éditeur graphique.
Durant le travail de semestre, la version 1.4 de cet ensemble d'outils avait été utilisée. Au début de ce travail de diplôme, les résultats du travail de semestre ont tout d'abord été readaptés pour la nouvelle version 2.1, cela afin de détecter de possibles différences d'implémentation.

4.2 MDWorkbench *version: 2.1.5*

Aussi basé sur Eclipse, MDWorkbench est un outil très puissant et facile a prendre en main qui permet la manipulation de modèles. Génération de code, ajout de connecteurs, gestion de métamodèles et de modèles, sont quelques unes des tâches que cet outil permet de réaliser.
Dans ce travail, cet outil permettra la génération de code C à partir d'un modèle créé à l'aide de l'éditeur graphique.

4.3 Cible

La cible utilisée pendant le dévelloppement est un **PIC18F458** de chez Microchip. Ce microcontrôleur est monté sur une carte de démonstration PICDEM 2 PLUS.

Compilateur cible

Pour programmer la cible, le compilateur HI-TECH et l'environnement de programmation pour PIC MPLAB IDE v8.10 ont été utilisés.

4.4 Autres

L'éditeur de texte Notepadd++ s'est aussi avéré très utile pour différentes tâches telles que l'observation du code généré.
Le compilateur gcc, MINGW et MSYS très utiles lors de la procédure de dévelloppement(pour des tests,etc...).

Remarques

Topcased 2.1.0 ne génère plus par défaut la ligne spécifiant l'ID du template par défaut à utiliser lors de la création de nouveaux diagrammes. Sur la version 1.4 cette configuration était générée automatiquement. Se rapporter au tutorial de Topcased page 16 en annexe pour retrouver la ligne à replacer.

5 Etapes

Afin de réaliser ce travail les étapes principales sont :

- Préparation
  Mise à jour des outils, recherche informations, définition du planning

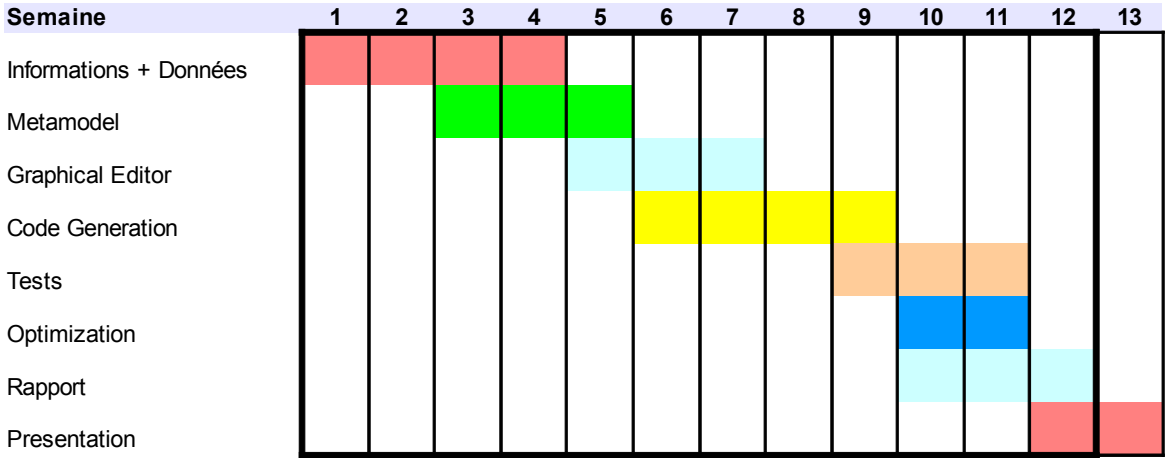
- Métamodèle
  Analyse et définition du métamodèle, implémentation de l'éditeur en arborescence

- Editeur graphique
  Definition de UI, implémentation et customisation de l'éditeur graphique

- Generation de code
  Conception et dévelloppement des fonctionnalitées et comportement, implémentation des règles de transformation

- Tests et optimization
  Mise à l'épreuve des fonctionnalitées, améliorations

Planning



6 Méta-modèle

6.1 Base du métamodèle

Le métamodèle est basé sur la spécification de la superstructure UML. La version employée a été le UML2.1 .

La spécification est énorme et comporte beaucoup de références récursives, même pour le **StateMachine package** . Il faut donc restreidre le nombre d'éléments et propriétées que l'on va garder pour le travail afin de pas se retrouver à recopier toute l'implémentation de la spécification dans l'application.

Pour déterminer les éléments substantiels, il faut donc bien comprendre le concept caché derrière chacun d'eux. Cela avait déjà en partie été fait pendant le travail de semestre car le diagramme représentant la machine d'états avait été étudié. Il faut cependant reprendre ce diagramme comme point de départ *(annexe A - UML_StateMachine)* et déterminer quels éléments ajouter ou supprimer.

6.2 Définition du métamodèle

6.2.1 StateMachine

(annexe A - statemachinediag)

Par rapport à la spécification *(annexe A - UML_StateMachine)*, on n'implémentera pas les **ConnectionPointReference** .

6.2.2 Application, ISR, Config et PackageableElements

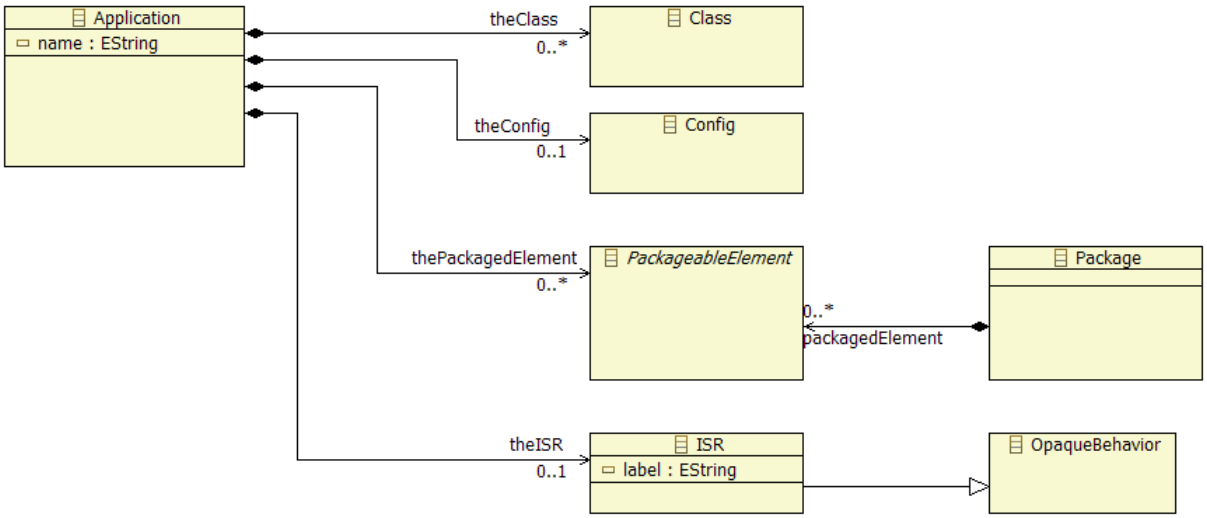


Diagramme 1: Metamodele - applicationdiag

L'élément **Application** est le *root* du modèle créé à partir de ce métamodèle. Il représente l'application dont on veut dessiner le comportement. L'attribut **name** représente le nom de l'application. L'élément **ISR** sert a contenir le code que le dévelloppeur veut insérer dans l'Interrupt Service Routine. Il est donc une spécialisation de l'élément **OpaqueBehavior** car il contient un champ **body** qui peut contenir le code. L'élément **Config** est un élément qui va contenir différentes informations concernant la cible. **Application** peut contenir plusieurs **PackageableElement** qui correspondent aux différents éléments tels que les **TimeEvent** , **Signal** et **SignalEvent** .

6.2.3 Events

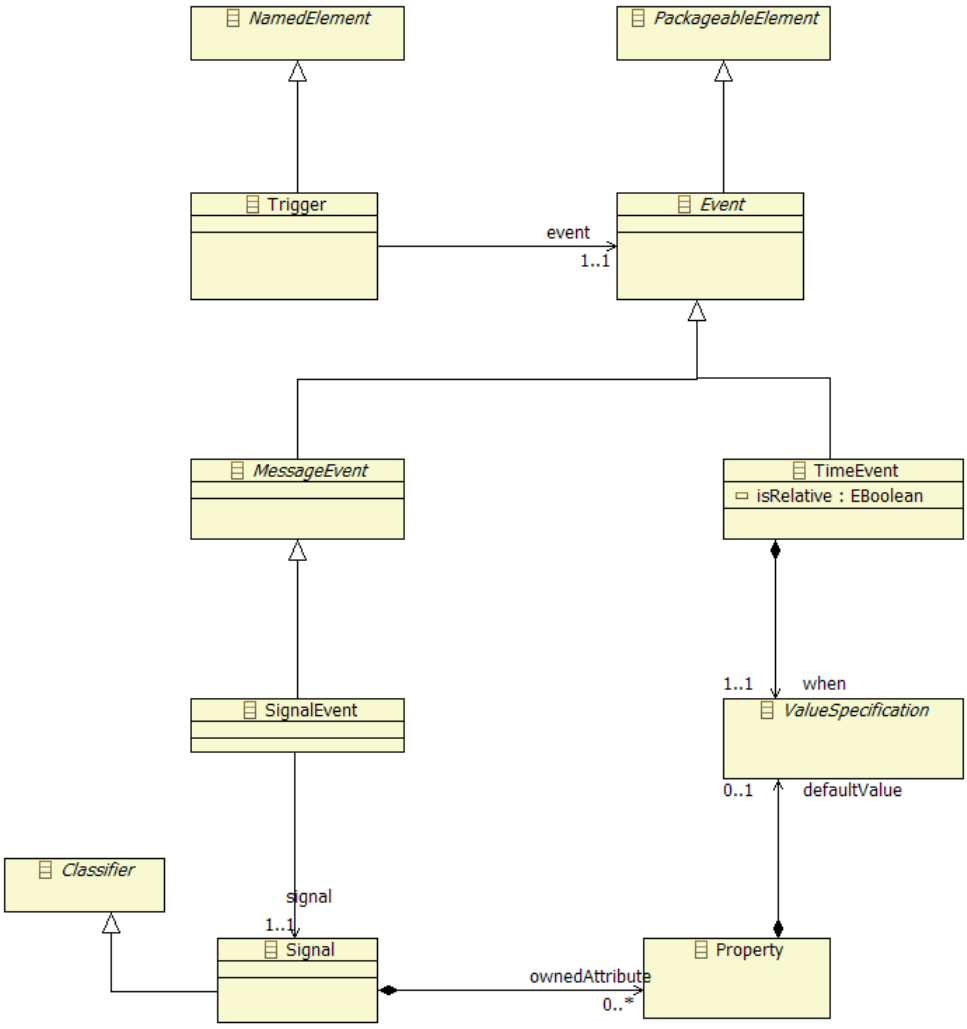


Diagramme 2: Metamodele - triggerdiag

Ce diagramme définit les évènements qui pourront servir de trigger aux transitions. On y constate qu'il a deux sortes d'events, le **TimeEvent** et le **SignalEvent** . Le **Signal** pourra contenir plusieurs **Property** en tant que **ownedAttribute** mais le **SignalEvent** ne reférence qu'un **Signal** .

6.2.4 ValueSpecification

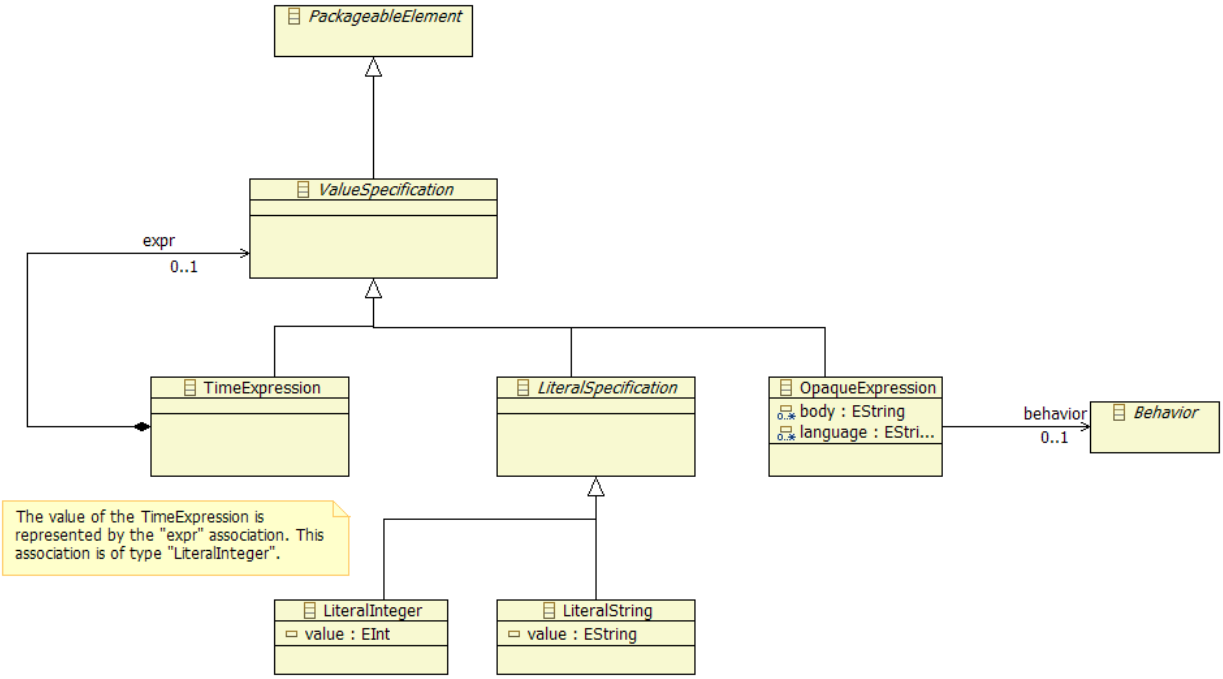


Diagramme 3: Metamodele - valuespecificationdiag

Dans ce diagramme sont définis les types qui peuvent représenter l'élement **ValueSpecification** . Il est défini par l'annotation texte présente sur le diagramme que la valeur **expr** de **TimeExpression** est de type **LiteralInteger** .

6.2.5 Behavior

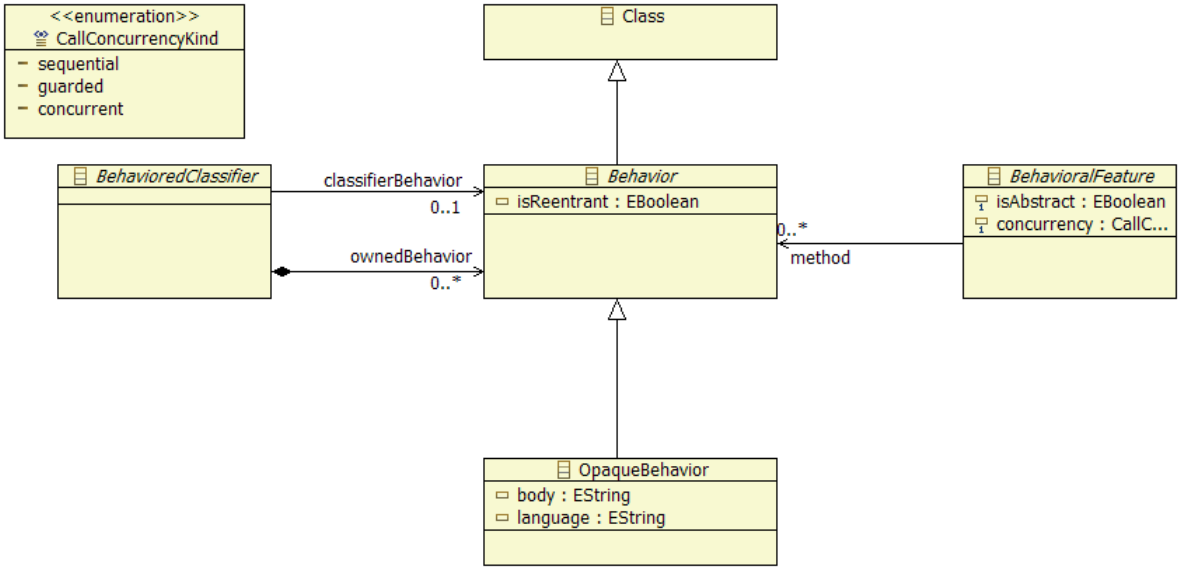


Diagramme 4: Metamodele - behaviordiag

En consultant les autres diagrammes dans les annexes, on voit que **Class** est une spécialisation de **BehavioredClassifier** . Revenant sur ce graphique on voit que **BehavioredClassifier** a une association et une aggréagation d'un **Behavior** . Cela veut dire

que **Class** peut posséder un comportement et que ce comportement peut lui-même définir la manière de comment elle se comporte.

6.2.6 Remarques

On constate que quelques éléments ne font pas partie de la spécification UML. Cependant ces éléments ont été consciamment choisis. (Application, Config...)

Eléments pour améliorations

Dans les diagrammes complets en annexe (annexe A) des propriétés et des éléments sont superflus dans l'état actuel de l'implémentation. Cependant ils ont été prévus pour une version ultérieure de l'outil. Parmi ces propriétées nous avons :

- les **Constraint** de **Operation** (precondition,postcondition,bodyCondition)
  (annexe A - operationdiag)
- le **Target** de **Config** (prévu pour makefile par exemple)
  (annexe A - configdiag)

Erreurs

Quelques erreurs ont été trouvées et n'ont malheureusement pas pu être corrigées pour cette version de l'outil, les erreurs décrites ci-dessous concernent uniquement le métamodèle. Le comportement érroné de quelques unes a pu être corrigé par la suite dans les étapes de dévelloppement ultérieures :

- le type OpaqueExpression n'était finalement pas utile
  (annexe A - configdiag)
- propriétée nestedClassifier inutile
  (annexe A - classdiag)
- le type Type n'a pas été implémenté par la suite

7 Editeur Graphique

7.1 Informations Utiles

Ce chapitre fournit les customisations réalisées ainsi que des concepts ou informations supplémentaires nécéssaires pour customiser l'éditeur graphique.
La marche à suivre pour la création d'un éditeur graphique est expliquée dans le tutorial que j'ai préparé. *(annexe T - Topcased From Metamodel to Graphical Editor Tutorial)*
Ce tutorial donne une explication pas à pas de la marche à suivre pour la création d'un éditeur graphique basique à partir d'un métamodèle quelconque.

7.1.1 Nodes et Edges

Les **nodes** sont des représentations graphiques d'objets. Ces nodes sont affichés à l'aide de figures.
Un **edge** est une relation représentée graphiquement par une ligne ou liaison.

7.1.2 Multi-Diagrams Ecore

Le modèle ecore peut être composé de multiples diagrammes. Ces diagrammes peuvent être contenus dans un seul fichier ou alors sous forme de plusieurs fichiers ecore. Au cas où les diagrammes sont dans différents fichiers, l'utilisation de "Load Resource" permet d'accéder aux propriétés des différents diagrammes. Les mêmes classes peuvent ainsi être reprises dans plusieurs diagrammes (par un simple drag&drop par exemple).

7.1.3 Multi-Diagram-Configurators

Pour donner la possibilitée de créer plusieurs diagrammes imbriqués sur l'éditeur graphique, plusieurs "diagramconfigurators" sont nécéssaires. Dans chaque "diagramconfigurator", un "Model Object" doit avoir un "Diagram Reference". C'est ce dernier qui reférence le diagramme qui peut être créé. Plusieurs "Load Ressource" permettent de lier les différents "diagramconfigurator" entre eux ainsi que leur "editorconfigurator" et "genmodel".

7.2 Les Différentes Phases

Pour réaliser l'éditeur graphique, plusieurs étapes sont nécéssaires.
Il faut d'abord implémenter le métamodèle sur le logiciel si ce n'est pas déjà fait.

On réalise ensuite le premier éditeur. Cet éditeur est un éditeur en arborescence aussi appellé **tree editor** . Il permet d'effectuer les premiers contrôles sur le métamodèle implémenté. Cet éditeur est en faite une très simple interface du modèle implémenté sous forme de `xml` .

7.3 Eclipse Modeling Framework et éditeur en arborescence

Eclipse propose un framework qui permet de créer ses modèles à l'aide d'un éditeur en arborescence.

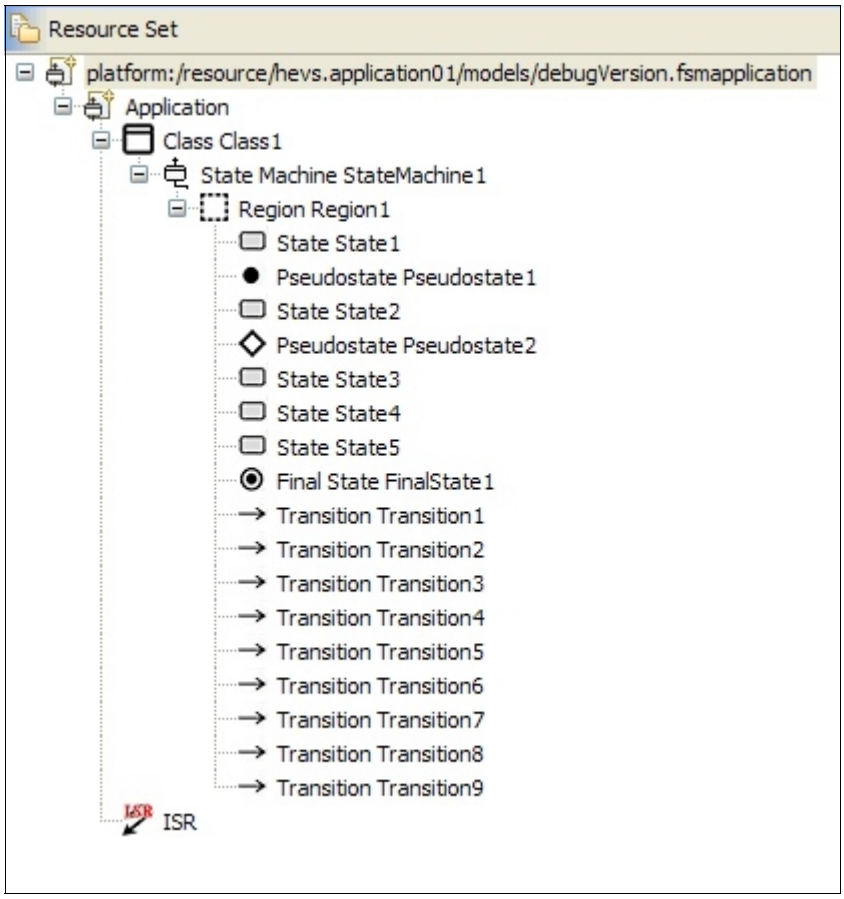


Illustration 1: Editeur en arborescence

Pour créer cet éditeur, deux fichiers seulement sont nécéssaires :

- fichier `.ecore` contenant le métamodèle
- fichier `.genmodel` contenant la configuration très basique pour cet éditeur

Cependant, on se rend vite compte que l'implémentation de modèles avec cet éditeur est une tâche pénible. De plus, il est très difficile d'avoir une vision de l'ensemble de l'implémentation avec cette vue.

7.4 Génération d'éditeur graphique par Topcased

Heureusement, il est possible d'obtenir un éditeur graphique plus conventionnel en ce qui concerne une machine d'états grâce à Topcased.

L'étape suivante consiste à configurer des fichiers spécifiques à Topcased afin qu'il soit en mesure de générer le code correspondant à l'éditeur voulu. Il faut confgurer au moins deux fichiers. Le premier fichier va générer un projet de plugin qui va servir d'interface entre le nouveau éditeur et le `tree editor`. Ce premier fichier est l' `éditorconfigurator`.

Le deuxième fichier, le `diagramconfigurator` permet de spécifier la configuration de base du nouveau éditeur graphique. Il est possible de créer des éditeurs à plusieurs niveaux en configurant plusieurs `diagramconfigurators`.

Une fois le code pour le plugin généré, il faudras l'étudier et analyser afin de déterminer les corrections et modifications à amener au code afin de customiser l'éditeur.

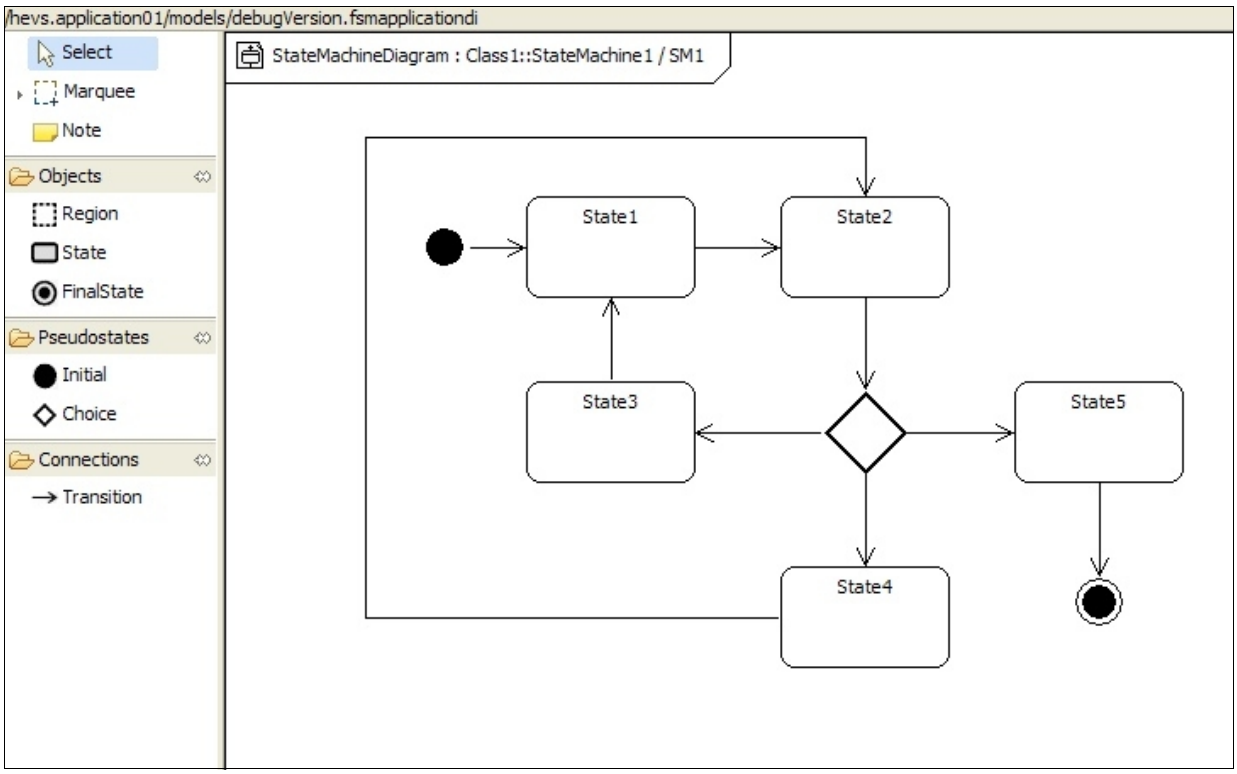


Illustration 2: Editeur généré par Topcased et customisé

L'éditeur de la figure précédente est un éditeur généré par Topcased et qui a été customisé.

Pour que Topcased puisse générer un éditeur graphique, il faut, en plus des deux fichiers précédents, aussi configurer :

- un fichier editorconfigurator
- un fichier diagramconfigurator

7.5 Deux éditeurs graphiques

Afin de séparer la partie concernant l'application en général (classes, signaux, ISR, ...) de la partie concernant les machines d'état. Deux diagramconfigurators ont été réalisés. L'éditorconfigurator n'a pas besoin d'être doublé car il ne fait que générer un interface entre l'éditeur graphique (généré par Topcased) et l'éditeur en arborescence. Les deux éditeurs graphiques peuvent utiliser le même editorconfigurator.

application.diagramconfigurator

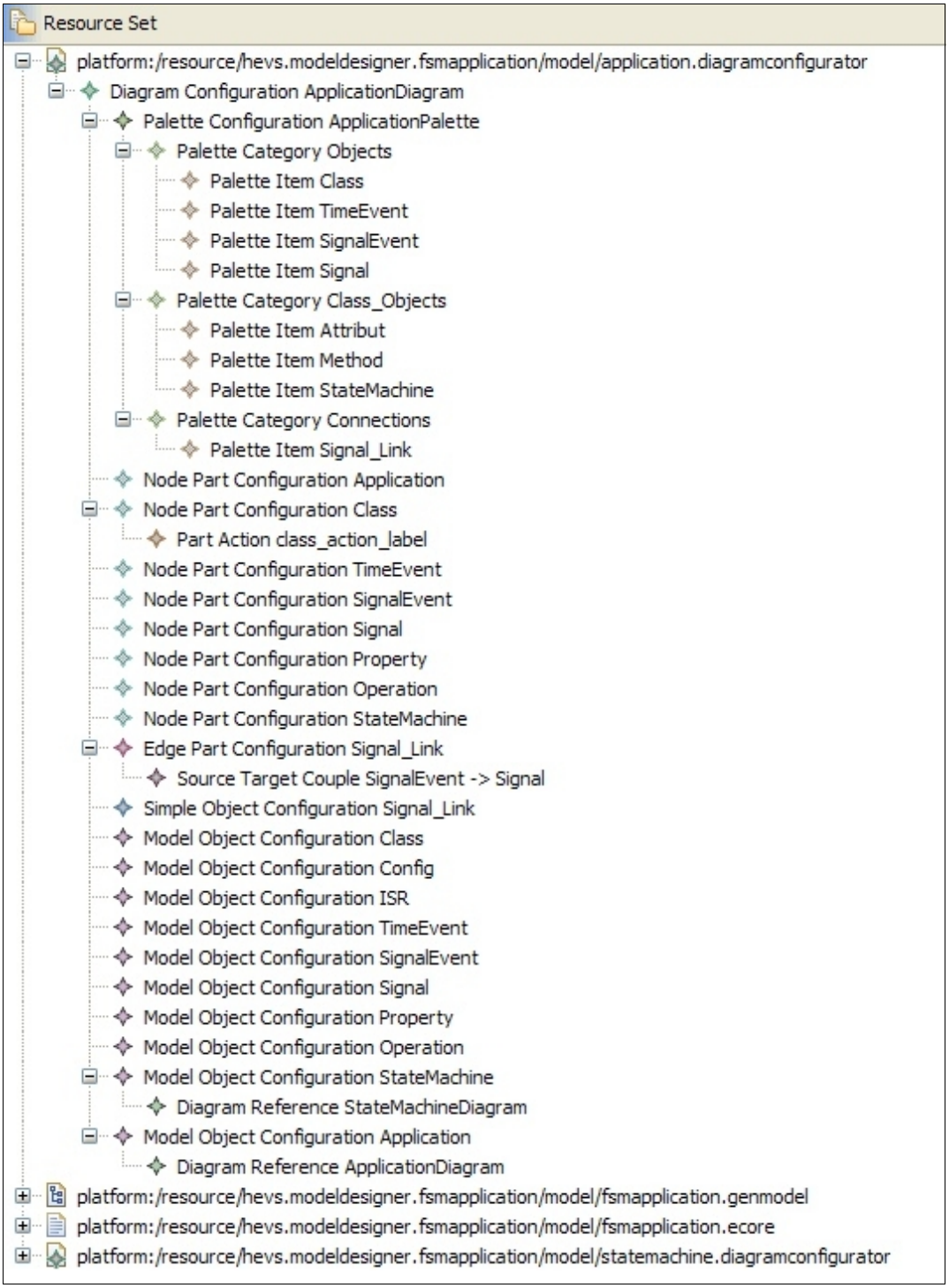


Illustration 3: Vue générale de application.diagramconfigurator

On peut apercevoir que ce **diagramconfigurator** possède deux **Diagram Reference** .

Afin de pouvoir placer le **Diagram Reference StateMachineDiagram** il faut auparavant avoir configuré le **statemachine.diagramconfigurator** .

statemachine.diagramconfigurator

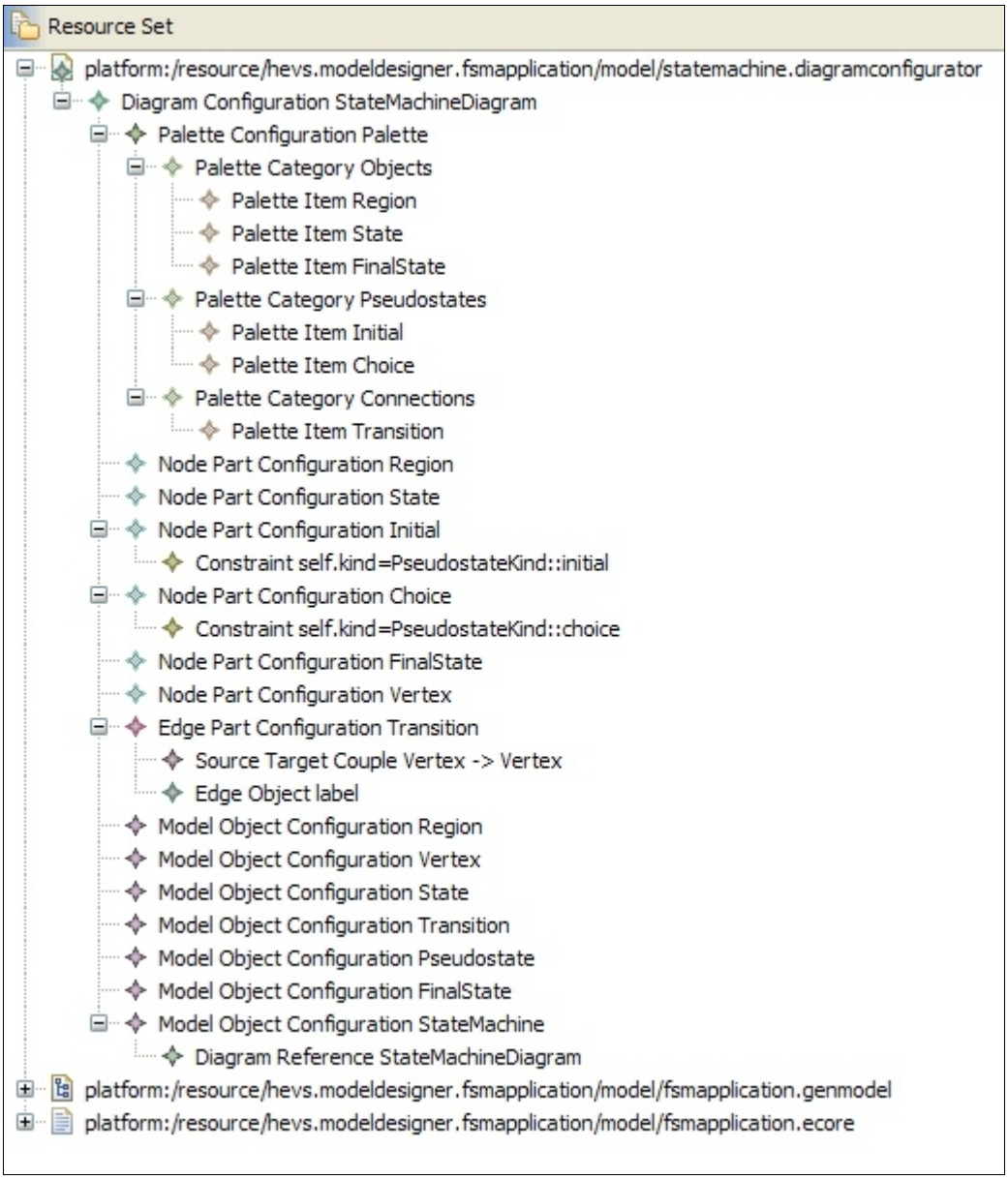


Illustration 4: Vue générale de statemachine.diagramconfigurator

Dans cette dernière figure on peut apercevoir l'utilisation d' **OCL**[1] pour définir deux sortes de **Node Part** différentes, un **Node Part** pour un **Pseudostate** avec la valeur **kind** qui vaut **initial** et un **Node Part** pour un **Pseudostate** avec la valeur **kind** qui vaut **choice** .

Dans ce diagramconfigurator il n'existe qu'un Diagram Reference car ce diagramme ne peut pas contenir un diagramme d'application. Le diagramme d'application peut , quant à lui, contenir un diagramme de statemachine.

1 OCL : Object Constraint Language, largement utilisé dans la définition du métamodèle UML

7.6 Architecture MVC utilisée dans l'éditeur graphique

Avant de customiser l'éditeur graphique il est préférable de comprendre le concept de fonctionnement de l'architecture Model-View-Controller.
Cela permet de mieux analyser et comprendre le comportement de l'éditeur lors de la modification du code.

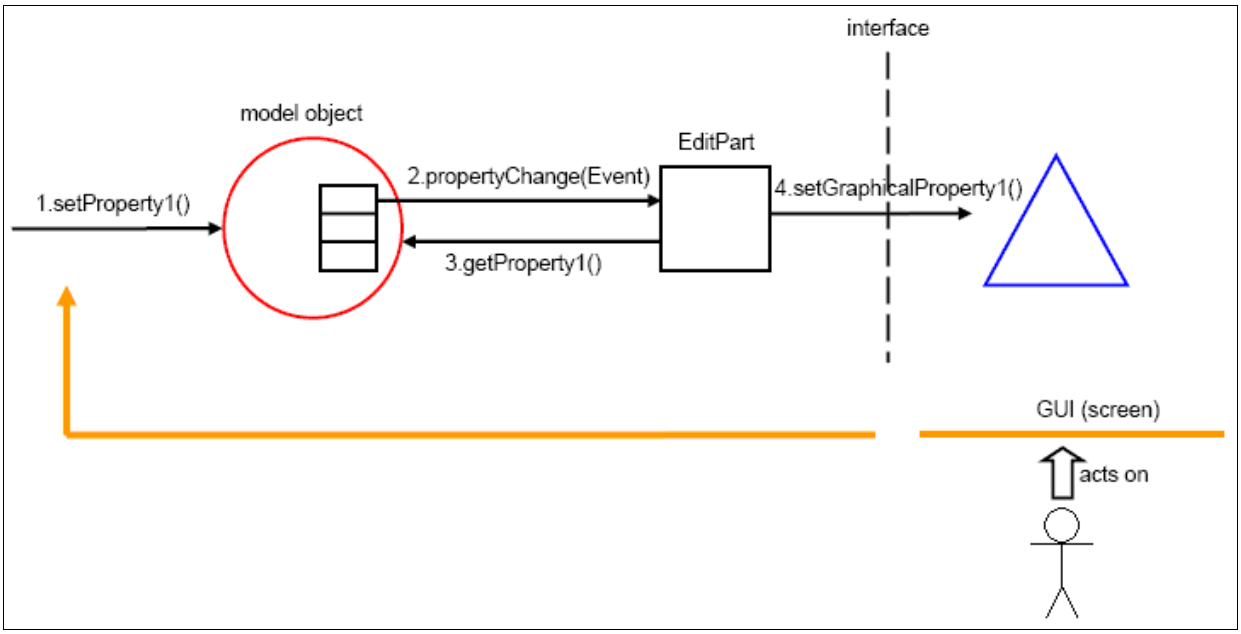


Illustration 5: Exemple d'architecture MVC

L'architecture MVC permet de séparer le model de la vue. La logique concernant l'éditeur est placée ailleurs et le code du model reste donc très propre. Cela s'applique aussi au niveau de la vue d'ailleurs.

En faite, chaque partie de l'éditeur à un rôle particulier très bien défini. Le modèle enregistre les données, la vue affiche les données et le contrôleur implémente la connection entre les deux.

7.7 Customisations

7.7.1 Fichiers, dossiers et leur rôle

Les rôles de différents fichiers générés sont expliqués ici. Il est ainsi plus facile de réaliser les modifications et customisations dans le code de l'éditeur.

Les informations sont séparées par parties. Chaque partie fait référence à un plugin différent. Le projet de création de l'éditeur en arborescence est dénommé ici Projet Genmodel, les projets générés à partir des diagramconfigurator sont dénommés Projet GraphicalEditor. La partie des noms donnés ci-dessous qui est en format italique désigne les noms de différents éléments du projet. Ceux ci peuvent être des noms de Classe, des noms de Nodes ou le nom du projet.

7.7.2 Fichiers dans le Projet Genmodel

Fichiers `elementSwitch.java`

Ces fichiers contiennent un système permettant le switch de type de l'object afin que chaque object puisse être appréhendé par tous les types dont il est la spécialisation.

Fichiers `elementItemProvider.java`

Ces fichiers contiennent les references vers les `utils` qui ont un rapport avec l' *element* concerné. Cela peut être un `ItemPropertyDescriptor`, le texte qui peut être affiché sur le **Label** ou une notification qui met à jour les **Children**. Il contient aussi le lien vers l'image utilisée dans l'éditeur **tree editor** (éditeur en arborescence) et retourne cette Image.

7.7.3 Projet GraphicalEditor

Fichiers en dehors des dossiers

Le nom de ces fichiers est généralement composé du prefix (ajouté auparavant dans la configuration) suivi de leur fonction. Ainsi :

FsmAppSimpleObjectConstants	s'occuppe des propriétés des Simple Objects[2]
FsmAppImageRegistry	s'occupe de fournir les informations relatives aux images (ressources)
etc ...	

Fichier `element`Configuration.java

`element` ici est le nom donné au plugin. Ce fichier met à disposition les **ICreationUtils** , **IPaletteManager** , **EditPartFactory** . Il utilise aussi le `EditPart2ModelAdapterFactory.java` pour enregistrer un adaptateur pour chaque type utilisé dans le diagramme. Il représente la configuration du diagramme.

Fichier `element`PaletteManager.java

La palette de l'éditeur graphique est faite à partir de ce fichier.

Fichier `element`CreationUtils.java

Ce fichier permet la création des éléments graphiques selon l'object du modèle qui lui est transmis.

Dossier `Policies`

Chaque node configuré en tant que **container** va générer un fichier `node`LayoutEditPolicy.java . Ce fichier contient les contrôles afin de valider ou pas la création d'un **node** dans ce conteneur.

A l'ntérieur de ce dossier on trouve aussi les fichiers `edge`EdgeCreationEditPolicy.java . Ces fichiers sont au même nombre que les **edges** configurés dans le **diagram configurator** . Ils contiennent les contrôles permettant la validation de la création d'un **edge** dans l'éditeur graphique.

2 Simple Objects : ce sont des objets graphiques qui ne sont pas associés à des éléments du modèle

7.7.4 Nodes et Figures

Afin de déterminer la figure que tel ou tel node doit employer on utilise le **diagram configurator** . Topcased nous propose que quelques figures dans les choix du **diagram configurator** . Afin de mieux customiser l'apparence d'un node, le choix de la figure **Graphic with Label** s'avère très utile.

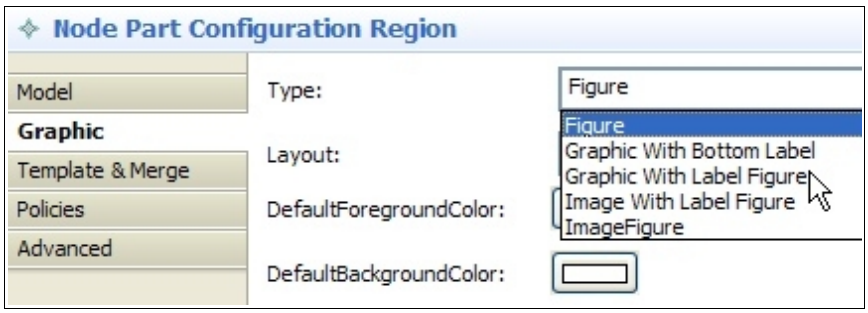


Illustration 6: Choix de figures

Cette figure implemente déjà l'interface **ILabelFigure** qui permet la gestion de labels, il suffit alors d'écraser les méthodes afin d'obtenir l'affichage désiré. Il est aussi possible de créer une classe qui implémente les méthodes de l'interface

7.7.5 Icônes

Les icônes utilisées par l'éditeur en arborescence sont par convention d'une taille de 16x16 pixels et de type gif. Par défaut, leur emplacement est le dossier ***icons***. Le dossier ***icons/ctool16*** contient les icônes qui sont affichées dans les menus **Create Child** et le ***icons/obj16*** contient celles présentes sur l'arborescence. Ces icônes sont aussi reprises pour l'affichage sur la fenêtre **Outline** , il est donc judicieux de les modifier afin d'avoir une meilleure représentation du modèle.

Les classes qui les chargent sont les classes `elementProvider.java` , dans le dossier **provider** généré

Les icônes utilisées par l'éditeur graphique dans, par exemple, la palette peuvent être ajoutées dans le dossier **icons** de ce projet. Leur nom et chemin d'accès est spécifié dans le fichier `images.properties` . Pour ces icônes il est possible de fournir une version 24x24 pixels de chacune des icônes pour permettre un affichage optimal lorsque l'utilisateur choisit "Use large Icons" dans la palette.

```
/**  @generated NOT
 *  MODS: GEN DONE -PseudostateItemProvider ->gets the right image to display
 */
@Override
public  Object getImage(Object object) {
if  (object instanceof Pseudostate) {
   if ( ((Pseudostate)object).getKind()==PseudostateKind.INITIAL )   {
       return  overlayImage(object,
                                getResourceLocator()
                                .getImage("full/obj16/Pseudostate_Initial"));
   }
   else
   if ( ((Pseudostate)object).getKind()==PseudostateKind.DEEP_HISTORY )     {
              return  overlayImage(object,
                                getResourceLocator()
                                .getImage("full/obj16/Pseudostate_DeepHistory"));
   }
   else  ...
```

Code 1: exemple sélection d'icône à afficher dans l'arborescence

7.7.6 Labels

Les labels sont gérés différamment selon leur type. Ainsi, le label d'un **node** du diagramme doit être customisé à l'intérieur de la classe représentant la figure. La figure doit implémenter l'interface **ILabelFigure** pour que le label puisse être rafraîchi lorsque des changements sont effectués sur le modèle (lorsque l'élément est renommé par exemple).

Le label d'un **edge** doit lui, être instancié par un **Edge Object** dans le **diagramconfigurator** :

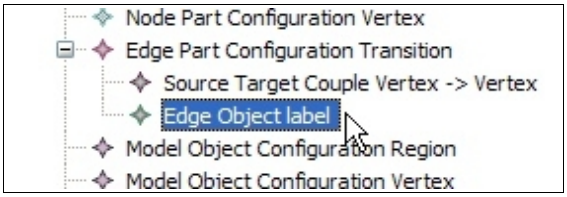


Illustration 7: Edge object dans diagramconfigurator

diagram.graphconf

Ce fichier contient des informations concernant la représentation graphique des éléments dans un diagramme donné. Il est conseillé de ne pas modifier directement le fichier mais de réaliser les changements nécéssaires directement dans le **diagram configurator** et de générer à nouveau le plugin. L'option `overwriteGraphConf` doit être sur `true` pour pouvoir écraser les réglages précédents.

7.7.7 Supprimer les Child Elements du Child Menu

L'éditeur graphique généré par défaut à partir du métamodèle va afficher dans le **Child Menu** tous les éléments enfants du type choisi. Cependant, que quelques-uns de ces éléments seront traités. Afin de restreindre le choix, le code de la méthode `collectNewChildDescriptors` des classes `elementItemProvider` dans le dossier **Provider** doit être modifié.

```
protected  void collectNewChildDescriptors(Collection<Object> newChildDescriptors,
                                           Object object) {
 super .collectNewChildDescriptors(newChildDescriptors, object);

    newChildDescriptors.add
    (createChildParameter
         (FsmapplicationPackage.Literals.STATE_MACHINE__REGION,
     FsmapplicationFactory.eINSTANCE.createRegion()));

    newChildDescriptors.add
    (createChildParameter
         (FsmapplicationPackage.Literals.STATE_MACHINE__CONNECTION_POINT,
     FsmapplicationFactory.eINSTANCE.createPseudostate()));
}
```

Code 2: Suppression de child elements

Chaque objet dans la collection `newChildDescriptors` correspond à une entrée dans le menu **Create child** , il suffit donc de les supprimer ou commenter.

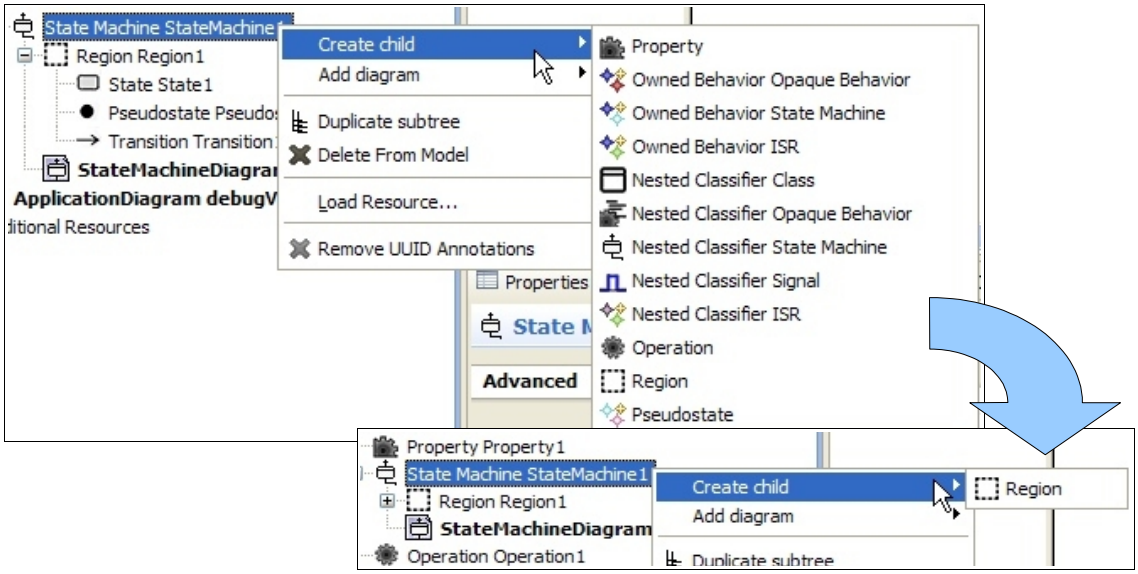


Illustration 8: Tri de child elements superflus

Reprennant l'exemple du code d'auparavant, aprés avoir supprimé les lignes contenant l'ajout des éléments enfants de l'élément parent et l'ajout de l'élément **Pseudostate** , la figure ci-dessus illustre les changements.

8 Design Patterns & Conception

8.1 State Pattern

Lors de la préparation du présent travail, j'ai cherché des informations sur le **state pattern** . Ce fameux pattern rédigé par le GOF (Gang Of Four) permet d'implémenter une machine qui changerait son état selon les événements.

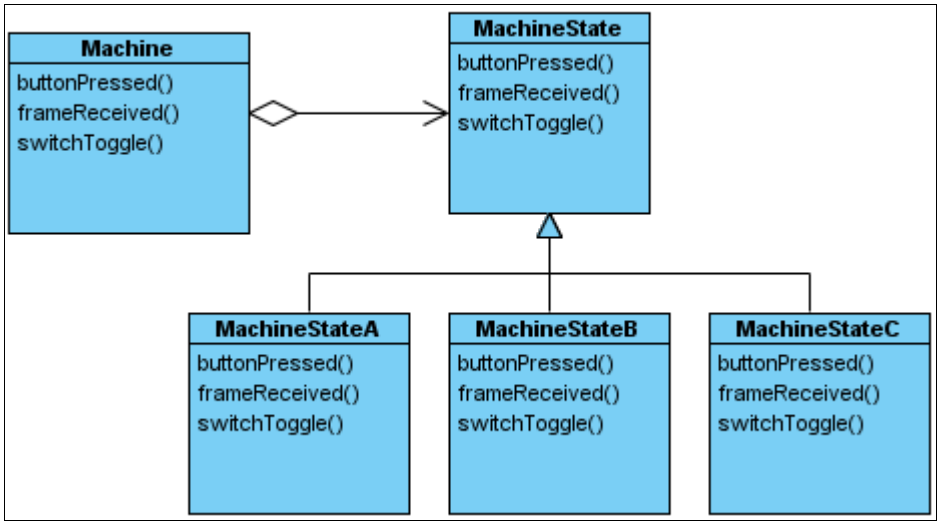


Diagramme 5: State pattern - exemple

Dans ce pattern, la classe **Machine** référence une instance d'une sous-classe de **MachineState** qui représente le comportement de la machine. La classe **MachineState** est donc un interface pour les classes **MachineStateA** , **MachineStateB** et **MachineStateC** qui représentent les différents états de la machine. Grâce à cette conception, il est possible de placer le comportement correspondant à chaque état dans une classe différente.

Cependant le pattern est destiné à des languages orientés objet. Cela rend l'implémentation en C difficile car il faut donc réaliser une programmation pseudo-orientée object. De plus, les cibles selon le cahier de charges sont des petits processeurs. Des opérations superflues dégraderaient très vite les performances.

8.2 Implémentation avec une structure switch

Lors de mon dernier travail, j'ai du choisir un algorithme pour l'implémentation d'une très simple machine d'états. Les informations que j'avais alors recueilli m'ont fait opter pour une implémentation sous la forme de switchs car cette solution intuitive me semblait la plus fiable et aussi la plus simple. Lors de l'implémentation, j'ai eu l'occasion de méditer sur plusieurs concepts d'amélioration et d'ajout de fonctionnalitées.

J'ai trouvé que générer la fonctionnalitée sous forme d'un switch était une bonne solution aussi car ce que l'on reproche souvent à cette approche est le fait que le code est difficile à maintenir. Vu que le code est généré automatiquement, je trouve que cet argument ne fait plus le poids. De plus, une implémentation de ce type est très séquentielle. Avec une fonctionnalitée bien structurée et organisée, on peut facilement modifier les scripts de transformation afin d'y ajouter de nouvelles possibilitées.

8.2.1 Fichiers de l'application

Les fichiers composant l'application une fois le code généré sont :

- un header de l'application
- un fichier C de l'application
- un header par classe
- un fichier C par classe
- un fichier XHALL
- un fichier XF

Le header d'application doit étendre la visibilité de certains éléments, variables ou fonctions à tous les autres fichiers.

Le fichier C de l'application implémente les fonctions possédant une visibilité globale, il contient aussi la méthode `main`.

Les fichiers header et C des classes contiennent les propriétées implémentées dans le modèle. Elles seront déclarées dans le header ou dans le C selon leur visibilité.

Le fichier XHALL qui contient l'interface vers les périphériques et qui permet l'initialisation du timer de XF.

Le fichier XF qui contient la pile des évènements, l'implémentation avancée du timer gérant les **TimeEvent**.

8.2.2 Structure à générer

Le model aboutit sur les fichiers application et class, le fichier header et c de l'application ne sont générés qu'une fois chacun alors que les fichiers class peuvent être multiples selon le nombre de classes créées dans l'éditeur graphique.

Dans la méthode `main` du fichier application le XF est initialisé via une commande mise à disposition par le XHALL. Les fichiers class peuvent dés lors utiliser le XF.

La méthode `main` va ensuite appeller la **behavior** spécifiée pour chaque classe. L'appel de **behavior** va être réalisé séquentiellement. Pour enclencher une machine d'états il faut donc avoir l'avoir définie en tant que comportement d'une classe. Cependant, lorsqu'un comportement de machine d'état est lancé, si un deuxième comportement est présent dans le corps de main, il ne sera pas atteint tant que toutes les régions de la machine d'état n'auront pas atteint leur `FinalState`.
Le fichier application comporte aussi la structure d'events utilisé par les éléments Class.

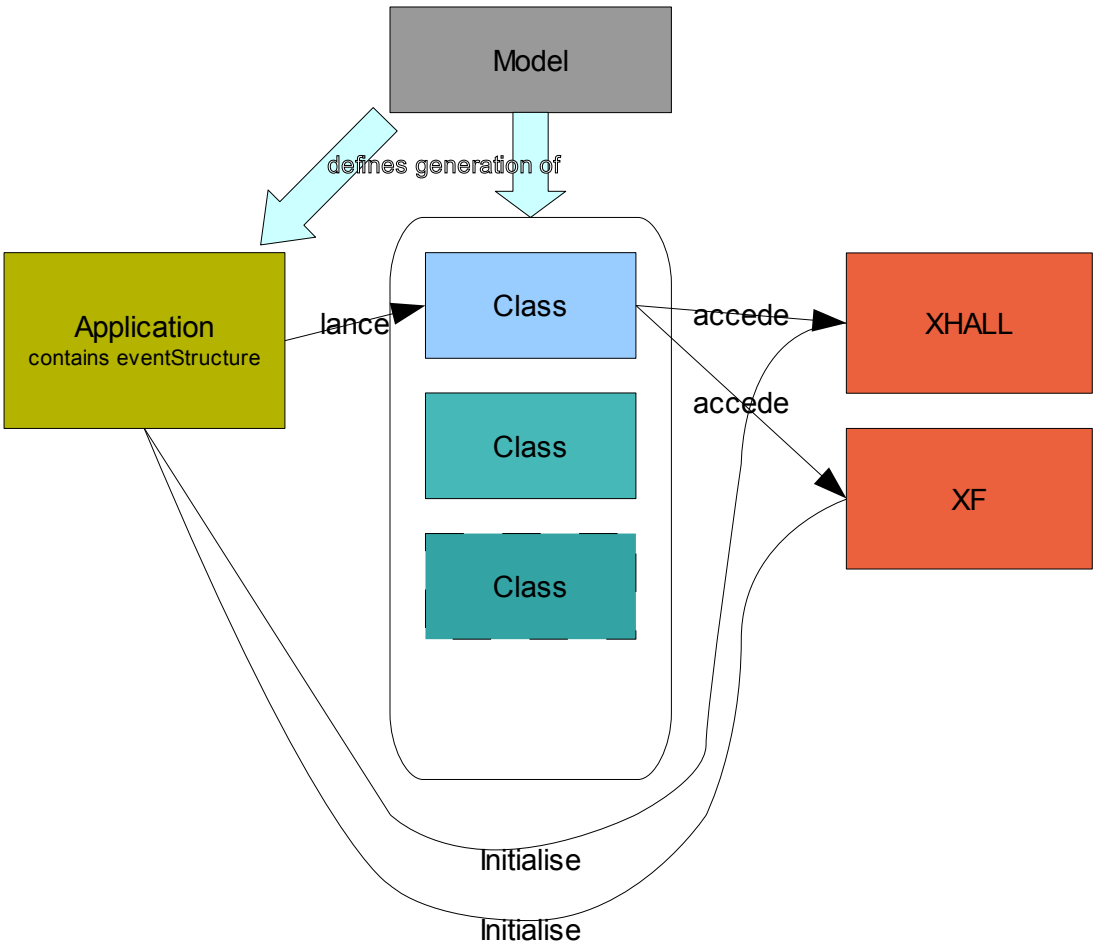


Schéma 2: Fonctionnement simplifié

8.2.3 Le fonctionnement de la machine d'états

Lors du lancement de l'application, toutes les variables globales vont être initiés. On rentre ensuite dans la fonction `main` du fichier c de l'application.

Le **XF** et autres variables éventuellement nécéssaires sont initialisées.

Les fonctions correspondant au comportement de chaque classe sont alors appellées. Les fonctions appellées n'ont pas un ordre précis. L'ordre dans lequel elles apparaissent est lié à l'ordre dans lequel le générateur de code a parsé le modèle.

Une machine d'état est implémentée sous la forme d'une fonction.

Pour pouvoir éxécuter une machine d'état, il faut donc l'avoir spécifiée en tant que comportement de classe.

Les fichiers des classes contiennent des attributs, et des fonctions. La valeur de la propriété `Visibility` définit l'emplacement de la déclaration des variables. La présence du prototype d'une fonction est aussi définie par ce champ.

Une fonction représentant une machine d'états comprend une boucle à son intérieur. La boucle va continuer d'itérer jusqu'à ce que toutes les régions aient atteint un `FinalState`.

Dans le cas où toutes les régions ont atteint leur FinalState, la boucle va finir son itération et la fonction va pouvoir finir son éxécution. A ce moment, si le main comporte d'autres appels à des fonctions, il va les éxécuter.

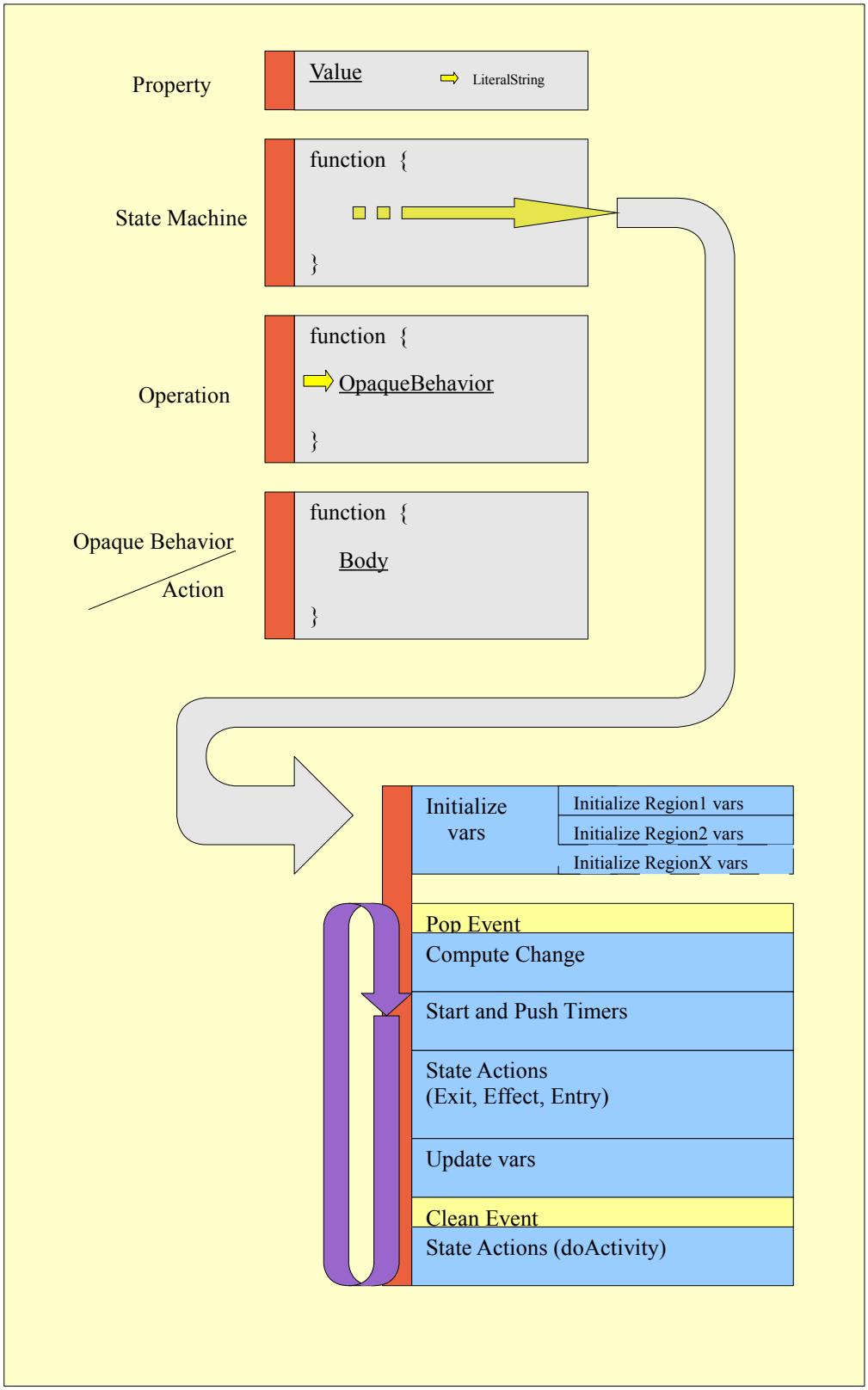


Schéma 3: Structure la fonction StateMachine

8.2.4 Fonction StateMachine

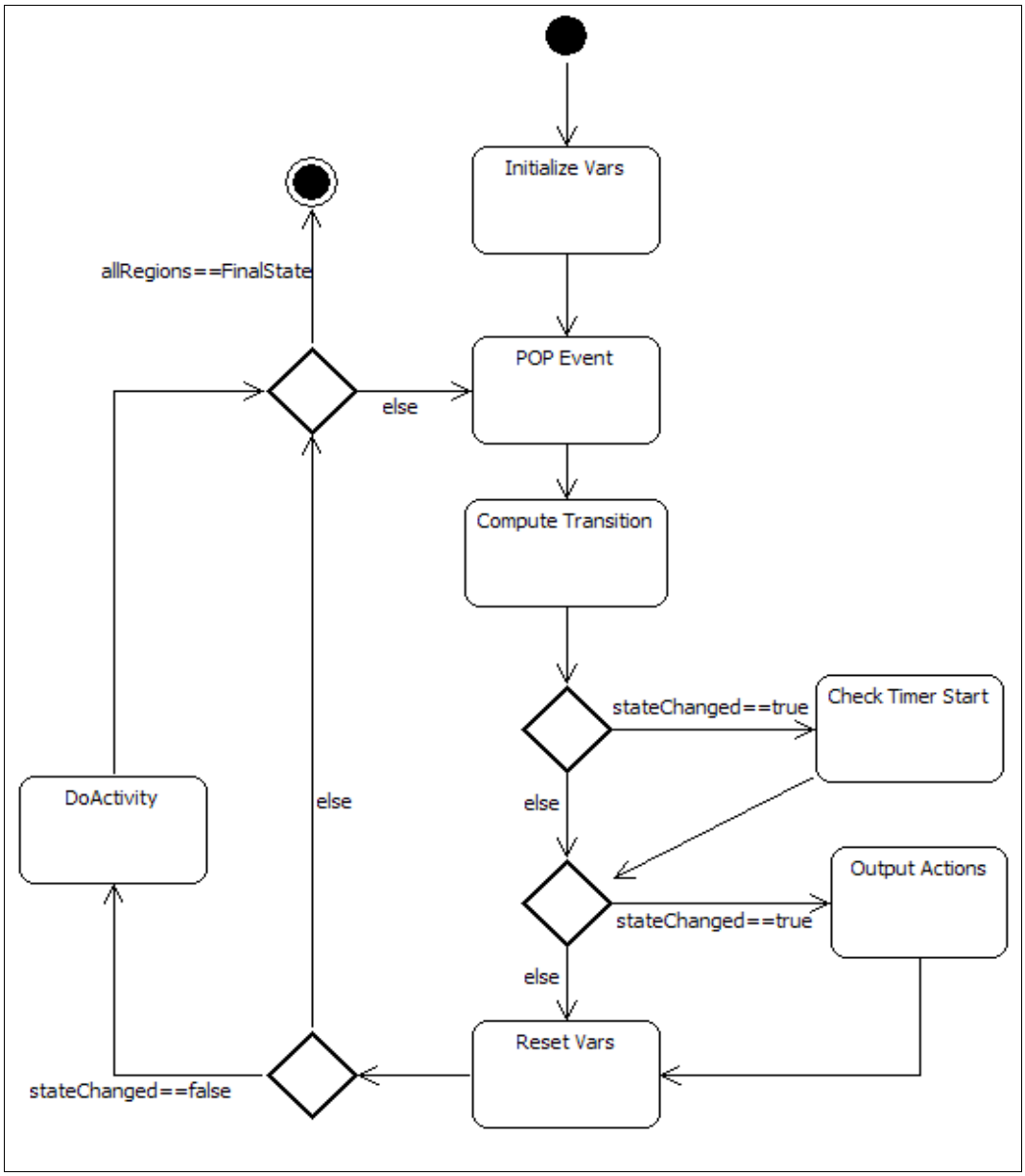


Diagramme 6: Fonction StateMachine

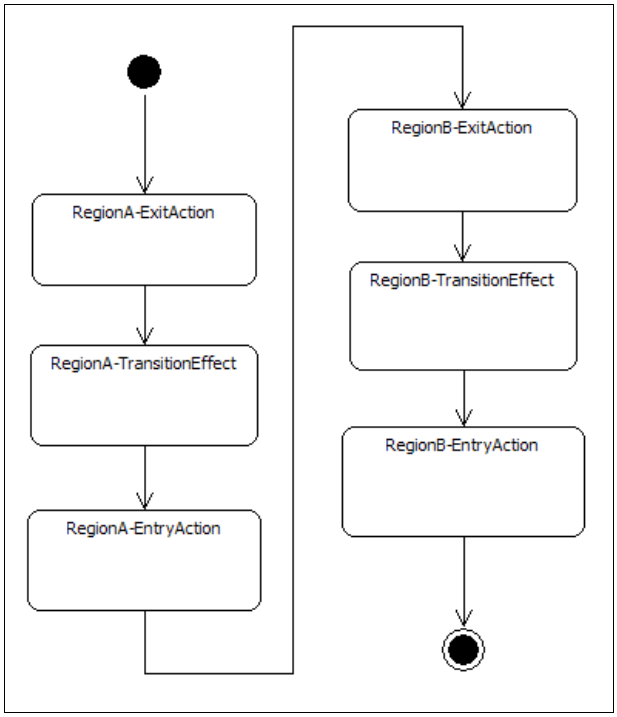


Diagramme 7: Output Actions
Regions en séquentielle

Les diagrammes précédents définissent une fonction comportant une machine d'états. Une machine comportant plusieurs régions va réagir séquentiellement, comme on peut voir sur le diagramme 7 représentant l'état Output Actions du diagramme 6, l'éxécution sur les différentes régions est cependant dans le même état. Toutes les régions auront été parcourrues avant ce que l'on décrirait comme un changement d'état dans le digramme 6.

8.2.5 Temps limite de vie des valeurs passées aux signaux

Les valeurs passées en paramètre aux signaux lors de la génération d'évènements ne sont stockées que pour un cycle (**Exit Action** , **Effect** et **Entry Action**) de la machine d'état, si le dévelloppeur prévoit de les employer dans un temps ultérieur, il doit prévoir de les stocker.

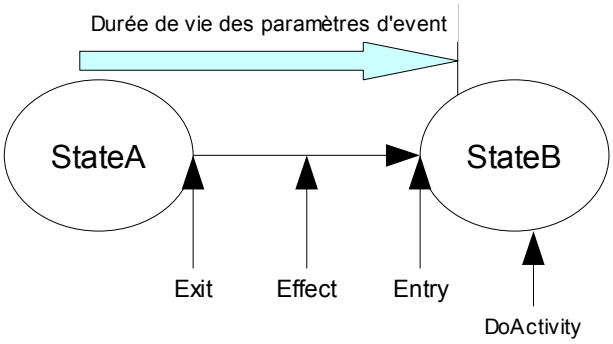


Schéma 4: Durée de vie des paramètres

8.2.6 Structure events

Pour la gestion des **events** une structure telle que décrite ci-dessous est créée :

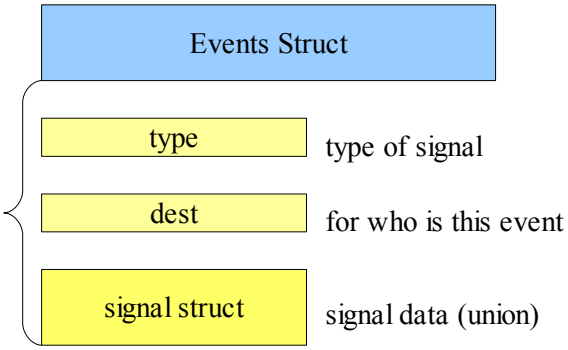


Figure 1: Structure utilisée pour gérer les events

Cette structure contient trois champs :

- **type** : il contiendra la valeur correspondant dans une énumération au type de signal utilisé.
- **dest** : il contiendra la valeur correspondant dans une énumération à la transition dont l'event est le trigger. Ce champ n'est valable que pour les **TimeEvent**
- **signal struct** : ici est stockée la structure correspondant aux données du signal. Ce champ est de type `union` et peut donc prendre n'importe laquelle des structures signal.

8.3 XF et XHALL

L'abstraction du matériel se fera par le biais d'un fichier XHALL.

La gestion des évènements sera implémentée dans un fichier XF. Ce fichier mets à disposition du model les fonctions evPush, evPop, et tmPush.

XF dispose aussi de la fonction Tick(). Cette function est appellée à chaque fois que lorsque l'on rentre dans la routine d'interruptions l'on détecte que la source de l'interruption est le timer(timer hardware de la cible) qui est dédié au XF.

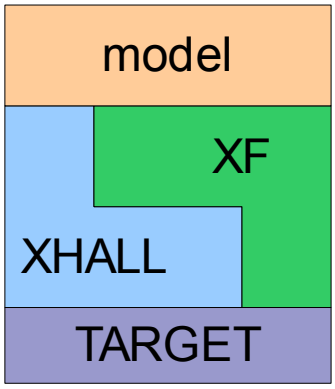


Figure 2: Abstraction du matériel

L'initialisation de XF est réalisée par un appell de XF_init() présente dans le XHALL.

Le fichier XF devra être totalement indépendant de la cible.

9 Dévellopement & Règles de Transformation

Pour la génération de code, MDWorkbench est utilisé. Voici quelques informations concernant les languages mis à disposition par cet outil.

9.1 Informations Utiles

9.1.1 Text Generation Language (TGL)

Ce language sert à définir des templates pour la génération de code. Il utilise des balises pour différencier le texte statique du code dynamique.

Le texte statique est généré tel quel en sortie alors que le texte dynamique va être remplacé par les valeurs calculées. Les directives sont des tags, ils ne sont pas générés en sortie. Les commentaires ne sont pas non plus générés en sortie.

```
[#template GenerateJavaFile(class : uml20.Class)]
[#file]${class.name}.java[/#file] [#-- the file where to write contents --]
public class ${class.name} {

    public String toString() {
        return "${class.name} instance";
    }
}
[/#template]
```

Code 3: Exemple TGL

Pour une classe `Toto`, ce template générerait un fichier `Toto.java` avec à son intérieur :

```
public class Toto {

    public String toString() {
        return "Toto instance";
    }
}
```

Code 4: Résultat de la génération d'exemple TGL

9.1.2 Model Query Language (MQL)

C'est le langage permettant de gérer les modèles. Il est utilisé dans les rule sets qui permettent de modifier les modèles et de lancer la génération de texte.

```
package com.mycompany.example;

// expects a loaded UML 2.0 model as input
public ruleset GenerateAllJavaClasses(in model : uml20) {

    public rule generate() {

        // loop on each Class of the UML 2.0 model
        foreach (class in model.getInstances("Class")) {

            // calls the text template GenerateJavaClass
            $GenerateJavaClass(class);
        }
    }
}
```

Code 5: Exemple de MQL

9.1.3 Extension du fichier contenant le modèle

Un modèle créé avec Topcased a par défaut l'extension de fichier donnée au plugin créé. MDWorkbench ne connait pas cette extension, pour ouvrir le modèle il faudras donc toujours lui forcer à ouvrir ce fichier. Afin de palier à ce désagrément, deux méthodes permettent à MDWorkbench de reconnaitre le fichier avec le modèle au format XMI créé par Topcased.

La première consiste à ajouter dans le fichier `metamodels.xml` la reconnaissance des fichiers avec l'extension désirée. Ce fichier est généré lors de l'import du métamodèle dans MDWorkbench. Il est dans le dossier **src** présent à la racine du projet généré. Cela permettra à MDWorkbench de reconnaître les fichiers avec l'extension que TOPCASED ajoute par défaut.

La deuxième méthode permet de modifier l'extension que TOPCASED ajoute aux fichiers comprenant les modèles. Lors de la configuration du fichier `.genmodel` , dans le **genpackage element** à **Model - File Extensions** spécifiez ***xmi*** (ou une autre extension désirée). *(testé sur TOPCASED version 2.1)*

9.2 Implémentation

9.2.1 XHALL

Le fichier d'abstraction de matériel (xhall.h) est implémenté.
Les premiers test portant sur :

Le PORT B

L'interface LEDS

ont abouti, d'autres tests seront réalisés ultérieurement dès que d'autres composants seront prêts. (XF, générateur de code)

9.2.2 XF

Les fichiers de l'execution framework sont implémentés. Les premiers tests ont été réalisés à l'aide de MSYS. XF doit être indépendant de la cible, il devra donc être fonctionnel sur la cible sans avoir à corriger de code.

9.2.3 Fichiers Générés

Une fois le lecteur de metamodel généré et exporté (ne pas oublier de redémarrer MDWorkbench pour l'activer), on définit les fichiers qui vont être créés.

```
package c.generator;

public ruleset CGeneration(in model : fsmapp) {

   public rule generateC() {
      foreach (application : fsmapp.Application in model.getInstances("Application")) {
         //Application template
         $appFile(application);
         $appHeader(application);
         foreach (class : fsmapp.Class in application.theClass)  {
            //Class template
            $classFile(class);
            $classHeader(class);
         }
      }
   }
}
```

Code 6: Fichiers qui vont être générés

Le générateur de code va générer un fichier **.c** du nom de l'application (donc `application.c`) ainsi que son **header** . Il génère aussi un fichier **.c** et **header** pour chaque classe.

Les fichiers templates sont prêts :

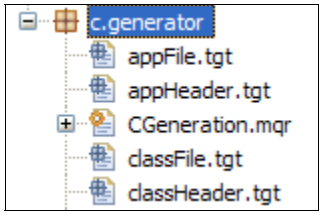


Illustration 9: Templates

9.3 Les règles de transformation

Remarque

La description complète de l'implémentation des règles de transformation étant très longue et prenant énormément de place, je me contenterai de spécifier la procédure d'implémentation que j'ai suivi. La totalité des règles de transformation est présente en annexe.

9.3.1 Procédure d'implémentation

Templates avec le texte statique

La première étape consiste à remplir les templates avec le texte statique. Des balises améliorent la présentation du code et permettent de contrôler plus facilement le code généré. Il est donc conseillé d'en placer. *(exemple : code 7)*

Ajout des transient links

Ajouter des transient links un peu partout dans les scripts est dangereux. (expérience vécue ;)). La meilleure solution consiste à placer les transient links dans un script MQL qui va être appellé tout au début du premier fichier à être généré (parsé). Lorsque vous avez besoin d'une information et que vous voulez l'obtenir grâce à un transient link, il suffit de l'ajouter dans ce script et vous pourrez dés lors en avoir accès partout.

Les appels de scripts pas encore implémentés depuis le template.

D'autres méthodes existent, cependant une technique d'implémentation maligne consiste à placer des appels à des scripts TGL avec un nom très expressif (je veux dire que leur nom permet de savoir ce que ces scripts vont générer) et ensuite de les implémenter. Il est possible de faire cela à plusieurs niveaux si votre implémentation est complexe.

```
[#package c.generator]

[#template public appHeader(application : fsmapp.Application)]
[#file]generated/${application.getHeaderFileName()}[/#file]
/*
*****************************************************************************
* This is a generated file (from a fsmapplication model)                    *
*****************************************************************************
*                                                                           *
* file      : ${application.getHeaderFileName()}                            *
* template  : appHeader.tgt                                                 *
*                                                                           *
* Info      : This file implements the interface of the main file of the    *
*          application.                                                     *
*                                                                           *
* Author : Hugo Nunes                                                       *
* Date      : 2008-11-06                                                    *
* Version   : 1.0                                                           *
*                                                                           *
*****************************************************************************
*/
#ifndef ${application.formApplicationName()}_H
#define ${application.formApplicationName()}_H
//---------------------------------------------------------------------------
// INCLUSIONS ---------------------------------------------------------------
//---------------------------------------------------------------------------
#include "xhal.h"
#include "xf.h"


//---------------------------------------------------------------------------
// PREPROCESSOR -------------------------------------------------------------
//---------------------------------------------------------------------------


//---------------------------------------------------------------------------
// ENUMERATIONS -------------------------------------------------------------
//---------------------------------------------------------------------------



//---------------------------------------------------------------------------
// TYPE DEFINITIONS ---------------------------------------------------------
//---------------------------------------------------------------------------



//---------------------------------------------------------------------------
// VARIABLES DECLARATIONS ---------------------------------------------------
//---------------------------------------------------------------------------



//---------------------------------------------------------------------------
// FUNCTIONS PROTOTYPES -----------------------------------------------------
//---------------------------------------------------------------------------
void interrupt ISR(void);



//---------------------------------------------------------------------------
#endif
[/#template]
```

Code 7: Templates avec texte statique

9.3.2 Les signaux et les events

Signaux

L'éditeur permet au dévellopeur de définir son signal qui peut ensuite être utilisé pour générer un ou plusieurs types de **SignalEvent** . Le signal peut comporter plusieurs propriétées.

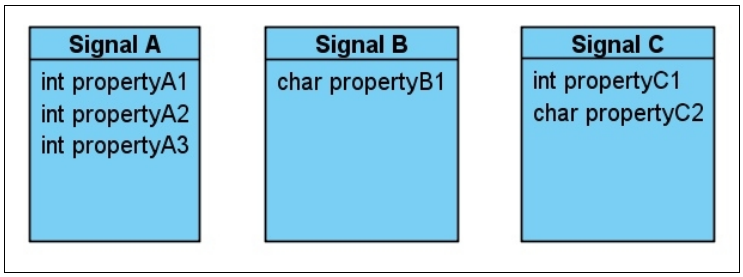


Diagramme 8: Exemple de signaux à implémenter

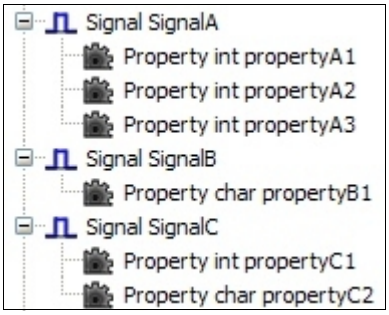


Illustration 10: Signaux définis dans l'éditeur

```
[#script public typewriteAllSignalStructs]
[#foreach signal : fsmapp.Signal in self.getThePackagedElement()
                                  →.select("getTypeName","Signal")]
 struct ${signal#formName.first()}${self.separationClassString()}struct{
        [#foreach property : fsmapp.Property in signal.getOwnedAttribute()]
   ${property.attributeModifiers()} ${property.attributeName()};
        [/#foreach]
};
[/#foreach]
[/#script]
```

Code 8: Génération des structures des signaux

```
//Signals
struct APP1_SignalA_struct{
   int propertyA1;
   int propertyA2;
   int propertyA3;
};
struct APP1_SignalB_struct{
   char propertyB1;
};
struct APP1_SignalC_struct{
   int propertyC1;
   char propertyC2;
};
```

Code 9: Structures Signaux (généré)

Les propriétés de la structure définie pour le signal permettent au dévelloppeur, lors de la génération d'évènements dans son implémentation, d'y placer des valeurs en tant que paramètres et de les récupèrer ensuite ailleurs dans la machine d'états.

Générer un événement

La commande permettant de générer un événement lors de l'éxécution du code est :

```
GEN(signal(parametervalue1,parametervalue2,...))
```

Cette commande n'est en effet qu'une directive de preprocessor déclarée dans le header de l'application :

```
#define GEN( x )  do {  \
                        ENTER_CRITICAL();  \
                        APP1_PUSH_##x;     \
                        LEAVE_CRITICAL();  \
                      } while (0)
```

Code 10: Define de commande pour la génération d'évènements

Cette directive concatène le code entré en tant que paramètre avec un prefix variable selon le nom de l'application et au moment de la compilation fait appel à des fonctions qui vont générer l'évènement.

Timer

Le timer est une structure comme celles des signaux, elle n'est par contre pas modifiable par le dévelloppeur. En faite cette structure a une forme définitive et sert à uniquement à transiter par le **XF** afin de déclencher un **TimerEvent** . En ajoutant les directives suivantes entre les deux balises fermantes du code donné précédémment on génère directement la structure pour le timer.

```
...
[/#foreach]
struct ${self.formTimerSignalString()}${self.separationClassString()}struct{
   int index;
   int value;
};
[/#script]
```

Code 11: Génération de la structure timer

```
struct APP1_timerSignal_struct{
   int index;
   int value;
};
```

Code 12: Structure timer (généré)

Cette structure est reprise pour chaque **push** dans le **XF** car un timer ne prends pas d'autres paramètres si ce n'est que la valeur de temps à laquelle le TimeEvent va se déclencher. (la propriété index est d'ailleurs superflue)

Lorsque l'un événement à été remis par le XF, lors de la vérification de changement d'état, le contrôle est effectué différamment selon que l'évènement attendu est soit un **TimeEvent** soit un **SignalEvent** .

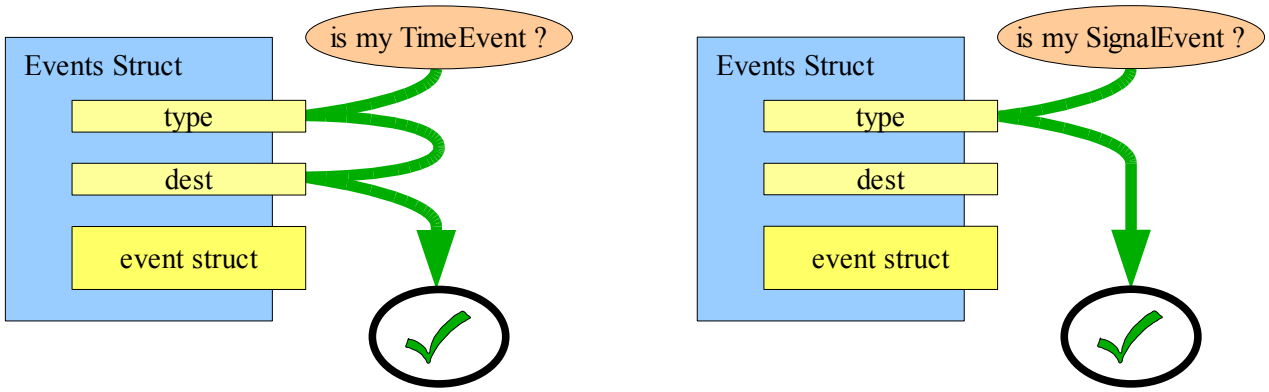


Figure 3: Contrôle event

Comme on peut le voir sur la figure précédente, pour un **event** de type **TimeEvent**, on doit réaliser un deuxième contrôle afin de valider la transition. Un **SignalEvent** ne nécéssite qu'un seul contrôle (contrôle que le type de signal correspond au type affecté dans l'éditeur graphique).

10 Remerciements

Je aimerais adresser un grand merci à toutes les personnes suivantes pour le temps qu'ils m'ont accordé, les conseils, l'aide à la réalisation de ce travail et les connaissances qu'ils ont partagé avec moi.

M. Medard Rieder	Professeur à la HES-SO Valais
M. Rico Steiner	Assistant scientifique à la HES-SO Valais
M. Philippe Pralong	Assistant scientifique à la HES-SO Valais
M. Thomas Sterren	Assistant scientifique à la HES-SO Valais

Tous mes camarades diplômants...

11 Conclusion

Ce travail arrivant à son terme, je suis satisfait des connaissances acquises et j'espère continuer dans ce domaine. Ce travail m'a particulièrement plu de par son sujet et aussi les possibilitées que ce domaine présente. Je regrette cependant ne pas avoir réussi à faire davantage de tests et de debug dans le temps imparti. Beaucoup du code implémenté n'as pas été optimisé non plus faute de temps. Cela me déçoit car je tenais à faire figurer dans ce rapport l'outil tel que je le prévois.

Concernant les outils utilisés, je peux dire qu'ils m'ont permis d'expériencer deux aspects très différents de prise en main. MDWorkbench a été vraiment très rapidement pris en main et, en un temps presque record, on maîtrise beaucoup des possibilitées offertes par cet outil. Tout cela avec une facilitée déconcertante. Topcased a été un outil ou il a fallu s'armer de patience et de persévérance, même pour réaliser les premiers pas, cependant, au même temps que l'on obtient de l'expérience et que l'on acquiert de nouvelles connaissances, l'intêret devient énorme de par les possibilités que cet outil nous laisse entrevoir. Les mécanismes employés par ces utilitaires logiciels (architecture MVC, plugins, extension points, etc...) permettent aussi d'acquérir des connaissances qui laissent envisager des solutions différentes.

12 Next Steps

Malgré que la génération de code corresponde à la spécification qui avait été faite, les outils n'ont pas encore subi de réèls tests et nécéssiteront surement un peu de debug. La prochaine étape sera de rendre cet outil encore bien plus performant en optimisant sa génération et de le rendre compatible avec d'autres produits existants.

Sion, le 21 novembre 2008 Hugo Nunes

13 Bibliographie et Références

Head First Design Patterns
O'Reilly Eric Freeman &Elisabeth Freeman

Design Patterns, Elements of Reusable Object-Oriented Software
Erich Gamma Richard Helm Ralph Johnson John Vlissides

http://lists.gforge.enseeiht.fr/pipermail/topcased-users/
The Topcased-users Archives

Model -. View - Controller architecture
http://wiki.eclipse.org/index.php/GEF_Description

Format XML optimisé pour la traduction de UML en code
Philippe Pralong

Diplôme 2007 : Environnement de cross-développement libre pour ARM EBS3
Joël Rochat

Machines à états finis Transitions
http://www.mathworks.com/access/helpdesk/help/toolbox/stateflow/ug/f18-74757.html#f18-8766
MathWorks - Stateflows

OCL
http://www2.lifl.fr/~nebut/ens/svl/coursOcl.html

ANNEXE A

```
/*
-------------------------------------------------------------------------------
-                                                                             -
- file      : xhal.h                                                         -
-                                                                             -
- Info      : This header file defines constants and macros to access the    -
-          PIC18Fxx8 hardware                                                 -
-                                                                             -
- Author : Hugo Nunes                                                         -
- Date      : 2008-11-10                                                      -
- Version   : 1.0                                                             -
-                                                                             -
-------------------------------------------------------------------------------
*/
#ifndef XHAL_H //=============================================================
#define XHAL_H
//TARGET INCLUSIONS-----------------------------------------------------------
#include <pic18.h>
#include "xf.h"


/*-----------------------------------------------------------------------------
- TYPES AND CONSTANTS                                                         -
-----------------------------------------------------------------------------*/

typedef unsigned char   UINT8;
typedef unsigned int    UINT16;
typedef unsigned long   UINT32;
typedef signed char     INT8;
typedef signed int      INT16;
typedef signed long     INT32;

typedef UINT8           BOOL;
#define FALSE           0x00
#define TRUE            0xFF
#define NULL         0x00

// PIC working with FOSC Hz frequency
#define FOSC                 4000000


/*-----------------------------------------------------------------------------
- FLOW CONTROLS                                                               -
-----------------------------------------------------------------------------*/

// Disables interruptions till LEAVE_CRITICAL ----------------------------------
//#define ENTER_CRITICAL()    do { \
//                          GIE = 0; \
//                      } while ( 0 )

// Re-enables interruptions leaving a critical section -------------------------
//#define LEAVE_CRITICAL()    do { \
//                          GIE = 1; \
//                      } while ( 0 )


/*-----------------------------------------------------------------------------
- INTERRUPTIONS CONTROLS                                                      -
-----------------------------------------------------------------------------*/

// Call the function ISR when getting VectorHigh interrupts --------------------
#define ISR()              InterruptVectorHigh(void)

// Enables global interrupts ---------------------------------------------------
#define INT_ENABLE()       do { GIE = 1; } while ( 0 )

// Disables global interrupts --------------------------------------------------
#define INT_DISABLE()         do { GIE = 0; } while ( 0 )

// Enables peripheral interrupts -----------------------------------------------
#define INT_PENABLE()         do { PEIE = 1; } while ( 0 )

// Disables peripheral interrupts ----------------------------------------------
#define INT_PDISABLE()    do ( PEIE = 0; } while ( 0 )


/*-----------------------------------------------------------------------------
- EXECUTION FRAMEWORK                                                         -
-----------------------------------------------------------------------------*/
```

```
// Timer of _XF ------------------------------------------------------------
#define XF_TIMER           TMR2

// Initializes the execution framework -------------------------------------
#define XF_INIT()          do { \
                           TMR2IE = 0; \
                           TMR2IF = 0; \
                           TMR2ON = 0; \
                           _XF_init(); \
                           T2CON = ( 0x49 ); \
                           PR2 = 0x9B; \
                           TMR2IE = 1; \
                           GIE = 1; \
                           TMR2ON = 1; \
                        } while ( 0 )

// Tick of software timer --------------------------------------------------
#define TICK()             do { \
                           if(TMR2IF!=0) { \
                              GIE = 0; \ \
                              _XF_Tick(); \
                              TMR2IF=0; \
                              GIE = 1; \
                           } \
                        } while ( 0 )


/*-----------------------------------------------------------------------------
- TIMER0                                                                      -
-----------------------------------------------------------------------------
----- T0_INIT( UINT8 prescale ) -----------------------------------------------
-                                                                            -
- prescale : The prescale factor timer 0 should use [0..7].                  -
-                                                                            -
- f(clk) = f(osc) / ( 8 * 2 ^ prescale ) = 1Mhz / ( 2 ^ prescale )           -
-                                                                            -
- f(int) = f(osc) / ( 8 * 2 ^ ( 8 + prescale ) )                             -
-      = 1Mhz ( 2 ^ ( 8 + prescale ) )                                        -
-----------------------------------------------------------------------------*/

// Initializes the TIMER0 --------------------------------------------------
#define T0_INIT( prescale )      do { \
                           T0CON = (prescale) & 0x07; \
                           T08BIT = 1; \
                           T0CS = 0; \
                           PSA = 0; \
                        } while ( 0 )

// Enables TIMER0 ----------------------------------------------------------
#define T0_ENABLE()           do { TMR0ON = 1; } while ( 0 )

// Disables TIMER0 ---------------------------------------------------------
#define T0_DISABLE()       do { TMR0ON = 0; } while ( 0 )

// Gets actual TIMER0 value as UINT16 --------------------------------------
#define T0_GET()              (( TMR0 ))

// Sets TIMER0 to given UINT16 value ---------------------------------------
#define T0_SET( value )       do { \
                           TMR0H = ( (value) & 0xFF00 ) >> 8; \
                           TMR0L = (value) & 0x00FF; \
                        } while ( 0 )

// Enables TIMER0 interruptions --------------------------------------------
#define T0_INTE()          do { TMR0IF = 0; TMR0IE = 1; } while ( 0 )

// Disables TIMER0 interruptions -------------------------------------------
#define T0_INTD()          do { TMR0IE = 0; } while ( 0 )

// Gets TIMER0 interrupt flag (!=0 <=> interrupt) --------------------------
#define T0_INT             (( TMR0IF ))

// Acknowledges TIMER0 interruption ----------------------------------------
#define T0_INTACK()           do { TMR0IF = 0; } while ( 0 )


/*-----------------------------------------------------------------------------
- TIMER1                                                                      -
```

```
-----------------------------------------------------------------------------
----- T1_INIT( UINT8 prescale ) -------------------------------------------
-                                                                           -
-* prescale : The prescale factor timer 1 should use [0..3].                -
-                                                                           -
- f(clk) = f(osc) / ( 4 * 2 ^ prescale )                                    -
-      = 1Mhz / ( 2 ^ prescale )                                            -
-                                                                           -
- f(int) = f(osc) / ( 4 * 2 ^ ( 16 + prescale ) )                           -
-      = 1Mhz ( 2 ^ ( 16 + prescale ) )                                     -
-----------------------------------------------------------------------------*/

// Initializes the TIMER1 ----------------------------------------------------
#define T1_INIT( prescale )   do { \
                           T1CON = ( (prescale) & 0x03 ) << 4; \
                           T1RD16 = 1; \
                           TMR1IP = 1; \
                        } while ( 0 )

// Enables TIMER1 ------------------------------------------------------------
#define T1_ENABLE()           do { TMR1ON = 1; } while ( 0 )

// Disables TIMER1 -----------------------------------------------------------
#define T1_DISABLE()       do { TMR1ON = 0; } while ( 0 )

// Gets actual TIMER1 value as UINT16 ----------------------------------------
#define T1_GET()              (( TMR1L | ( TMR1H << 8 ) ))

// Sets TIMER1 to given UINT16 value -----------------------------------------
#define T1_SET( value )       do { \
                           TMR1H = ( (value) & 0xFF00 ) >> 8; \
                           TMR1L = (value) & 0x00FF; \
                        } while ( 0 )

// Enables TIMER1 interruptions ----------------------------------------------
#define T1_INTE()          do { TMR1IF = 0; TMR1IE = 1; } while ( 0 )

// Disables TIMER1 interruptions ---------------------------------------------
#define T1_INTD()          do { TMR1IE = 0; } while ( 0 )

// Gets TIMER1 interrupt flag (!=0 <=> interrupt) ----------------------------
#define T1_INT             (( TMR1IF ))

// Acknowledges TIMER1 interruption ------------------------------------------
#define T1_INTACK()           do { TMR1IF = 0; } while ( 0 )


/*---------------------------------------------------------------------------
- TIMER2                                                                    -
-----------------------------------------------------------------------------
----- T2_INIT( UINT8 prescale , UINT8 postscale , UINT8 max ) -------------
-                                                                           -
- prescale : The prescale factor timer 2 should use [0..2].                 -
- postscale : The postscale factor timer 2 should use [0..15].              -
-                                                                           -
- f(clk) = f(osc) / ( 4 * 2 ^ ( 2 * prescale ) )                            -
-      = 1Mhz / ( 2 ^ ( 2 * prescale ) )                                    -
-                                                                           -
- f(int)                                                                    -
-  = f(osc) / ( 4 * 2 ^ ( 2 * prescale ) * ( postscale + 1 ) * ( max + 1 ) )  -
-  = 1MHz / ( 2 ^ ( 2 * prescale ) * ( postscale + 1 ) * ( max + 1 ) )      -
-----------------------------------------------------------------------------*/

// Initializes the TIMER2 ----------------------------------------------------
#define T2_INIT( prescale , postscale , max ) \
                        do { \
                           T2CON = ( (prescale) & 0x03 ) | ( ( (postscale) & 0x0F ) << 3 ); \
                           PR2 = (max); \
                           TMR2IP = 1; \
                        } while ( 0 )

// Enables TIMER2 ------------------------------------------------------------
#define T2_ENABLE()           do { TMR2ON = 1; } while ( 0 )

// Disables TIMER2 -----------------------------------------------------------
#define T2_DISABLE()       do { TMR2ON = 0; } while ( 0 )

// Gets actual TIMER2 value as UINT8 -----------------------------------------
#define T2_GET()              (( TMR2 ))
```

```
// Sets TIMER2 to given UINT8 value ---------------------------------------------
#define T2_SET( value )       do { TMR2 = (value); } while ( 0 )

// Enables TIMER2 interruptions -------------------------------------------------
#define T2_INTE()          do { TMR2IF = 0; TMR2IE = 1; } while ( 0 )

// Disables TIMER2 interruptions ------------------------------------------------
#define T2_INTD()          do { TMR2IE = 0; } while ( 0 )

// Gets TIMER2 interrupt flag (!=0 <=> interrupt) -------------------------------
#define T2_INT             (( TMR2IF ))

// Acknowledges TIMER2 interruption ---------------------------------------------
#define T2_INTACK()           do { TMR2IF = 0; } while ( 0 )


/*-----------------------------------------------------------------------------
- TIMER3                                                                      -
-----------------------------------------------------------------------------
----- T3_INIT( UINT8 prescale ) ----------------------------------------------
-                                                                            -
- prescale : The prescale factor timer 3 should use [0..3].                  -
-            f(clk) = f(osc) / ( 4 * 2 ^ prescale )                       -
-              = 2Mhz / ( 2 ^ prescale )                                  -
-                                                          -
-            f(int) = f(osc) / ( 4 * 2 ^ ( 16 + prescale ) )              -
-              = 2Mhz ( 2 ^ ( 16 + prescale ) )                           -
-----------------------------------------------------------------------------*/

// Initializes the TIMER3 -------------------------------------------------------
#define T3_INIT( prescale )   do { \
                          T3CON = ( (prescale) & 0x03 ) << 4; \
                          T3RD16 = 1; \
                          TMR3IP = 1; \
                       }while ( 0 )

// Enables TIMER3 ---------------------------------------------------------------
#define T3_ENABLE()           do { TMR3ON = 1; } while ( 0 )

// Disables TIMER3 --------------------------------------------------------------
#define T3_DISABLE()       do { TMR3ON = 0; } while ( 0 )

// Gets actual TIMER3 value as UINT16 -------------------------------------------
#define T3_GET()              (( TMR3L | ( TMR3H << 8 ) ))

// Sets TIMER3 to given UINT16 value --------------------------------------------
#define T3_SET( value )       do { TMR3H = ( (value) & 0xFF00 ) >> 8; \
                          TMR3L = (value) & 0x00FF; \
                       } while ( 0 )

// Enables TIMER3 interruptions -------------------------------------------------
#define T3_INTE()          do { TMR3IF = 0; TMR3IE = 1; } while ( 0 )

// Disables TIMER3 interruptions ------------------------------------------------
#define T3_INTD()          do { TMR3IE = 0; } while ( 0 )

// Gets TIMER3 interrupt flag (!=0 <=> interrupt) -------------------------------
#define T3_INT             (( TMR3IF ))

// Acknowledges TIMER3 interruption ---------------------------------------------
#define T3_INTACK()           do { TMR3IF = 0; } while ( 0 )


/*-----------------------------------------------------------------------------
- INTERRUPT INT0                                                              -
-----------------------------------------------------------------------------*/

// Initialise the INT0 ----------------------------------------------------------
#define INT0_INIT()        do { \
                          TRISB |= 0x01; \
                       } while ( 0 )

// Enable INT0 interruptions ----------------------------------------------------
#define INT0_INTE()           do { INT0IF = 0; INT0IE = 1; } while ( 0 )

// Disables INT0 interruptions --------------------------------------------------
#define INT0_INTD()           do { INT0IE = 0; } while ( 0 )
```

```
// Rising edge triggers INT0 ---------------------------------------------------
#define INT0_TRIGRE()         do { INTEDG0 = 1; } while ( 0 )

// Falling edge triggers INT0 ---------------------------------------------------
#define INT0_TRIGFE()         do { INTEDG0 = 0; } while ( 0 )

// Toggle edge trigger setting --------------------------------------------------
#define INT0_TRIGTOG()        do { INTEDG0 = !INTEDG0; } while ( 0 )

// INT0 flag (RB0 caused interrupt) ---------------------------------------------
#define INT0_INT           (( INT0IF ))

// INT0 ack ---------------------------------------------------------------------
#define INT0_INTACK()         do { INT0IF = 0; } while ( 0 )


/*-----------------------------------------------------------------------------
- WATCHDOG                                                                    -
-----------------------------------------------------------------------------
----- WD_INIT( UINT8 postscale ) -------------------------------------------
-                                                                            -
- postscale : The scale factor to use with the watchdog timer [0..15].       -
-              t(wd) = ~18ms * 2 ^ postscale.                                -
-----------------------------------------------------------------------------*/

// Initialise the watchdog ------------------------------------------------------
#define WD_INIT( postscale )  do { \
                          CONFIG2H = ( (postscale) & 0x07 ) << 1; \
                        } while ( 0 )

// Enables the watchdog ---------------------------------------------------------
#define WD_ENABLE()           do { SWDTEN = 1; } while ( 0 )

// Disables the watchdog --------------------------------------------------------
#define WD_DISABLE()          do { SWDTEN = 0; } while ( 0 )

// Clears the watchdog (normal use) ---------------------------------------------
#define WD_CLEAR()            do { asm( "CLRWDT" ); } while ( 0 )


/*-----------------------------------------------------------------------------
- UART                                                                        -
-----------------------------------------------------------------------------
----- UART_INIT( UINT16 baudrate ) -----------------------------------------
- Initializes the UART hardware.                                             -
-                                                                            -
- baudrate : The baudrate the UART should use in bauds [300..96000].         -
-----------------------------------------------------------------------------*/

// Initializes the UART ---------------------------------------------------------
#define UART_INIT( baudrate ) \
                        do { \
                          TRISC &= ~0x40; \
                          TRISC |= 0x80; \
                          SPBRG = ( ( ( FOSC / (baudrate) ) / 16 ) - 1 ); \
                          SPEN = 1; \
                          BRGH = 1; \
                          RCIP = 1; \
                          TXIP = 1; \
                        } while ( 0 )

// Enables the UART transmitter -------------------------------------------------
#define UART_TX_ENABLE()      do { TXEN = 1; } while ( 0 )

// Disables the UART transmitter ------------------------------------------------
#define UART_TX_DISABLE()     do { TXEN = 0; } while ( 0 )

// UART ready to transmit next character flag -----------------------------------
#define UART_TX_READY         (( TRMT ))

// Transmits the character given as value ---------------------------------------
#define UART_TX( value )      do { TXREG = (value); } while ( 0 )

// Interrupt of end of transmission is enabled ----------------------------------
#define UART_TX_INTE()        do { TXIF = 0; TXIE = 1; } while ( 0 )

// Interrupt of end of transmission is disabled ---------------------------------
#define UART_TX_INTD()        do { TXIE = 0; } while ( 0 )
```

```
// End of transmission interrupt flag ----------------------------------------
#define UART_TX_INT           (( TXIF ))

// End of transmission interrupt acknowledge ---------------------------------
#define UART_TX_INTACK()      do { TXIF = 0; } while ( 0 )

// Enables the UART receiver -------------------------------------------------
#define UART_RX_ENABLE()      do { CREN = 1; } while ( 0 )

// Disables the UART receiver ------------------------------------------------
#define UART_RX_DISABLE()     do { CREN = 0; } while ( 0 )

// UART ready to receive next character --------------------------------------
#define UART_RX_READY         (( RCIF ))

// Gives the received character as UINT8 -------------------------------------
#define UART_RX()           (( RCREG ))

// Interrupt of end of reception is enabled ----------------------------------
#define UART_RX_INTE()        do { RCIF = 0; RCIE = 1; } while ( 0 )

// Interrupt of end of reception is disabled ---------------------------------
#define UART_RX_INTD()        do { RCIE = 0; } while ( 0 )

// End of reception interrupt flag -------------------------------------------
#define UART_RX_INT           (( RCIF ))

// End of reception interrupt acknowledge ------------------------------------
#define UART_RX_INTACK()      do { RCIF = 0; } while ( 0 )


/*----------------------------------------------------------------------------
- LEDS                                                                       -
----------------------------------------------------------------------------*/

// LEDs mask (0x00=no LED, 0xFF=all LEDs) ------------------------------------
#define LEDS_MASK          0xFE

// Port of LEDs (3 settings must be related) ---------------------------------
#define LEDS_PORT          PORTB
#define LEDS_TRIS          TRISB
#define LEDS_LAT           LATB

// LEDs initialize -----------------------------------------------------------
#define LEDS_INIT()        do { \
                           LEDS_TRIS &= (0xFF ^ LEDS_MASK); \
                           LEDS_PORT = LEDS_LAT & (0xFF ^ LEDS_MASK); \
                        } while ( 0 )

// LEDs state (UINT8) --------------------------------------------------------
#define LEDS_STATE()       ((LEDS_LAT))

// Set LEDs with the 1 value in the UINT8 ledsmask ---------------------------
#define LEDS_SET( ledsmask )     do { \
                           LEDS_PORT = \
                           LEDS_LAT | (LEDS_MASK & (ledsmask)); \
                        } while ( 0 )

// Clear LEDs with the 1 value in the UINT8 ledsmask -------------------------
#define LEDS_CLEAR( ledsmask )   do { \
                           LEDS_PORT = \
                           LEDS_LAT & (~LEDS_MASK & ~(ledsmask)); \
                        } while ( 0 )



#endif // XHALL_H ==============================================================
```

ANNEXE B

```
#ifndef XF_H //=================================================================
#define XF_H
// INCLUSIONS -----------------------------------------------------------------
#include "../app_constants.h"

//PREPROCESSOR----------------------------------------------------------------

//this define specifies the type or structure of data to be used with XF
#define EVENT_TYPE            struct event
//#define DEBUG


#ifdef DEBUG
#define DEBUG_PRINT(x)        _XF_printout(x)
#else
#define DEBUG_PRINT(x)        {} // {} -> don't print
#endif

//STRUCTURES------------------------------------------------------------------
typedef enum _XF_evType{
   emptyEv        = 0,  //last node, means the list is empty
   defaultEv      = 1,  //the event is placed directly in the Events List
   timerEv        = 2,  //the event is a timer, it goes to Timers List
   otherEv        = 3      //not used... yet!...
};

//this is the events node structure
struct _XF_evNode{
   struct _XF_evNode*      next;
   struct _XF_evNode*      previous;
   int                  evType;     //as in _XF_evType enumeration
   int                  index;      //identity of the event
   int                  counter; //decreasing counter for timer event
   char              overwrite;  //event goes in list without overwritting event with same index
   EVENT_TYPE           event;      //the event data
};


//VARIABLES-------------------------------------------------------------------
struct _XF_evNode* _XF_event;             //the event node iterator
struct _XF_evNode* _XF_tempEv;               //used to ease internal operations

struct _XF_evNode* _XF_firstEvent;           //the top of the event list node
struct _XF_evNode _XF_evEmptyNode;           //the empty event node of the list (or last node)

struct _XF_evNode* _XF_firstTimer;           //the top of the timer list node
struct _XF_evNode _XF_tmEmptyNode;           //the empty timer node of the list (or last node)

//FUNCTIONS-------------------------------------------------------------------
//initialises functions
void _XF_init(void);

////////////////////////////////////////////////////////////////////////////////
//Parameters :
//          index :
//                The event to stock (with all needed information), the
//                structure is defined in a constants file.
//
//          evType   :
//                Specifies the list to compute...
//                1 - eventsList
//                2 - timersList
//
//Return :
//          int      :
//                0  - the _XF_event is placed on the event
//                     and _XF_tempEv is placed on the preceding event in list
//
//                1  - the list is empty
//
//                2  - the list don't contain the event specified by the
//                     given index
//
//                3  - wrong evType parameter input...
//
////////////////////////////////////////////////////////////////////////////////
//place iterator at next event with given index
int _XF_placeAtEvIndex(int index,int evType);
```

```
//clears timer events in the timer list
void _XF_clearTms(void);
//clears events in the event list
void _XF_clearEvs(void);

////////////////////////////////////////////////////////////////////////////////
//Parameters :
//          newEv :
//                The event to stock (with all needed information), the
//                structure is defined in a constants file.
//          indexEv  :
//                Index of the event. Events with same index can be overwrited
//                in the list. indexEv=0 will automatically disable the
//                overwrite function.
//          overwrite   :
//                If this setting is activated then the event will overwrite
//                the already present event with same indexEv.
//                If deactivated, the event will be pushed to event list
//                as a separated event even if the indexEv is already
//                used by another event present in the list.
//
//Return :
//          int      :
//                0  - operation ok
//
//                1  - not enough memory
//
//                2  - Call of _XF_placeAtEvIndex reported a problem.
//                     (call returned 3 or greater)
//
////////////////////////////////////////////////////////////////////////////////
//add an event to list
int _XF_evPush(EVENT_TYPE newEv, int indexEv, char overwrite);

////////////////////////////////////////////////////////////////////////////////
//Parameters :
//          *outEv      :
//                The event structure used for returning the event.
//          indexEv  :
//                This is the index of the event to pop out of the list.
//                If indexEv=0 is used then the lastmost event of the list
//                will be pop.
//
//Return :
//          int      :
//                0  - the specified event was given
//
//                1  - the list is empty
//
//                2  - the specified event was not found
//
//                3  - an error occured...
//
////////////////////////////////////////////////////////////////////////////////
//gets an event from the list
int _XF_evPop(EVENT_TYPE* outEv,int indexEv);

//adds a timer event to the timer event list, when timedout it goes to event list
int _XF_tmPush(EVENT_TYPE newEv, int indexEv, int duration, char overwrite);
//decreases unit time of timer events on timer event list and when timedout it goes to event list
//moves timer events
void _XF_Tick(void);

////////////////////////////////////////////////////////////////////////////////
// DEBUG                                                        //
////////////////////////////////////////////////////////////////////////////////
void _XF_listEvState(void);

void _XF_listEvState(void);

void _XF_printout(char* mes);


#endif // XF_H ==================================================================
```

```
#include "xf.h"

void _XF_init(void)  {
   //Empty Event
   EVENT_TYPE emptyEvent;
   //EVENT STACK
   //current stack node initialisation
   _XF_firstEvent = &_XF_evEmptyNode;
   //empty stack node initialisation
   _XF_event         = &_XF_evEmptyNode;
   _XF_event->next   = _XF_event;
   _XF_event->previous  = _XF_event;
   _XF_event->evType    = 0;
   _XF_event->index  = 0;
   _XF_event->counter   = 0;
   _XF_event->event  = emptyEvent;
   //TIMER STACK
   //current stack node initialisation
   _XF_firstTimer = &_XF_tmEmptyNode;
   //empty stack node initialisation
   _XF_event         = &_XF_tmEmptyNode;
   _XF_event->next   = _XF_event;
   _XF_event->previous  = _XF_event;
   _XF_event->evType    = 0;
   _XF_event->index  = 0;
   _XF_event->counter   = 0;
   _XF_event->event  = emptyEvent;
   //STRUCT POINTERS
   _XF_event         = _XF_firstEvent;
   _XF_tempEv        = _XF_firstEvent;
}

int _XF_placeAtEvIndex(int index,int evType){
   //choses wich list to use
   struct _XF_evNode* firstEv;
   switch(evType) {
      case defaultEv:
         firstEv=_XF_firstEvent;
         break;
      case timerEv:
         firstEv=_XF_firstTimer;
         break;
      default:
         DEBUG_PRINT("unknown evType...(placeAtEvIndex)");
         return 3;
         break;
   }
   //check - list empty?
   if(firstEv->evType==emptyEv)  {
      DEBUG_PRINT("the list is empty...(placeAtEvIndex)");
      _XF_event = firstEv;
      _XF_tempEv = firstEv;
      return 1;
   }
   _XF_event = firstEv;
   while(_XF_event->evType != emptyEv) {                  //sort all the events
      if(_XF_event->index == index) {
         break;
      }
      _XF_event = _XF_event->next;
   }
   //check - list end ?
   if(_XF_event->evType==emptyEv)   {
      DEBUG_PRINT("the list don't contain what you want...(placeAtEvIndex)");
      _XF_event = firstEv;
      _XF_tempEv = firstEv;
      return 2;
   }
   //update _XF_tempEv
   if(_XF_event == firstEv)   {
      _XF_tempEv = _XF_event;
   }
   else  {
      _XF_tempEv = _XF_event->previous;
   }
   //exits with _XF_event at specified position
   //_XF_tempEv in preceding position, first if _XF_event is same as _XF_firstEv
   DEBUG_PRINT("search done!(placeAtEvIndex)");
```

```
    return 0;
}

void _XF_clearTms(void) {
    _XF_event = _XF_firstTimer;
    while(_XF_event->evType != emptyEv) {
        _XF_tempEv = _XF_event;
        _XF_event = _XF_event->next;
        free(_XF_tempEv);
    }
    _XF_tempEv = _XF_event;
    _XF_firstTimer = _XF_event;
    _XF_firstTimer->previous = _XF_firstTimer;
    DEBUG_PRINT("clearTms done!");
}

void _XF_clearEvs(void) {
    _XF_event = _XF_firstEvent;
    while(_XF_event->evType != emptyEv) {
        _XF_tempEv = _XF_event;
        _XF_event = _XF_event->next;
        free(_XF_tempEv);
    }
    _XF_tempEv = _XF_event;
    _XF_firstEvent = _XF_event;
    _XF_firstEvent->previous = _XF_firstEvent;
    DEBUG_PRINT("clearEvs done!");
}

int _XF_evPush(EVENT_TYPE newEv, int indexEv, char overwrite) {
    if((overwrite==0)||(indexEv==0)) {
    //do this when event overwrite disabled
        //gets memory
        _XF_event = malloc(sizeof(struct _XF_evNode));
        if(_XF_event == 0)   {  //no more memory
            DEBUG_PRINT("Not enough memory available...");
            return 1;
        }
        //configure event node
        _XF_event->next         = &_XF_evEmptyNode;
        _XF_event->evType       = 1;
        _XF_event->index     = indexEv;
        _XF_event->counter      = 0;
        _XF_event->overwrite    = overwrite;
        _XF_event->event     = newEv;
        if(_XF_firstEvent->evType==emptyEv) {  //update firstEvent Ptr
            _XF_event->previous  =_XF_event;
            _XF_evEmptyNode.previous = _XF_event;
            _XF_firstEvent = _XF_event;
            DEBUG_PRINT("update firstEvent Ptr");
        }
        else  {                        //update list elements
            _XF_event->previous     = _XF_evEmptyNode.previous;
            (_XF_evEmptyNode.previous)->next = _XF_event;
            _XF_evEmptyNode.previous = _XF_event;
            DEBUG_PRINT("update list elements");
        }
        //exit message and state
        DEBUG_PRINT("PUSH OK (evPush)");
        return 0;
    }
    else  {
    //do this when event overwrite enable
        //place _XF_event and _XF_tempEv
        int placeRet;
        placeRet = _XF_placeAtEvIndex(indexEv,1);
        if(placeRet>=3)   {
        //call of place _XF_event reported problems
            DEBUG_PRINT("call of place _XF_event reported problems...");
            return 2;
        }
        else  {
                    //checks if pointer at first node of list(used later)
                    int first = 0;
                    if(_XF_event==_XF_firstEvent) {  first = 1;  }
            //gets memory (_XF_event pointer used)
            _XF_event = malloc(sizeof(struct _XF_evNode));
            if(_XF_event == 0)   {
            //no more memory
```

```
            DEBUG_PRINT("Not enough memory available...");
            _XF_event   = _XF_firstEvent;
            _XF_tempEv  = _XF_firstEvent;
            return 1;
         }
         else  {
            //configure event
            _XF_event->evType       = 1;
            _XF_event->index     = indexEv;
            _XF_event->counter      = 0;
            _XF_event->overwrite    = overwrite;
            _XF_event->event     = newEv;
            if(placeRet==1)   {
            //the list is empty
               _XF_event->next = &_XF_evEmptyNode;
               _XF_event->previous = _XF_event;
               _XF_evEmptyNode.previous = _XF_event;
               _XF_firstEvent = _XF_event;
            }
            else {
               if(placeRet==2)   {
               //this event index is not present in the event list
                  _XF_event->next = &_XF_evEmptyNode;
                  _XF_event->previous = _XF_evEmptyNode.previous;
                  _XF_evEmptyNode.previous = _XF_event;
                  (_XF_event->previous)->next = _XF_event;
               }
               else  {
                  if(first!=0)   {
                  //event to replace is the first node in the event list
                     _XF_event->next = _XF_firstEvent->next;
                     _XF_event->previous = _XF_event;
                     (_XF_event->next)->previous = _XF_event;
                     free(_XF_firstEvent);
                     _XF_firstEvent = _XF_event;
                  }
                  else  {
                  //event node to replace is not the firstEvent
                     _XF_event->next = (_XF_tempEv->next)->next;
                     _XF_event->previous = _XF_tempEv;
                     (_XF_event->next)->previous = _XF_event;
                     _XF_tempEv = _XF_tempEv->next;
                     free(_XF_tempEv);
                     (_XF_event->previous)->next = _XF_event;
                  }
               }
            }
            DEBUG_PRINT("PUSH OK (evPush)");
            return 0;
         }
      }
   }
}

int _XF_evPop(EVENT_TYPE* outEv,int indexEv){
   int ret;
   //_XF_event positionment
   if(indexEv==0) {  //index search not required
      //test if list is empty
      if(_XF_firstEvent->evType==emptyEv) {
         DEBUG_PRINT("the list seems empty...");
         ret = 1;
      }
      else  {
         //get the first event node
         DEBUG_PRINT("getting the first event node...");
         _XF_tempEv = _XF_firstEvent;
         _XF_event = _XF_firstEvent;
         ret = 0;
      }
   }
   else  {        //index search
      //get the event node
      DEBUG_PRINT("getting the specified event node...");
      ret = _XF_placeAtEvIndex(indexEv,1);
   }
   //result and pop
   switch(ret) {
      case 0:
```

```
         //updates list
         if(_XF_event==_XF_firstEvent) {  //updates first list event
            _XF_firstEvent=_XF_event->next;
            _XF_firstEvent->previous = _XF_firstEvent;
            DEBUG_PRINT("updated first part of list...");
         }
         else  {
            (_XF_event->previous)->next = _XF_event->next;
            (_XF_event->next)->previous = _XF_event->previous;
            DEBUG_PRINT("updated the list nodes...");
         }
         //sends event
         (*outEv) = (_XF_event->event);
         DEBUG_PRINT("POP  OK");
         //free memory allocated
         free(_XF_event);
         DEBUG_PRINT("memory freed...");
         return 0;
         break;
      case 1:
         DEBUG_PRINT("the list is empty, cannot POP !");
         return 1;
         break;
      case 2:
         DEBUG_PRINT("the event searched was not found, no POP !");
         return 2;
         break;
      default :
         return 3;
         break;
   }
   return 3;
}

int _XF_tmPush(EVENT_TYPE newEv, int indexEv, int duration, char overwrite)  {
   if(duration!=0)   {
      if(overwrite == 0)   {
      //do this when event overwrite disabled
         //gets memory
         _XF_event = malloc(sizeof(struct _XF_evNode));
         if(_XF_event == 0)   {  //no more memory
            DEBUG_PRINT("Not enough memory available...");
            return 1;
         }
         //configure event node
         _XF_event->next         = &_XF_tmEmptyNode;
         _XF_event->evType       = 2;
         _XF_event->index     = indexEv;
         _XF_event->counter      = duration;
         _XF_event->overwrite    = 0;
         _XF_event->event     = newEv;
         if(_XF_firstTimer->evType==emptyEv) {
         //this timer is the first on the list
            _XF_event->previous  =_XF_event;
            _XF_tmEmptyNode.previous = _XF_event;
            _XF_firstTimer = _XF_event;
            DEBUG_PRINT("update firstTimer Ptr");
         }
         else  {                        //update list elements
            _XF_event->previous     = _XF_tmEmptyNode.previous;
            (_XF_tmEmptyNode.previous)->next = _XF_event;
            _XF_tmEmptyNode.previous = _XF_event;
            DEBUG_PRINT("update list elements");
         }
         //exit message and state
         DEBUG_PRINT("PUSH OK (tmPush)");
         return 0;
      }
      else  {
      //do this when event overwrite enable
         //place _XF_event and _XF_tempEv
         int placeRet;
         placeRet = _XF_placeAtEvIndex(indexEv,2);
         if(placeRet>=3)   {
         //call of place _XF_event reported problems
            DEBUG_PRINT("call of place _XF_event reported problems...");
            return 2;
         }
         else  {
```

```
                    //checks if pointer at first node of list(used later)
                    int first = 0;
                    if(_XF_event==_XF_firstTimer) {  first = 1;  }
            //gets memory (_XF_event pointer used)
            _XF_event = malloc(sizeof(struct _XF_evNode));
            if(_XF_event == 0)   {
            //no more memory
               DEBUG_PRINT("Not enough memory available...");
               _XF_event   = _XF_firstTimer;
               _XF_tempEv  = _XF_firstTimer;
               return 1;
            }
            else  {
               //configure event
               _XF_event->evType       = 2;
               _XF_event->index     = indexEv;
               _XF_event->counter      = duration;
               _XF_event->overwrite    = overwrite;
               _XF_event->event     = newEv;
               if(placeRet==1)   {
               //the list is empty
                  _XF_event->next = &_XF_tmEmptyNode;
                  _XF_event->previous = _XF_event;
                  _XF_tmEmptyNode.previous = _XF_event;
                  _XF_firstTimer = _XF_event;
               }
               else {
                  if(placeRet==2)   {
                  //this event index is not present in the event list
                     _XF_event->next = &_XF_tmEmptyNode;
                     _XF_event->previous = _XF_tmEmptyNode.previous;
                     _XF_tmEmptyNode.previous = _XF_event;
                     (_XF_event->previous)->next = _XF_event;
                  }
                  else  {
                     if(first!=0)   {
                     //event to replace is the first node in the event list
                        _XF_event->next = _XF_firstTimer->next;
                        _XF_event->previous = _XF_event;
                        (_XF_event->next)->previous = _XF_event;
                        free(_XF_firstTimer);
                        _XF_firstTimer = _XF_event;
                     }
                     else  {
                     //event node to replace is not the firstEvent
                        _XF_event->next = (_XF_tempEv->next)->next;
                        _XF_event->previous = _XF_tempEv;
                        (_XF_event->next)->previous = _XF_event;
                        _XF_tempEv = _XF_tempEv->next;
                        free(_XF_tempEv);
                        (_XF_event->previous)->next = _XF_event;
                     }
                  }
               }
               DEBUG_PRINT("PUSH OK (tmPush)");
               return 0;
            }
         }
      }
   }
   else  {
   //duration makes timer act as event
      DEBUG_PRINT("tmPush detects duration==0, so using evPush instead");
      _XF_evPush(newEv,indexEv,overwrite);
   }
}

void _XF_Tick(void){
   _XF_event = _XF_firstTimer;
   struct _XF_evNode* tempTimerE;
   struct _XF_evNode* tempTimerTE;
   while(_XF_event->evType != emptyEv) {
      (_XF_event->counter)=(_XF_event->counter)-1;
      _XF_tempEv = _XF_event->next;
      //check if timer as reached timeout
      if((_XF_event->counter)<=0)   {
         //remove from tmList
         if(_XF_event == _XF_firstTimer)  {
         //is firstTimer in list
```

```
            (_XF_event->next)->previous = _XF_event->next;
            _XF_firstTimer = _XF_event->next;
         }
         else  {
            (_XF_event->previous)->next = _XF_event->next;
            (_XF_event->next)->previous = _XF_event->previous;
         }
         //move to evList
         if((_XF_firstEvent->evType)==emptyEv)  {
         //no events in evList yet
            _XF_event->previous = _XF_event;
            _XF_event->next = &_XF_evEmptyNode;
            _XF_evEmptyNode.previous = _XF_event;
            _XF_firstEvent = _XF_event;
         }
         else  {
         //events already present in the event list
            if((_XF_event->overwrite)==0) {
            //overwrite mode disabled
               _XF_event->next = &_XF_evEmptyNode;
               _XF_event->previous = _XF_evEmptyNode.previous;
               _XF_evEmptyNode.previous = _XF_event;
               (_XF_event->previous)->next = _XF_event;
            }
            else  {
            //overwrite mode
               if(_XF_firstEvent->index == _XF_event->index)   {
               //event to replace is the first on event list
                  _XF_event->previous = _XF_event;
                  _XF_event->next = _XF_firstEvent->next;
                  if(_XF_evEmptyNode.previous == _XF_firstEvent)  {
                     _XF_evEmptyNode.previous = _XF_event;
                  }
                  else  {
                     (_XF_firstEvent->next)->previous = _XF_event;
                  }
                  free(_XF_firstEvent);
                  _XF_firstEvent = _XF_event;
               }
               else  {
               //event to replace is not the first in event list
                  tempTimerE = _XF_event;
                  tempTimerTE = _XF_tempEv;
                  //place _XF_event and _XF_tempEv
                  int placeRet;
                  placeRet = _XF_placeAtEvIndex(_XF_event->index,1);
                  if(placeRet>=3)   {
                  //call of place _XF_event reported problems
                     DEBUG_PRINT("call of place _XF_event reported problems...(Tick)");
                  }
                  else  {
                     if(placeRet==2)   {
                     //this event index is not present in the event list
                        tempTimerE->next = &_XF_evEmptyNode;
                        tempTimerE->previous = _XF_evEmptyNode.previous;
                        _XF_evEmptyNode.previous = tempTimerE;
                        (tempTimerE->previous)->next = tempTimerE;
                     }
                     else  {
                     //event node to replace is not the firstEvent
                        tempTimerE->next = _XF_event->next;
                        tempTimerE->previous = _XF_event->previous;
                        (tempTimerE->next)->previous = tempTimerE;
                        (tempTimerE->previous)->next = tempTimerE;
                        free(_XF_event);
                     }
                  }
                  _XF_event = tempTimerE;
                  _XF_tempEv = tempTimerTE;
               }
            }
         }
      }
      _XF_event = _XF_tempEv;
   }
   DEBUG_PRINT("Tick !!");
}

////////////////////////////////////////////////////////////////////////////////
```

```
// DEBUG SECTION                                                //
////////////////////////////////////////////////////////////////////////////////
void _XF_listEvState(void){
   printf("\n");
   printf("STATE OF EVENTS LIST\n");
   int i = 0;
   _XF_tempEv=_XF_firstEvent;
   while((_XF_tempEv->evType)!=0)   {

      printf("Node %i - EvIndex %i  Counter %i  Over %u   ",i,(int)(_XF_tempEv->index),(int)
(_XF_tempEv->counter),(char)(_XF_tempEv->overwrite));
      printf("<=> %s - %d \n",(_XF_tempEv->event).name,(_XF_tempEv->event).index);

      i++;
      _XF_tempEv=_XF_tempEv->next;
   }
   printf("\n");
}

void _XF_listTmState(void){
   printf("\n");
   printf("STATE OF TIMERS LIST\n");
   int i = 0;
   _XF_tempEv=_XF_firstTimer;
   while((_XF_tempEv->evType)!=0)   {

      printf("Node %i - EvIndex %i  Counter %i  Over %u   ",i,(int)(_XF_tempEv->index),(int)
(_XF_tempEv->counter),(char)(_XF_tempEv->overwrite));
      printf("<=> %s -  %d \n",(_XF_tempEv->event).name,(_XF_tempEv->event).index);

      i++;
      _XF_tempEv=_XF_tempEv->next;
   }
   printf("\n");
}

void _XF_printout(char* mes){
   printf("XF message: ");
   printf(mes);
   printf("\n");
}
```

ANNEXE C

[#package c.generator]

```
[#template public appFile(application : fsmapp.Application)]
[#file]generated/${application.getCFileName()}[/#file]
/*
*******************************************************************************
* This is a generated file (from a fsmapplication model)                      *
*******************************************************************************
*                                                                             *
* file      : ${application.getCFileName()}                                   *
* template  : appFile.tgt                                                     *
*                                                                             *
* Info      : This file implements the main functionality of the              *
*          application.                                                       *
*                                                                             *
* Author : Hugo Nunes                                                         *
* Date      : 2008-11-06                                                      *
* Version   : 1.0                                                             *
*                                                                             *
*******************************************************************************
*/
[#-- Add the TRANSIENT LINKS --]
${application.addTransientLinks()}
// INCLUSIONS ------------------------------------------------------------------
#include "${application.getHeaderFileName()}"

// ENUMERATIONS ----------------------------------------------------------------

//-----------------------------------------------------------------------------
// STRUCTURES ------------------------------------------------------------------
//-----------------------------------------------------------------------------
//Signals
${application.typewriteAllSignalStructs()}
${application.typewriteSignalsUnion()}
//Events
${application.typewriteEventStruct()}

// TYPE DEFINITIONS ------------------------------------------------------------

// DECLARATIONS ----------------------------------------------------------------

/*-----------------------------------------------------------------------------
-     MAIN                                                                    -
-----------------------------------------------------------------------------*/
void main( void ) {

   //initialisation of the XF
   XF_INIT();
   ${application.eventPtrString()} = NULL;

[#--Call the first Class Classifier Behaviour--]
   //call of the statemachine (or Behavior) of Class
[#foreach class : fsmapp.Class in application.getTheClass()]
   [#if class.getClassifierBehavior()!=null]
${class.getClassifierBehavior()#formName.first()};
   [/#if]
[/#foreach]
}
// MAIN END ========================================================================

/*-----------------------------------------------------------------------------
-     ISR                                                                     -
-----------------------------------------------------------------------------*/
// INTERRUPTION Handler --------------------------------------------------------
void ISR(void) {
   //XHALL tick call
   TICK();

   //User Implementation from model
${application.theISR.body}[#trim]

[#protectedStartTag]//CUSTOM USER IMPLEMENTATION[/#protectedStartTag]
//TODO: add here your modifications to the ISR

[#protectedEndTag]//END CUSTOM USER IMPLEMENTATION[/#protectedEndTag]
}
// ISR END ========================================================================
```

```
/*-----------------------------------------------------------------------------
-     EVENT PUSH OPERATIONS                                                   -
-----------------------------------------------------------------------------*/
${application.typewriteAllEventsOperationsImplementation()}

[/#template]
```

```
[#package c.generator]

[#template public appHeader(application : fsmapp.Application)]
[#file]generated/${application.getHeaderFileName()}[/#file]
/*
*****************************************************************************
* This is a generated file (from a fsmapplication model)                    *
*****************************************************************************
*                                                                           *
* file      : ${application.getHeaderFileName()}                            *
* template  : appHeader.tgt                                                 *
*                                                                           *
* Info      : This file implements the interface of the main file of the    *
*          application.                                                     *
*                                                                           *
* Author : Hugo Nunes                                                       *
* Date      : 2008-11-06                                                    *
* Version   : 1.0                                                           *
*                                                                           *
*****************************************************************************
*/
#ifndef ${application.formApplicationName()}_H
#define ${application.formApplicationName()}_H
//---------------------------------------------------------------------------
// INCLUSIONS ---------------------------------------------------------------
//---------------------------------------------------------------------------
#include "xhal.h"
#include "xf.h"
${application.typewriteAllInclusions()}

//---------------------------------------------------------------------------
// PREPROCESSOR -------------------------------------------------------------
//---------------------------------------------------------------------------
#define GEN( x )     do {  \
                     ENTER_CRITICAL();  \
                     ${application.signalPushPrefix()}##x;   \
                     LEAVE_CRITICAL();  \
                  } while (0)
#define EVPARAMS     ${application.eventPtrString()}

//---------------------------------------------------------------------------
// ENUMERATIONS -------------------------------------------------------------
//---------------------------------------------------------------------------
${application.typewriteAllSignalsEnum()}

//---------------------------------------------------------------------------
// TYPE DEFINITIONS ---------------------------------------------------------
//---------------------------------------------------------------------------
${application.typewriteAllSignalTypedefs()}

//---------------------------------------------------------------------------
// VARIABLES DECLARATIONS ---------------------------------------------------
//---------------------------------------------------------------------------
${application.formApplicationName}${application.separationClassString()}event$
{application.separationClassString()}t* ${application.eventPtrString()};

//---------------------------------------------------------------------------
// FUNCTIONS PROTOTYPES -----------------------------------------------------
//---------------------------------------------------------------------------
void ISR(void);
${application.typewriteTimeEventCreationPrototype()}
${application.typewriteAllEventsOperationsPrototypes()}


//---------------------------------------------------------------------------
#endif
[/#template]
```

```
[#package c.generator]

[#template public classFile(class : fsmapp.Class)]
[#file]generated/${class.getCFileName}[/#file]
/*
*****************************************************************************
* This is a generated file (from the ${class.name} Class of the              *
*${class#application.first().getName()} application)                        *
*****************************************************************************
*                                                                            *
* file      : ${class.getCFileName}                                          *
* template  : classFile.tgt                                                  *
*                                                                            *
* Info      : This file implements the ${class.getName()} fonctionnality of the*
*          application.                                                      *
*                                                                            *
* Author : Hugo Nunes                                                        *
* Date      : 2008-11-07                                                     *
* Version   : 1.0                                                            *
*                                                                            *
*****************************************************************************
*/
//----------------------------------------------------------------------------
// INCLUSIONS ---------------------------------------------------------------
//----------------------------------------------------------------------------
${class.typewriteOwnClassHeaderInclusion}

//----------------------------------------------------------------------------
// PROPERTIES DECLARATION ---------------------------------------------------
//----------------------------------------------------------------------------

//class attributes
${class.typewriteAllProperties()}
//statemachines position
${class.typewriteStateMachineVariables()}

//----------------------------------------------------------------------------
// INITIALISATIONS ----------------------------------------------------------
//----------------------------------------------------------------------------

//class attributes
${class.typewriteAllAttributesInitialisation()}

//----------------------------------------------------------------------------
// FUNCTIONS ----------------------------------------------------------------
//----------------------------------------------------------------------------

${class.typewriteAllFunctionsImpl()}

//----------------------------------------------------------------------------
// STATEMACHINES ------------------------------------------------------------
//----------------------------------------------------------------------------

${class.typewriteAllStateMachines()}

//----------------------------------------------------------------------------
// CLASS BEHAVIOR -----------------------------------------------------------
//----------------------------------------------------------------------------
//void main( void ) {//MAIN START ************************************************
//
//              Behavior of the class
//
//}//MAIN END *********************************************************


[/#template]
```

```
[#package c.generator]

[#template public classHeader(class : fsmapp.Class)]
[#file]generated/${class.getHeaderFileName}[/#file]
/*
*****************************************************************************
* This is a generated file (from the ${class.name} Class of the              *
* ${class#application.first().getName()} application)                        *
*****************************************************************************
*                                                                            *
* file      : ${class.getHeaderFileName}                                     *
* template  : classHeader.tgt                                                *
*                                                                            *
* Info      : This file implements the ${class.getName()} interface of the   *
*          application.                                                      *
*                                                                            *
* Author : Hugo Nunes                                                        *
* Date      : 2008-11-12                                                     *
* Version   : 1.0                                                            *
*                                                                            *
*****************************************************************************
*/
#ifndef ${class.getName().toUpperCase()}_H
#define ${class.getName().toUpperCase()}_H
//---------------------------------------------------------------------------
// INCLUDES ----------------------------------------------------------------
//---------------------------------------------------------------------------
#include "xhal.h"
#include "xf.h"
#include "${class#application.first().getHeaderFileName()}"
${class.typewriteOtherClassHeadersInclusions()}

//---------------------------------------------------------------------------
// ENUMERATIONS ------------------------------------------------------------
//---------------------------------------------------------------------------

//regions vertices enumerations
${class.typewriteAllRegionsVerticesEnum()}[#rtrim]

//regions transitions enumerations
${class.typewriteAllRegionsTransitionsEnum()}[#rtrim]

//---------------------------------------------------------------------------
// TYPE DEFINITIONS --------------------------------------------------------
//---------------------------------------------------------------------------
${class.typewriteEnumsTypeDefs()}

//---------------------------------------------------------------------------
// VARIABLES DECLARATIONS --------------------------------------------------
//---------------------------------------------------------------------------
${class.typewritePublicProperties()}

//---------------------------------------------------------------------------
// FUNCTIONS PROTOTYPES ----------------------------------------------------
//---------------------------------------------------------------------------
${class.typewritePublicFunctionsProto()}[#trim]
${class.typewriteStateMachinesProto()}


//---------------------------------------------------------------------------
#endif
[/#template]
```

```java
package c.generator;

import java.io.StreamTokenizer;
import java.io.StringReader;
import java.util.Collection;
import java.util.Vector;

import com.sodius.mdw.metamodel.fsmapp.scripts.ApplicationScriptContainer;

public class fsmapp_Application extends ApplicationScriptContainer {

// USES PARSER TO GIVE SPECIFIC RESULTS ----------------------------------------
   //Return the attribute name without the type
   public String getStringAttributeName(String input) throws Exception {
      return getStringOperationName(input);
   }

   //Return all the attribute modifiers in a String
   public String getStringAttributeModifiers(String input) throws Exception  {
      return getStringOperationReturnType(input);
   }

   //Return the operation name without the type
   public String getStringOperationName(String input) throws Exception {
      String res1 = (getStringBeforeParenthesis(input));
      if(res1!=null) {
         String[] res2 = (getStringTokens(res1," ").toArray(new String[0]));
         if(res2!=null)
            return res2[res2.length-1];
         else
            return null;
      }
      else
         return null;
   }

   //Return all the operation return type in a String
   public String getStringOperationReturnType(String input) throws Exception {
      String res1 = (getStringBeforeParenthesis(input));
      String[] res2 = (getStringTokens(res1," ").toArray(new String[0]));
      if(res2.length>1) {
         String ret = "";
         for (int i = 0; i < (res2.length)-1; i++) {
            ret = ret+res2[i];
            if(i!=((res2.length)-2))   {
               ret = ret + " ";
            }
         }
         return ret;
      }
      else
         return null;
   }

   //Return the operation parameters in a String
   public String getStringOperationParameters(String input) throws Exception {
      String res1 = (getStringAfterParenthesis(input));
      if(res1!=null)
         return res1.substring(0, (res1.length()-1));
      else
         return null;
   }

// INTERMEDIATE PARSING
   //Returns the String token before the parenthesis
   public String getStringBeforeParenthesis(String input)  throws Exception {
      String[] res = (getStringTokens(input,"(").toArray(new String[0]));
      if(res!=null)
         return res[0];
      else
         return null;
   }
   public String getStringAfterParenthesis(String input)throws Exception {
      String[] res = (getStringTokens(input,"(").toArray(new String[0]));
      if((res!=null) && (res.length==2))
         return res[1];
      else
         return null;
   }
```

```
// PARSER ---------------------------------------------------------------------
   // Returns a List of Strings
   public Collection<String> getStringTokens(String input, String separators) throws Exception {
      //used for result
      Collection<String> res = new Vector<String>();
      //temporary result
      String tempRes = new String();
      //variables
      boolean isSeparator = false;
      boolean needToRemoveFirstChar = true;

      if( input.isEmpty() )   {
      //input String contains nothing
         return null;
      }
      else  {
      //input String contains something
         //creates stream tokenizer
         String   code     = input;
         StreamTokenizer stoken = new StreamTokenizer(new StringReader(code));
         //StreamTokenizer configuration
         stoken.resetSyntax();

         //Process the string
         while(stoken.ttype != StreamTokenizer.TT_EOF)   {
            //check if token is a separator
            for(int i=0; i<separators.length();i++)   {
               if(stoken.ttype == separators.charAt(i))  {
                  isSeparator=true;
               }
            }

            if(isSeparator)   {
            //value is a separator
               if(needToRemoveFirstChar)  {
                  //remove unknown interrogation point  :S ??????
                  tempRes=tempRes.substring(1);
                  needToRemoveFirstChar=false;
               }
               //add new String to List
               res.add(tempRes);
               tempRes = "";
               isSeparator = false;
            }
            else  {
            //add char to String
               tempRes+=(char)(stoken.ttype);
            }
            stoken.nextToken();
         }
         //return results;
         if((tempRes==null)||(tempRes.equals("")) {
            //last String is empty...
            return res;
         }
         else  {
            //add last String result before returning result
            if(needToRemoveFirstChar)  {
               //remove unknown interrogation point  :S ??????
               tempRes=tempRes.substring(1);
            }
            res.add(tempRes);
            return res;
         }
      }
   }
}
```

```
package c.generator;

metatype fsmapp.Application;

//-----------------------------------------------------------------------------
// CONFIGURATIONS -------------------------------------------------------------
//-----------------------------------------------------------------------------

// Returns the Class prefix used in naming functions, properties, ...
public script prefixClassString() : String {
   return "Cl_";
}

// Returns the separation String used in naming functions, properties, ...
public script separationClassString() : String {
   return "_";
}

//FORMATTING SCRIPT
public script formApplicationName() : String {
   return self.name.toUpperCase();
}

//Returns the String representing the Timer Signal Structure
public script formTimerSignalString() : String {
   return self.formApplicationName()+self.separationClassString()+"timerSignal";
}

//
public script timerSignalString() : String {
   return "timerSignal";
}

//Returns the prefix of signals used to call the push to XF
public script signalPushPrefix() : String {
   return "APP1_PUSH_";
}

//Returns the prefix of timer used to call the push to XF
public script timerPushPrefix() : String {
   return "TMR_PUSH_";
}

//Returns the String used as name of the event pointer
public script eventPtrString() : String {
   return self.formApplicationName()+self.separationClassString()+"eventPtr";
}

//Returns a String representing the event type
public script eventTypeString() : String {
   return self.formApplicationName()+self.separationClassString()
+"event"+self.separationClassString()+"t";
}

//Returns a String representing the timer to push
public script timerPushFunction() : String{
   return self.timerPushPrefix()+self.timerSignalString();
}

//-----------------------------------------------------------------------------
// FILE RELATED SCRIPTS -------------------------------------------------------
//-----------------------------------------------------------------------------

// Returns the header file name of this application
public script getHeaderFileName() : String {
   return (((self.getName()).toLowerCase())+".h");
}

// Returns the C file name of this application
public script getCFileName() : String {
   return (((self.getName()).toLowerCase())+".c");
}

//-----------------------------------------------------------------------------
// GENERAL PURPOSE SCRIPTS ----------------------------------------------------
//-----------------------------------------------------------------------------
```

```
//-----------------------------------------------------------------------------
// TRANSIENT LINKS ------------------------------------------------------------
//-----------------------------------------------------------------------------
// Adds usefull transient links for later use
public script addTransientLinks() {
//-- APPLICATION ----------------------------------------------------------

   //-- SIGNAL ------------------------------------------------------
   foreach (signal : fsmapp.Signal in self.getThePackagedElement().select("getTypeName","Signal"))
   {
      //link for containing (parent) application
      signal#application.add(self);
      //-- PROPERTY ------------------------------------------
      foreach (property : fsmapp.Property in signal.getOwnedAttribute()) {
         //link for containing (parent) signal
         property#application.add(self);
         property#signal.add(signal);

         property#formName.add(property.formSignalAttributeName());
      }//= END PROPERTY ======================================

      signal#formName.add(signal.formSignalName());
   }//= END SIGNAL ==================================================

   //-- CLASS -------------------------------------------------------
   //Child Class transient links
   foreach (class : fsmapp.Class in self.getTheClass()) {
      //link for containing (parent) application
      class#application.add(self);
      //link for configurations
      class#prefix.add(self.prefixClassString());
      class#separation.add(self.separationClassString());

      //for use in Class behavior mode
      class#classifierBehavior.add(class.getClassifierBehavior());
      //-- PROPERTY ------------------------------------------
      foreach (property : fsmapp.Property in class.getOwnedAttribute()) {
         //link for containing (parent) application
         property#application.add(self);
         property#class.add(class);

         property#formName.add(property.formAttributeName());
      }//= END PROPERTY ======================================
      //-- OPERATION -----------------------------------------
      foreach (operation : fsmapp.Operation in class.getOwnedOperation()) {
         //link for containing (parent) application
         operation#application.add(self);
         operation#class.add(class);

         operation#formName.add(operation.formOperationName());
      }//= END OPERATION =====================================
      //-- STATEMACHINE --------------------------------------
      foreach (statemachine : fsmapp.StateMachine in class.getAllStateMachines()) {
         //link for containing (parent) application
         statemachine#application.add(self);
         statemachine#class.add(class);
         //-- REGION -------------------------------------
         //System.out.println(statemachine.name  statemachine.getTypeName());
         foreach (region : fsmapp.Region in statemachine.region) {
            region#application.add(self);
            region#class.add(class);
            region#statemachine.add(statemachine);
            region#vertexEnum.add(region.vertexEnumName());
            //-- VERTEX ----------------------------
            foreach (vertex : fsmapp.Vertex in region.getSubvertex()) {
               vertex#application.add(self);
               vertex#class.add(class);
               vertex#statemachine.add(statemachine);
               vertex#region.add(region);
               vertex#indnbr.add(region.getSubvertex().indexOf(vertex));

               vertex#formName.add(vertex.formVertexName());
            }//= END VERTEX ========================
            //-- TRANSITION ------------------------
            foreach (transition : fsmapp.Transition in region.getTransition()) {
               transition#application.add(self);
               transition#class.add(class);
               transition#statemachine.add(statemachine);
```

```
                transition#region.add(region);
                transition#indnbr.add(region.getTransition().indexOf(transition));

                transition#formName.add(transition.formTransitionName());
             }//= END TRANSITION ====================
             region#formName.add(region.formRegionName());
          }//= END REGION ==================================

          statemachine#formName.add(statemachine.formStateMachineName());
       }//= END STATEMACHINE ====================================

       class#formName.add(class.formClassName());
    }//= END CLASS ====================================================

    self#formName.add(self.formApplicationName());
}//= END APPLICATION ======================================================



[#package c.generator]

[#metatype fsmapp.Application]

[#-----------------------------------------------------------------------------]
[#-- FILES AND INCLUDES RELATED SCRIPTS ---------------------------------------]
[#-----------------------------------------------------------------------------]

[#-- TypeWrites all the header inclusions -------------------------------------]
[#script public typewriteAllInclusions]
   [#foreach class : fsmapp.Class in self.getTheClass()]
${self.typewriteClassHeadersInclusions()}[#trim]
${self.typewriteStandardHeadersInclusions()}
   [/#foreach]
[/#script]



[#-- Prints all the headers from the classes that belong to this application --]
[#script public typewriteClassHeadersInclusions]
   [#foreach class : fsmapp.Class in self.getTheClass()]
#include "${class.getHeaderFileName}"
   [/#foreach]
[/#script]



[#-- TypeWrites all the standard headers inclusions ---------------------------]
[#script public typewriteStandardHeadersInclusions]
   [#foreach entry : java.lang.String in
self.theConfig.theSettings.getStandardHeaders_StringCollection]
#include ${entry}
   [/#foreach]
[/#script]



[#-----------------------------------------------------------------------------]
[#-- SIGNAL RELATED SCRIPTS ---------------------------------------------------]
[#-----------------------------------------------------------------------------]

[#-- TypeWrites all the signal structures -------------------------------------]
[#script public typewriteAllSignalStructs]
   [#foreach signal : fsmapp.Signal in self.getThePackagedElement().select("getTypeName","Signal")]
struct ${signal#formName.first()}${self.separationClassString()}struct{
      [#foreach property : fsmapp.Property in signal.getOwnedAttribute()]
   ${property.attributeModifiers()} ${property.attributeName()};
      [/#foreach]
};
   [/#foreach]
struct ${self.formTimerSignalString()}${self.separationClassString()}struct{
   int index;
   int value;
};
[/#script]



[#-- TypeWrites an union of all signal structures -----------------------------]
```

```
[#script public typewriteSignalsUnion]
union ${self.formApplicationName}${self.separationClassString()}sigUnion{
   [#foreach signal : fsmapp.Signal in self.getThePackagedElement().select("getTypeName","Signal")]
   struct ${signal#formName.first()}${self.separationClassString()}struct ${signal.name};
   [/#foreach]
   struct ${self.formTimerSignalString()}${self.separationClassString()}struct $
{self.timerSignalString()};
};
[/#script]


[#-- TypeWrites all the signal event typedef names ----------------------------]
[#script public typewriteAllSignalTypedefs]
   [#foreach signal : fsmapp.Signal in self.getThePackagedElement().select("getTypeName","Signal")]
typedef struct ${signal#formName.first()}${self.separationClassString()}struct $
{signal#formName.first()}${self.separationClassString()}t;
   [/#foreach]
typedef union ${self.formApplicationName}${self.separationClassString()}sigUnion $
{self.formApplicationName}${self.separationClassString()}signalU;
typedef struct ${self.formApplicationName}${self.separationClassString()}event$
{self.separationClassString()}struct ${self.eventTypeString()};
[/#script]


[#--TypeWrites all the signals enumeration ------------------------------------]
[#script public typewriteAllSignalsEnum]
//enumeration of signals
enum ${self.formApplicationName}${self.separationClassString()}SIGNALS$
{self.separationClassString()}enum{
   ${self.formTimerSignalString()}[#trim]
[#if (self.getThePackagedElement().select("getTypeName","Signal")).size()!=0],[/#if]
   [#foreach signal : fsmapp.Signal in self.getThePackagedElement().select("getTypeName","Signal")]
   ${signal#formName.first()}[#trim]
[#if (self.getThePackagedElement().select("getTypeName","Signal")).last()!=signal],[/#if]
   [/#foreach][#trim]
};
[/#script]


[#----------------------------------------------------------------------------]
[#-- EVENT RELATED SCRIPTS ---------------------------------------------------]
[#----------------------------------------------------------------------------]

[#-- TypeWrites the Events structure ----------------------------------------]
[#script public typewriteEventStruct]
struct ${self.formApplicationName}${self.separationClassString()}event$
{self.separationClassString()}struct{
   int type;
   int dest;
   union ${self.formApplicationName}${self.separationClassString()}sigUnion signals;
};
[/#script]


[#-- TypeWrites the signal events push operations prototypes ------------------]
[#script public typewriteAllEventsOperationsPrototypes]
[#foreach signal : fsmapp.Signal in self.getThePackagedElement().select("getTypeName","Signal")]
int ${self.signalPushPrefix()}${signal.name} ([#trim]
[#foreach property : fsmapp.Property in signal.getOwnedAttribute()][#trim]
${property.getName()}[#if !(signal.getOwnedAttribute().last()==property)],[/#if][#trim]
[/#foreach]);
[/#foreach]
[/#script]

[#-- TypeWrites the time event push operation prototypes ----------------------]
[#script public typewriteTimeEventCreationPrototype]
int ${self.timerPushPrefix()}${self.timerSignalString()} (int value,int dest);
[/#script]


[#-- TypeWrites the signal events push operations implemetation ---------------]
[#script public typewriteAllEventsOperationsImplementation]
   [#foreach signal : fsmapp.Signal in self.getThePackagedElement().select("getTypeName","Signal")]
int ${self.signalPushPrefix()}${signal.name} ([#trim]
```

```
[#foreach property : fsmapp.Property in signal.getOwnedAttribute()][#trim]
${property.getName()}[#if !(signal.getOwnedAttribute().last()==property)],[/#if][#trim]
[/#foreach]) {
   ${self.eventPtrString()} = malloc(sizeof(${self.eventTypeString()}));
   [#foreach property : fsmapp.Property in signal.getOwnedAttribute()]
   (${self.eventPtrString()}->signals).${signal.name}.${property.attributeName()}=
${property.attributeName()};
   [/#foreach]
   (${self.eventPtrString()}->type)=${signal.formSignalName()};
   (${self.eventPtrString()}->dest)=0;
   _XF_push(${self.eventPtrString()},0,0);
   ${self.eventPtrString()} = NULL;
};
   [/#foreach]
${self.typewriteTimeEventCreationImplementation()}
[/#script]


[#-- TypeWrites the time event push operation implementation ------------------]
[#script public typewriteTimeEventCreationImplementation]
int ${self.timerPushPrefix()}${self.timerSignalString()} (int value,int dest) {
   ${self.eventPtrString()} = malloc(sizeof(${self.eventTypeString()}));
   (${self.eventPtrString()}->signals).${self.timerSignalString()}.value = value;
   (${self.eventPtrString()}->type)=${self.formTimerSignalString()}:
   (${self.eventPtrString()}->dest)=dest;
   _XF_tmPush(${self.eventPtrString()},0,value,0);
};
[/#script]



[#-- TypeWrites the event pointer memory free and clear functionnality --------]
[#script public typewriteFreeEvent]
   free(${self.eventPtrString});
   ${self.eventPtrString} = NULL;
[/#script]

[#-- TypeWrites the pop of XF and setting of event pointer --------------------]
[#script public typewriteGetEvent]
   _XF_pop(${self.eventPtrString},0);
[/#script]
```

```
package c.generator;

metatype fsmapp.Behavior;

public script getTypeName() : String {
   return self.eClass().name;
}
```

```
package c.generator;

metatype fsmapp.Class;

// FILE RELATED SCRIPTS -------------------------------------------------------

// Returns the header file name of this class
public script getHeaderFileName() : String {
   return (((self.getName()).toLowerCase())+".h");
}

// Returns the C file name of this class
public script getCFileName() : String {
   return (((self.getName()).toLowerCase())+".c");
}

// STATEMACHINE RELATED SCRIPTS ------------------------------------------------

//Returns the first state machine in the nested classifiers
public script getStateMachine() {
   var behaviorc : fsmapp.Behavior = self.getClassifierBehavior();
   if(behaviorc.eClass().name == "StateMachine" ) {
      return behaviorc;
```

```
   }
   else {
      foreach (behavior : fsmapp.Behavior in self.getOwnedBehavior())  {
         if(behavior.eClass().name == "StateMachine" )  {
            return behavior;
         }
      }
   }
   return null;
}

public script getAllStateMachines() {
 return self.getOwnedBehavior().select("getTypeName","StateMachine");
}

// CLASS RELATED SCRIPTS -------------------------------------------------------




// PROPERTY RELATED SCRIPTS ----------------------------------------------------

//FORMATTING SCRIPTS -----
public script formClassName() : String{
   return self.getName().toUpperCase();
}
//END FORMATTING SCRIPTS -----



[#package c.generator]

[#metatype fsmapp.Class]
[#-----------------------------------------------------------------------------]
[#-- ATTRIBUTS RELATED SCRIPTS -----------------------------------------------]
[#-----------------------------------------------------------------------------]

[#--TypeWrites all the attribute bodies ---------------------------------------]
[#script public typewriteAllProperties]
   [#foreach property : fsmapp.Property in self.getOwnedAttribute()]
      [#if property.getLiteralStringValueSpecification()!= null]
${property.attributeModifiers()} ${property.formAttributeName()};
      [/#if]
   [/#foreach]
[/#script]


[#--TypeWrites the public attribute bodies ------------------------------------]
[#script public typewritePublicProperties]
   [#foreach property : fsmapp.Property in self.getOwnedAttribute()]
      [#if property.visibility.toString().equals("public")]
         [#if property.getLiteralStringValueSpecification()!= null]
${property.attributeModifiers()} ${property.formAttributeName()};
         [/#if]
      [/#if]
   [/#foreach]
[/#script]


[#--TypeWrites the private attribute bodies -----------------------------------]
[#script public typewritePrivateProperties]
   [#foreach property : fsmapp.Property in self.getOwnedAttribute()]
      [#if property.visibility.toString().equals("private")]
         [#if property.getLiteralStringValueSpecification()!= null]
${property.attributeModifiers()} ${property.formAttributeName()};
         [/#if]
      [/#if]
   [/#foreach]
[/#script]


[#--TypeWrites all the attributes initialisation ------------------------------]
[#script public typewriteAllAttributesInitialisation]
   [#foreach property : fsmapp.Property in self.getOwnedAttribute()]
      [#if property.getLiteralStringValueSpecification().getValue() != null]
${property.attributeModifiers()} ${property.formAttributeName()} = $
{property.getLiteralStringValueSpecification().getValue()};
      [/#if]
   [/#foreach]
```

```
[/#script]


[#-----------------------------------------------------------------------------]
[#-- OPERATION RELATED SCRIPTS ------------------------------------------------]
[#-----------------------------------------------------------------------------]

[#--TypeWrites all the functions prototypes -----------------------------------]
[#script public typewriteAllFunctionsProto]
   [#foreach operation : fsmapp.Operation in self.getOwnedOperation()]
${operation.operationReturnType()} ${operation.formOperationName()}($
{operation.operationParameters()});
   [/#foreach]
[/#script]


[#--TypeWrites the public functions prototypes -------------------------------]
[#script public typewritePublicFunctionsProto]
   [#foreach operation : fsmapp.Operation in self.getOwnedOperation()]
      [#if operation.visibility.toString().equals("public")]
${operation.operationReturnType()} ${operation.formOperationName()}($
{operation.operationParameters()});
      [/#if]
   [/#foreach]
[/#script]


[#--TypeWrites the private functions prototypes ------------------------------]
[#script public typewritePrivateFunctionsProto]
   [#foreach operation : fsmapp.Operation in self.getOwnedOperation()]
      [#if operation.visibility.toString().equals("private")]
${operation.operationReturnType()} ${operation.formOperationName()}($
{operation.operationParameters()});
      [/#if]
   [/#foreach]
[/#script]


[#--TypeWrites all the functions implementation ------------------------------]
[#script public typewriteAllFunctionsImpl]
   [#foreach operation : fsmapp.Operation in self.getOwnedOperation()]
${operation.operationReturnType()} ${operation.formOperationName()}($
{operation.operationParameters()}) {
      [#foreach opaqueBehavior : fsmapp.OpaqueBehavior in operation.getMethod()]
//This body corresponds to ${opaqueBehavior.name} used as Method of ${operation.formOperationName()}
${opaqueBehavior.body}
      [/#foreach]
}

   [/#foreach]
[/#script]


[#-----------------------------------------------------------------------------]
[#-- STATEMACHINE RELATED SCRIPTS ---------------------------------------------]
[#-----------------------------------------------------------------------------]

[#--TypeWrites the prototypes of all the statemachines-------------------------]
[#script public typewriteStateMachinesProto]
   [#foreach statemachine : fsmapp.StateMachine in self.getAllStateMachines()]
${statemachine.statemachineReturnType()} ${statemachine.formStateMachineName()}($
{statemachine.statemachineParameters()});
   [/#foreach]
[/#script]



[#--TypeWrites the implementation of all the statemachines---------------------]
[#script public typewriteAllStateMachines]
   [#foreach statemachine : fsmapp.StateMachine in self.getAllStateMachines()]
// STATE MACHINE IMPLEMENTATION ----------------------------
${statemachine.statemachineReturnType()} ${statemachine.formStateMachineName()}($
{statemachine.statemachineParameters()}) {

[#-- Initialize the Region variables that have at least a Pseudostate initial state--]
   // INITIALIZATIONS -------------------------------

//regions initialize
```

```
      [#foreach region : fsmapp.Region in statemachine.getRegion().select("hasInitialState","true")]
   //region ${region.name}
${region.vertexEnumName} ${region.oldPosition()};
${region.transitionEnumName} ${region.activatedTransition()};
char ${region.changedStateVar()} = 0x00;
${region.currentPosition()} = ${region.getInitialState.formVertexName()};
${region.oldPosition()} = ${region.currentPosition()};
${region.activatedTransition()} = ${region.noneTransition()};

      [/#foreach]


//-->LOOP<-- to EXIT, ALL regions must have reached final state
while(TRUE) {
[#-- REGION BEHAVIOR IMPLEMENTATION --]
   // POP AN EVENT -----------------------------------
${self#application.first().typewriteGetEvent()}

   // COMPUTE CHANGES IN REGIONS ---------------------
         [#foreach region : fsmapp.Region in statemachine.getRegion()]
         [#if !(region.hasInitialState())]
//Region ${region.name} cannot be implemented because it is not valid
//(it contains no initial state)
         [/#if]
      [/#foreach]
${statemachine.typewriteRegionChangeCheck()}[#-- implemented in StateMachine --]

   // START AND PUSH TIMERS -------------------------
${statemachine.typewriteTimersCheckAndStart()}[#-- implemented in StateMachine --]

   // OUTPUT ACTIONS ---------------------------------
${statemachine.typewriteRegionExecuteActions()}[#-- implemented in StateMachine --]

   // CLEAR VARS AND EVENT ---------------------------
${statemachine.typewriteClearVars()}[#-- implemented in StateMachine --]
${self#application.first().typewriteFreeEvent()}[#-- implemented in Application --]

   // DO ACTIVITY ------------------------------------
${statemachine.typewriteDoActivity()}[#-- implemented in StateMachine --]

}//end of LOOP
}// END OF STATE MACHINE -----------------------------------

   [/#foreach]
[/#script]


[#--TypeWrites the required declarations for the statemachines-----------------]
[#script public typewriteStateMachineVariables]
   [#foreach statemachine : fsmapp.StateMachine in self.getAllStateMachines()]
      [#foreach region : fsmapp.Region in statemachine.getRegion().select("hasInitialState","true")]
${region.vertexEnumName} ${region.currentPosition()};
      [/#foreach]
   [/#foreach]
[/#script]



[#-----------------------------------------------------------------------------]
[#-- REGION RELATED SCRIPTS ---------------------------------------------------]
[#-----------------------------------------------------------------------------]

[#--TypeWrites all the vertices from regions ----------------------------------]
[#script public typewriteAllRegionsVerticesEnum]
   [#foreach statemachine : fsmapp.StateMachine in
self.getOwnedBehavior().select("getTypeName","StateMachine")]
      [#foreach region : fsmapp.Region in statemachine.getRegion().select("hasInitialState","true")]
//enumeration of vertices in ${region.formRegionName} of ${statemachine.formStateMachineName}
enum ${region.vertexEnumName()} {
         [#foreach vertex : fsmapp.Vertex in region.getSubvertex()]
            [#if vertex != region.getSubvertex().last()]
   ${vertex#formName.first()},
            [#else]
   ${vertex#formName.first()}
            [/#if]
         [/#foreach]
};
      [/#foreach]
   [/#foreach]
```

```
[/#script]


[#--TypeWrites all the transitions from regions ------------------------------]
[#script public typewriteAllRegionsTransitionsEnum]
   [#foreach statemachine : fsmapp.StateMachine in
self.getOwnedBehavior().select("getTypeName","StateMachine")]
      [#foreach region : fsmapp.Region in statemachine.getRegion().select("hasInitialState","true")]
//enumeration of transitions in ${region.formRegionName} of ${statemachine.formStateMachineName}
enum ${region.transitionEnumName()} {
         [#if region.getTransition().size()!=0]
   ${region.noneTransition()},
         [#else]
   ${region.noneTransition()}
         [/#if]
         [#foreach transition : fsmapp.Transition in region.getTransition()]
            [#if transition != region.getTransition().last()]
   ${transition#formName.first()},
            [#else]
   ${transition#formName.first()}
            [/#if]
         [/#foreach]
};
      [/#foreach]
   [/#foreach]
[/#script]


[#--TypeWrites the typedefs of elements of regions ---------------------------]
[#script public typewriteEnumsTypeDefs]
   [#foreach statemachine : fsmapp.StateMachine in
self.getOwnedBehavior().select("getTypeName","StateMachine")]
      [#foreach region : fsmapp.Region in statemachine.getRegion().select("hasInitialState","true")]
typedef enum ${region.vertexEnumName()} ${region.vertexEnumName()};
typedef enum ${region.transitionEnumName()} ${region.transitionEnumName()};
      [/#foreach]
   [/#foreach]
[/#script]



[#-----------------------------------------------------------------------------]
[#-- FUNCTIONNAL RELATED SCRIPTS ----------------------------------------------]
[#-----------------------------------------------------------------------------]


[#-----------------------------------------------------------------------------]
[#-- VERTEX RELATED SCRIPTS ---------------------------------------------------]
[#-----------------------------------------------------------------------------]


[#-----------------------------------------------------------------------------]
[#-- FILE RELATED SCRIPTS -----------------------------------------------------]
[#-----------------------------------------------------------------------------]

[#--TypeWrites all the headers from other classes ----------------------------]
[#script public typewriteOtherClassHeadersInclusions]
   [#foreach class : fsmapp.Class in (self#application.first().getTheClass())]
      [#if class != self]
#include "${class.getHeaderFileName}"
      [/#if]
   [/#foreach]
[/#script]


[#--TypeWrites the header of own class ---------------------------------------]
[#script public typewriteOwnClassHeaderInclusion]
#include "${self.getHeaderFileName}"
[/#script]


[#--TypeWrites the header of application header ------------------------------]
[#script public typewriteApplicationHeaderInclusion]
#include "${self#application.first().getHeaderFileName}"
[/#script]
```

```
package c.generator;

metatype fsmapp.Config;

public script hasSettings() : boolean {
   setting = self.getTheSettings();
   if(setting.additionnal_sources.isEmpty())
      if(setting.include_paths.isEmpty())
         if(setting.libraries.isEmpty())
            if(setting.standard_headers.isEmpty())
               return false;
   return true;
}
```

```
package c.generator;

metatype fsmapp.Constraint;

public script containsValueElse() : boolean {
   if(self.getSpecificationLiteralString.value == "else") {
      return true;
   }
   return false;
}

public script getSpecificationLiteralString() : fsmapp.LiteralString {
   if(self.specification.eClass().name == "LiteralString") {
      return self.specification;
   }
   return null;
}
```

```
package c.generator;

metatype fsmapp.Operation;

//FORMATTING SCRIPTS -----
public script formOperationName() : String  {
   return (self#class.first().getName().toUpperCase())  +
(self#application.first().separationClassString())+(self.operationName());
}
//END FORMATTING SCRIPTS -----

//PARSING SCRIPTS -----
public script operationName() : String   {
   return self#application.first().getStringOperationName(self.getName());
}

public script operationReturnType() : String{
   if(self#application.first().getStringOperationReturnType(self.getName())==null)
      return "void";
   else
      return self#application.first().getStringOperationReturnType(self.getName());
}

public script operationParameters() : String{
   if(self#application.first().getStringOperationParameters(self.getName())==null)
      return "void";
   else
      return self#application.first().getStringOperationParameters(self.getName());
}
//END PARSING SCRIPTS -----
```

```
package c.generator;

metatype fsmapp.PackageableElement;

public script getTypeName() : String {
   return self.eClass().name;
}
```

```
package c.generator;

metatype fsmapp.Property;

public script getLiteralStringValueSpecification() : fsmapp.LiteralString {
   if(self.getDefaultValue().eClass().name == "LiteralString")   {
      return self.getDefaultValue();
   }
   else return null;
}

//FORMATTING SCRIPTS -----
public script formAttributeName() : String  {
   return (self#class.first().getName().toUpperCase())+
(self#application.first().separationClassString())+(self.attributeName());
}

public script formSignalAttributeName() : String  {
   return (self#signal.first().getName().toUpperCase())+
(self#application.first().separationClassString())+(self.attributeName());
}
//END FORMATTING SCRIPTS -----

//PARSING SCRIPTS -----
public script attributeName() : String   {
   return self#application.first().getStringAttributeName(self.name);
}

public script attributeModifiers() : String {
   return self#application.first().getStringAttributeModifiers(self.name);
}
//END PARSING SCRIPTS -----
```

```
package c.generator;

metatype fsmapp.Region;

//returns true if this region disposes of a pseudostate of kind initial
public script hasInitialState() : boolean {
      foreach (pseudostate : fsmapp.Pseudostate in
self.getSubvertex().select("getTypeName","Pseudostate")){
            if(pseudostate.kind.toString().equals("initial"))  {
               return true;
            }
      }
   return false;
}

//returns the first pseudostate of kind initial found
public script getInitialState() : com.sodius.mdw.metamodel.fsmapp.Pseudostate {
   foreach (pseudostate : fsmapp.Pseudostate in
self.getSubvertex().select("getTypeName","Pseudostate")){
      if(pseudostate.kind.toString().equals("initial"))  {
         return pseudostate;
      }
   }
   return null;
}

//returns the first finalState found
public script getFinalState() : com.sodius.mdw.metamodel.fsmapp.FinalState {
   foreach (finalState : fsmapp.FinalState in
self.getSubvertex().select("getTypeName","FinalState"))  {
      return finalState;
   }
   return null;
```

```
}

//FORMATTING SCRIPTS -----
public script formRegionName() : String  {
   return (self#class.first()#formName.first())+(self#application.first().separationClassString())+
(self.getName())+"";
}

//returns the string representing the vertices enumeration of this region
public script vertexEnumName() : String {
   return self.formRegionName()+(self#application.first().separationClassString())+"vertexEnum";
}

//returns the string representing the transitions enumeration of this region
public script transitionEnumName() : String {
   return self.formRegionName()+(self#application.first().separationClassString())+"transitionEnum";
}

//returns a string representing the "no transition activated"
public script noneTransition() : String {
   return self.name.toUpperCase() + self#application.first().separationClassString() + "NONE";
}

//returns the actual vertex position of the region
public script currentPosition() {
   return self.formRegionName()+(self#application.first().separationClassString())+"pos";
}

//returns the last vertex position of the region before a transition
public script oldPosition() {
   return self.formRegionName()+(self#application.first().separationClassString())+"old";
}

//returns the actual activated transition
public script activatedTransition() {
   return self.formRegionName()+(self#application.first().separationClassString())+"actTrans";
}

//returns the var name used to check changes in region
public script changedStateVar() : String{
   return self.name.toUpperCase() + self#application.first().separationClassString() + "hasChanged";
}

//END FORMATTING SCRIPTS -----
```

```
package c.generator;

metatype fsmapp.Transition;

public script formTransitionName() : String {
   return self#region.first().name.toUpperCase()+self#application.first().separationClassString()
+self#indnbr.first()+self.name;
}

public script hasGuard() : boolean {
   if((self.guard != null)&&(self.getGuardLiteralString() != null)) {
      return true;
   }
   return false;
}

public script getGuardValue() : String {
   return self.getGuardLiteralString().value;
}

public script guardValueIsElse() : boolean {
   if(self.guard!=null) {
      return self.guard.containsValueElse();
   }
   return false;
}

public script getGuardLiteralString() : fsmapp.LiteralString {
   if(self.guard.specification.eClass().name == "LiteralString") {
      return self.guard.specification;
   }
   return null;
```

```
}

public script hasEffect() : boolean {
   if((self.getEffect()!=null)&&(self.getEffectActionString()!=null)) {
      return true;
   }
   return false;
}

public script getEffectActionOpaqueBehavior() : fsmapp.OpaqueBehavior {
   if(self.getEffect().eClass().name == "OpaqueBehavior") {
      return self.getEffect();
   }
   return null;
}

public script getEffectActionString() : String {
   return self.getEffectActionOpaqueBehavior().body;
}

public script hasTrigger() : boolean {
   if(self.getTrigger() != null) {
      foreach (trigger : fsmapp.Trigger in self.getTrigger()) {
         if (trigger.event != null)
            return true;
      }
   }
   return false;
}

public script hasTimeTrigger() : boolean {
   if( self.hasTrigger() ) {
      foreach (trigger : fsmapp.Trigger in self.getTrigger()) {
         if(trigger.event.eClass().name == "TimeEvent")
            return true;
      }
   }
   return false;
}
```

```
[#package c.generator]

[#metatype fsmapp.Transition]

[#--TypeWrites the transition event check ----------------]
[#script public typewriteTransitionTriggerEventCheck]
   [#if self.hasTrigger()][#trim]
         if([#trim]
      [#foreach trigger : fsmapp.Trigger in self.getTrigger()][#trim]
         [#if trigger.isTimeTrigger][#trim]
            ((${self#application.first().eventPtrString()}->type == $
{self#application.first().formTimerSignalString()})[#trim]
            &&(${self#application.first().eventPtrString()}->dest ==${self.formTransitionName()}))
[#trim]
            [#if (self.getTrigger().last()!=trigger)][#trim]
            ||[#trim]
            [/#if]
         [#else]
            (${self#application.first().eventPtrString()}->type == $
{trigger.getTriggerSignalEnumString()})[#trim]
            [#if (self.getTrigger().last()!=trigger)][#trim]
            ||[#trim]
            [/#if][#trim]
         [/#if][#trim]
      [/#foreach])[#trim]
   [/#if][#trim]
[/#script]
```

```
package c.generator;

metatype fsmapp.Trigger;

public script isTimeTrigger() : boolean{
   if ((self.event!=null)&&(self.event.eClass().name == "TimeEvent"))
      return true;
   return false;
}

public script getTimeTriggerValue() : int{
   var timeEvent : fsmapp.TimeEvent = self.event;
   var timeExpr : fsmapp.TimeExpression = timeEvent.when;
   var literalInteger : fsmapp.LiteralInteger = timeExpr.expr;
   return literalInteger.value;
}

public script getTriggerSignalEnumString() : String{
   if(self.getEvent().eClass().name == "SignalEvent"){
      var signalEvent : fsmapp.SignalEvent = self.getEvent();
      if(signalEvent.signal!=null)
         return signalEvent.signal.formSignalName();
   }
   return null;
}
```

```
package c.generator;

metatype fsmapp.Vertex;

public script getTypeName() : String {
   return self.eClass().name;
}

public script formVertexName() : String {
   return self#region.first().name.toUpperCase()+self#application.first().separationClassString()
+self#indnbr.first()+self.name;
}

public script hasTimerOnOutgoing() : boolean {
   foreach (transition : fsmapp.Transition in self.getOutgoing()) {
      if(transition.hasTimeTrigger())
         return true;
   }
   return false;
}
```

Hes·so VALAIS WALLIS

TUTORIAL

TOPCASED

From MetaModel to Graphical Editor

Hugo Nunes

2008

Table of Contents

Introduction

This document shows step by step how to get a graphical editor from a metamodel.

That procedure is composed of three main parts :

- Creation of the metamodel
- Generation of the tree structure editor
- Generation of the graphical editor

The first main part, creation of a metamodel will not be detailed in this document. The creation of a metamodel implies a knowledge of graphical modeling languages such as UML. Introducing these languages are not the purpose of this document, however some basic steps are given.

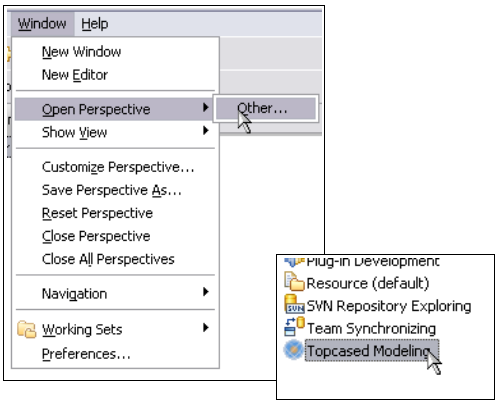
The generation of a tree structure editor from the metamodel and the generation of the graphical editor will be explained step by step.

The purpose of this document is to provide sufficient knowledge so that making a graphical editor could become an easy task once the metamodel is defined.

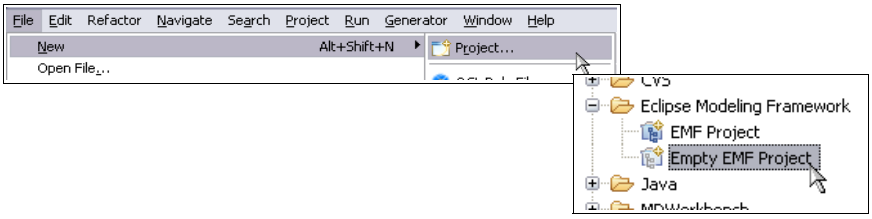
Metamodel Creation

Here a sample metamodel will be created so we can have an example to use in the next parts of this tutorial.

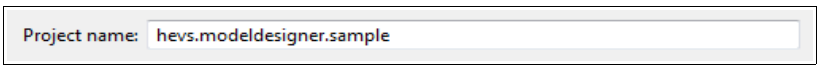
First of all, open the TopCased Perspective :



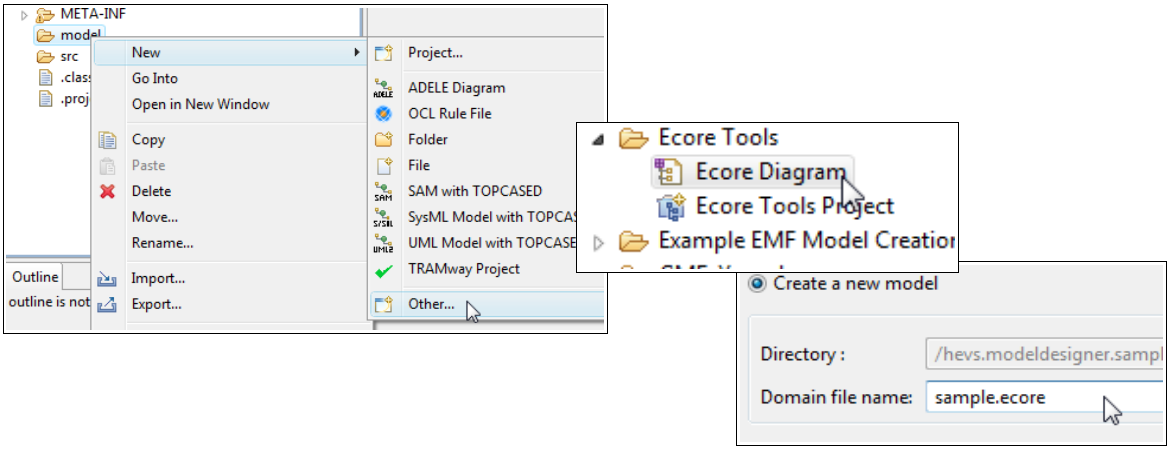
Then you create a new Project of type Eclipse Modeling Framework (EMF) :



Choose a name for your project...

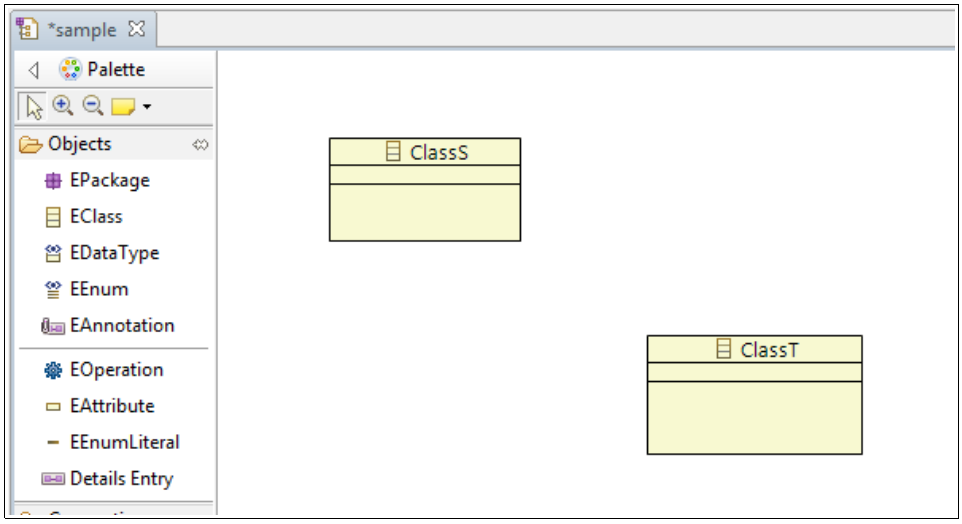


Now you'll need to create your new metamodel. Use the model folder of your project and select a new Ecore Diagram, don't forget to enter the name :

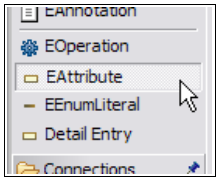


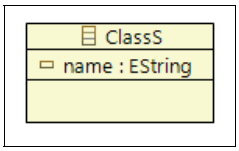
At this moment, you have created your project. Now you can use the tools to draw your metamodel.

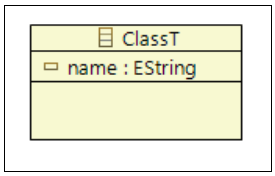
First, draw the EClass model objects :



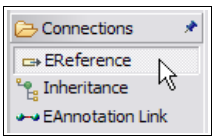
Then, add the EAttibutes to the EClasses :

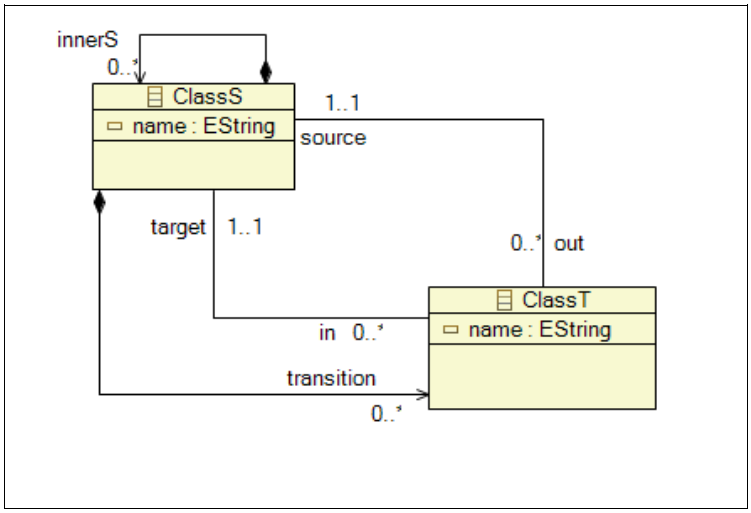






And finally, create all the required connections :



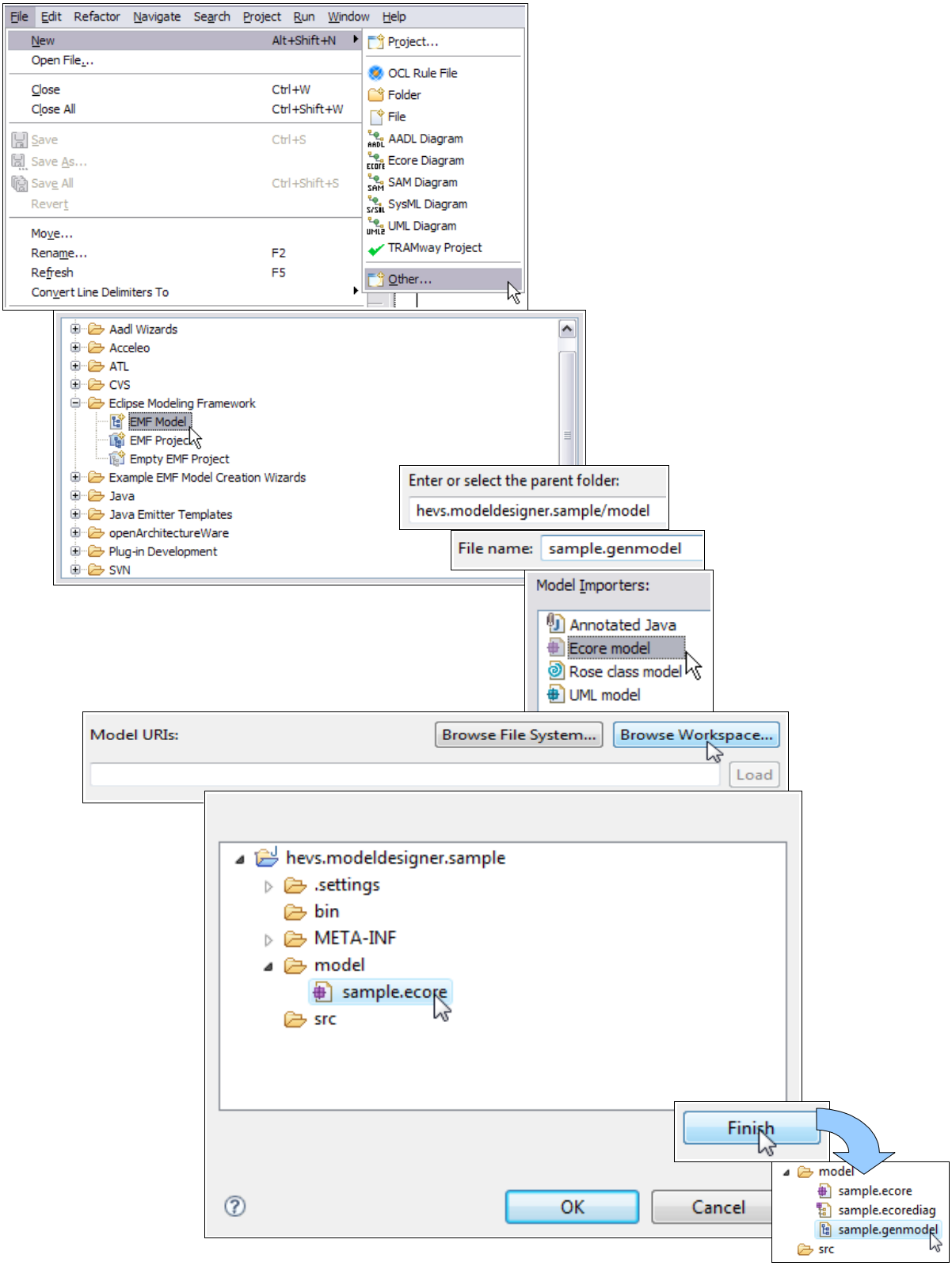


Use all the tools you require to create and draw your metamodel.

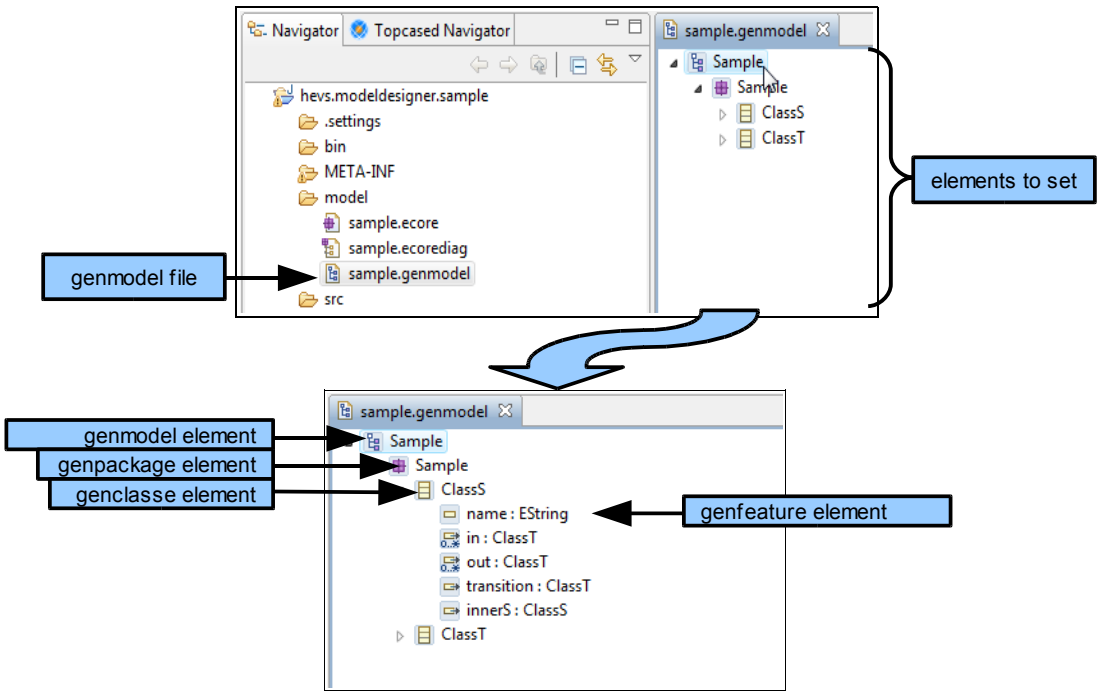
GenModel file

Once your metamodel is created, you must create the **genmodel** file. This file defines what will be generated at the creation of the **tree strucuture editor**.

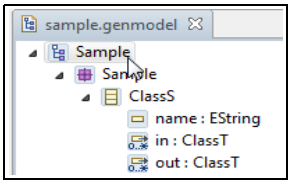
How to create the **genmodel** file (just follow the steps below) :



Configuration of the Genmodel file



GenModel element



Configuration of the root element of theGenModel.
On the properties pane you will have to set :

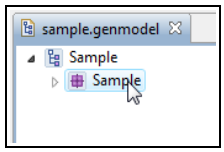
Edit			
Color Providers	false		
Creation Commands	true		
Creation Icons	true		
Edit Directory	/hevs.modeldesigner.sample.edit/src	Edit Directory	/hevs.modeldesigner.sample/src
Edit Plug-in Class	sample.provider.SampleEditPlugin		
Edit Plugin ID	hevs.modeldesigner.sample.edit		

Editor			
Creation Sub-menus	false		
Editor Directory	/hevs.modeldesigner.sample.editor/src	Editor Directory	/hevs.modeldesigner.sample/src
Editor Plug-in Class	sample.presentation.SampleEditorPlugin		

Model			
Array Accessors	false		
Binary Compatible Reflective Methods	false		
Containment Proxies	false		
Feature Delegation	None		
Generate Schema	false		
Minimal Reflective Methods	true		
Model Directory	/hevs.modeldesigner.sample/src		
Model Plug-in Class		Model Plug-in Class	hevs.modeldesigner.sample.SamplePlugin

Tests			
Tests Directory	/hevs.modeldesigner.sample.tests/src	Tests Directory	
Tests Plugin ID	hevs.modeldesigner.sample.tests		
Tests Plugin Variables			
Test Suite Class	sample.tests.SampleAllTests	Test Suite Class	

GenPackage element



On the properties pane you will have to set :

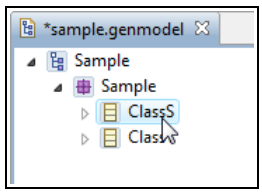
All	
Base Package	
Prefix	Sample

Base Package	hevs.modeldesigner

Model	
Adapter Factory	true
Content Type Identifier	
Data Type Converters	false
File Extensions	sample
Initialize by Loading	false
Literals Interface	true
Resource Type	None

Resource Type	XMI

GenClasses elements

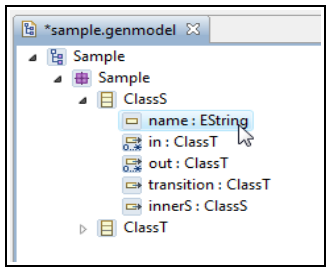


It's better to display a label to represent the model object.

Property	Value
Ecore	
Class	ClassS
Edit	
Image	true
Label Feature	name : EString
Provider Type	Singleton
Model	
Dynamic	false

Choose it here !

GenFeatures elements



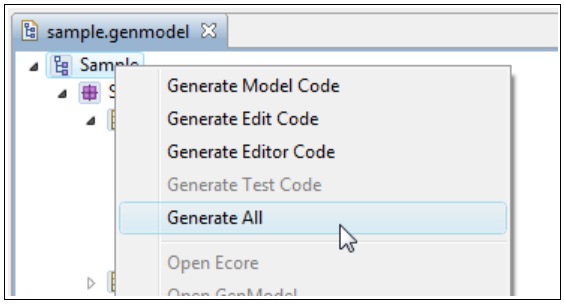
You can specify how the properties will be displayed later in the Properties View frame.

Default values are OK.

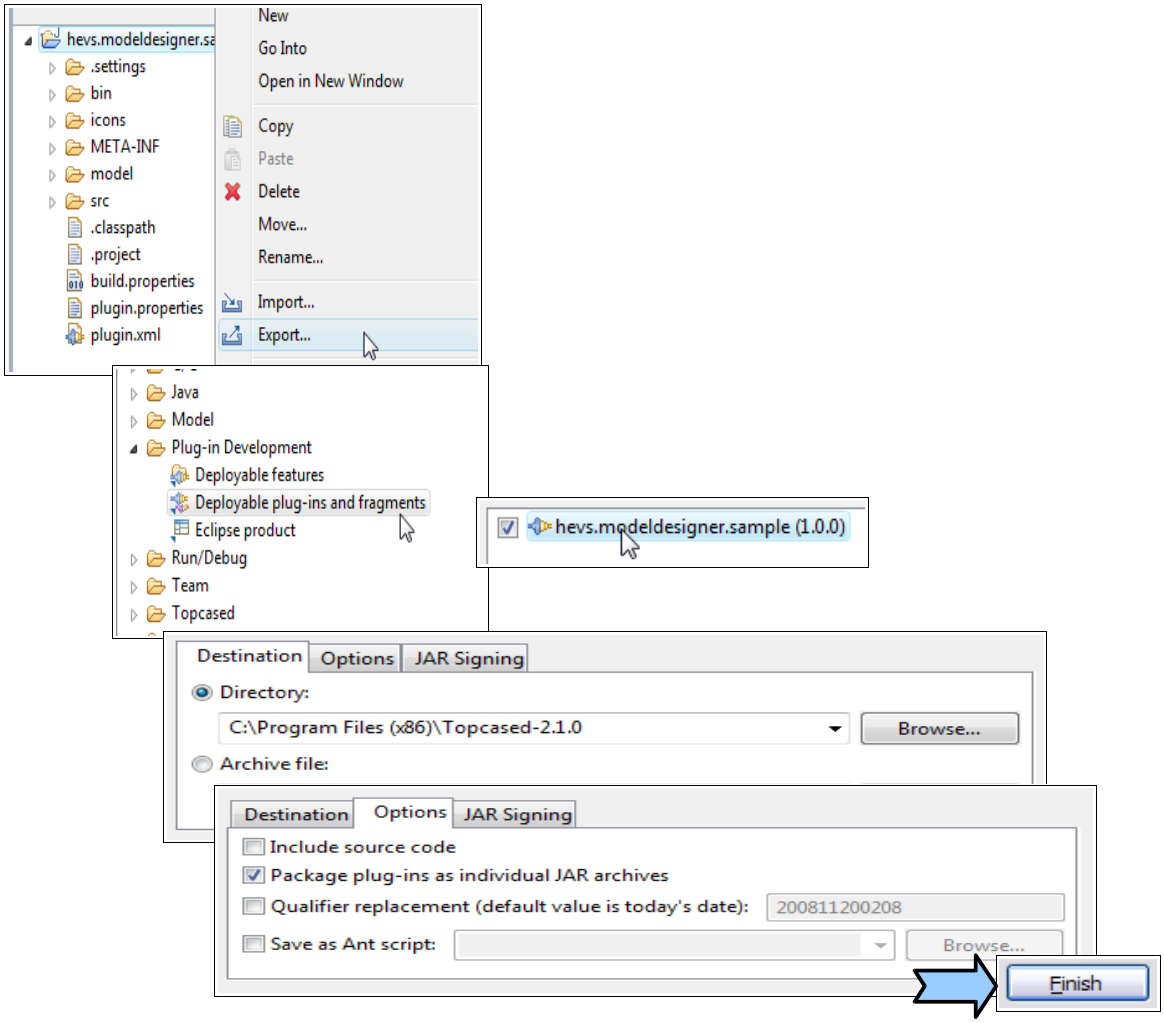
EMF Tree Structure Editor

Once the **genmodel** file is configurated, you'll be able to generate the **tree strucuture editor**. With this editor, you'll be able to create simple models. They can be usefull to verify your metamodel.

How to generate the EMF tree strucuture editor :



How to export the Tree Structure Editor Plugin (follow the steps below) :



Eclipse (or TOPCASED) must be restarted in order to reload the plugins...

Graphical Editor Generation

You've created some models and are satisfied with your metamodel. Let's show you the next step : the **graphical editor**. Here's how to create it.

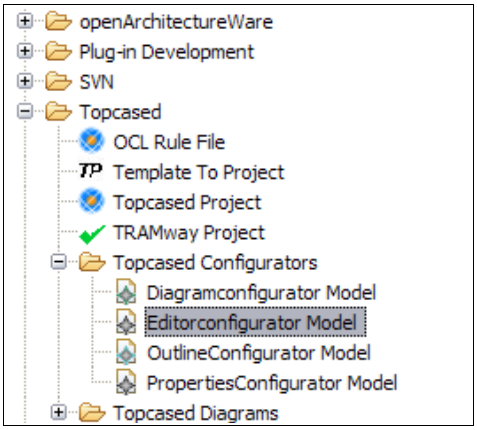
The first step is to create two files that TOPCASED will need.
The two files are :

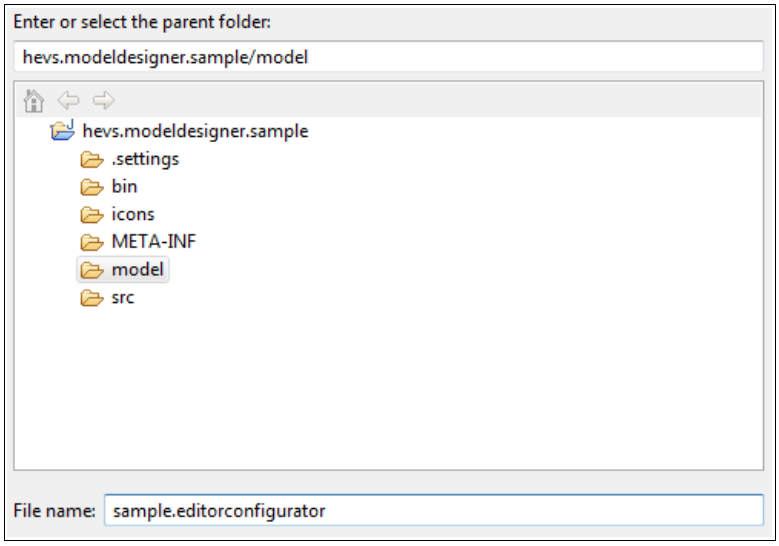
- the **Editor Configurator**
- the **Diagram Configurator**

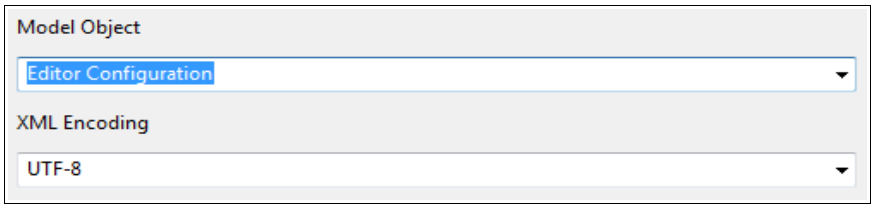
Once these files are correctly configurated, you'll be able to generate a graphical editor with TOPCASED. This editor provide a much more convivial interface to create your models. Most part of the settings will already be set by default but some of them require a particular attention. The examples provided here can be considered as a basis. They'll grant you a succesfull generation of the graphical editor. But fell free to adapt some of the settings to your needs.

Editor Configurator

The first file you'll need is an **Editor Configurator**.

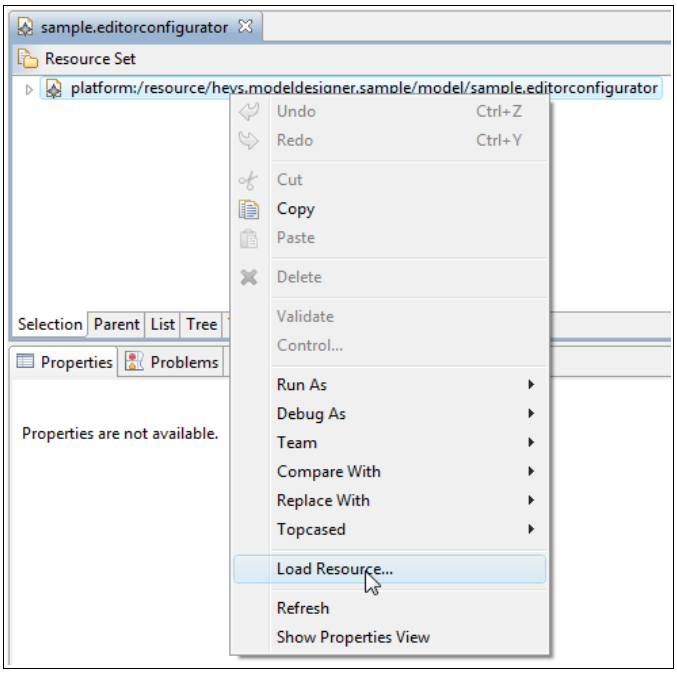






To be able to set some properties, you'll need to load the **genmodel** file as a **resource**.

Load Resource :



Once it's done, you'll be able to apply the configuration in the "Properties Pane".
Here the kind of configuration that you'll need to have :

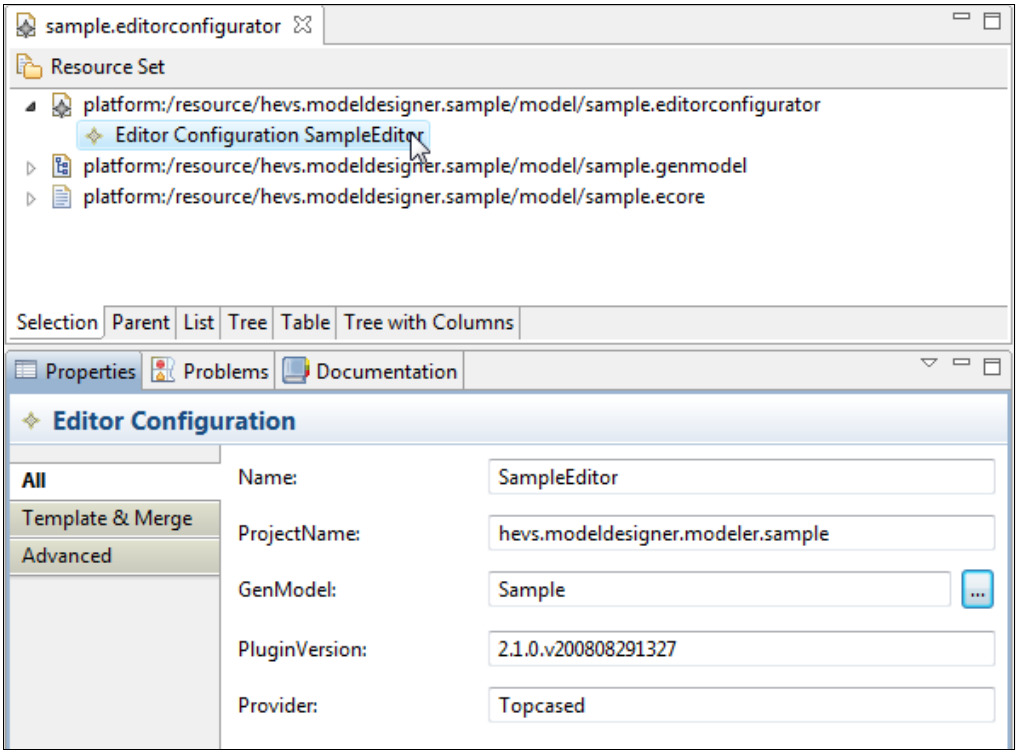
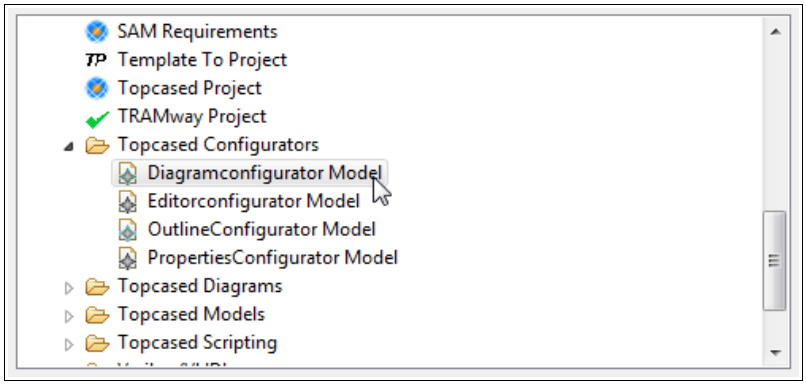
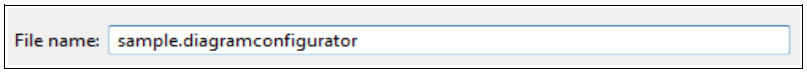
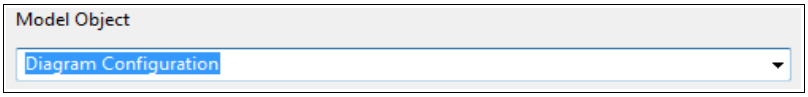


Diagram Configurator

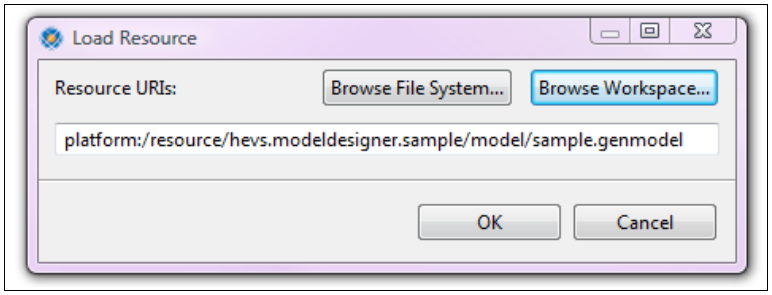
For the diagramconfigurator creation, just do as you did with the editorconfigurator



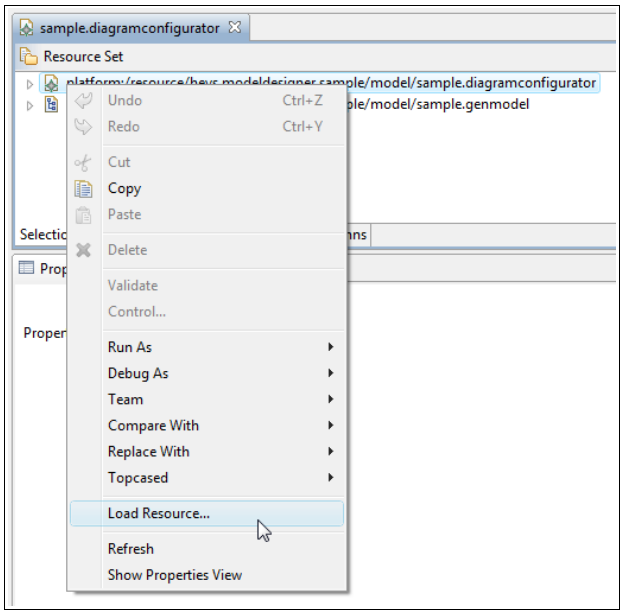


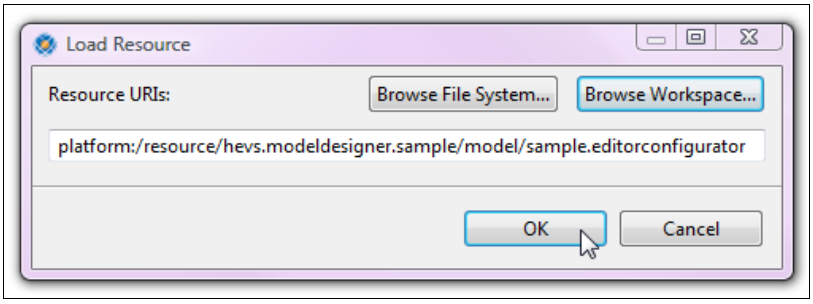


Load the .genmodel...

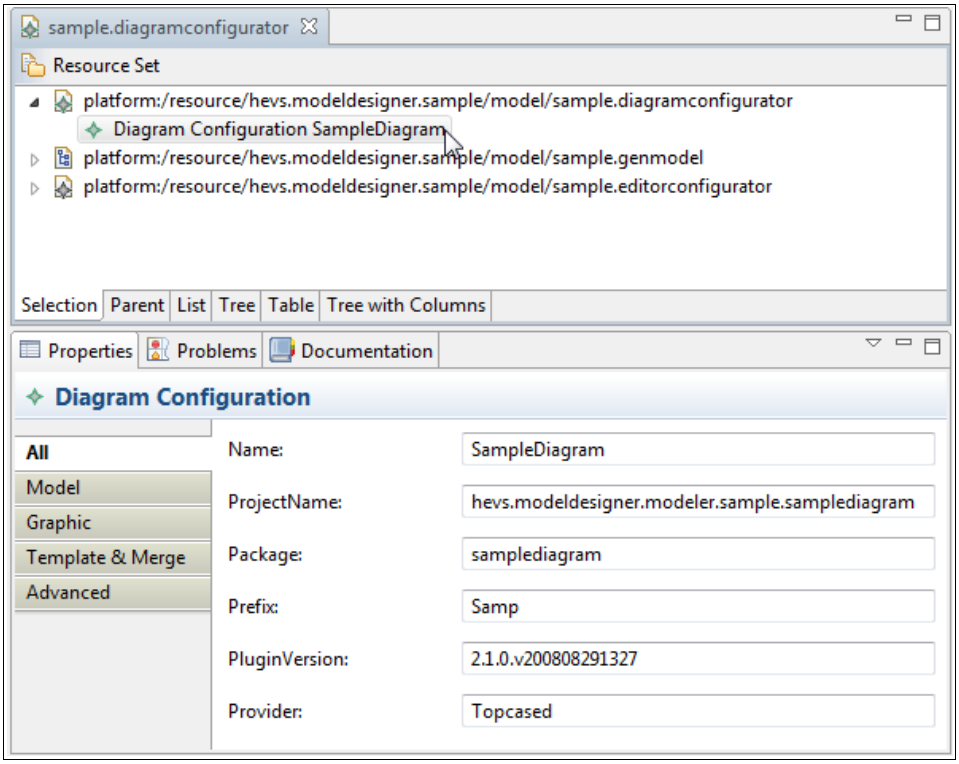


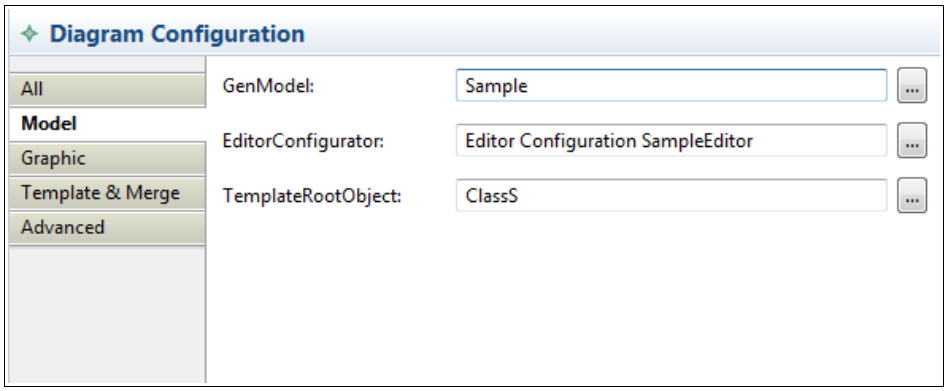
But this time load the .editorconfigurator also.





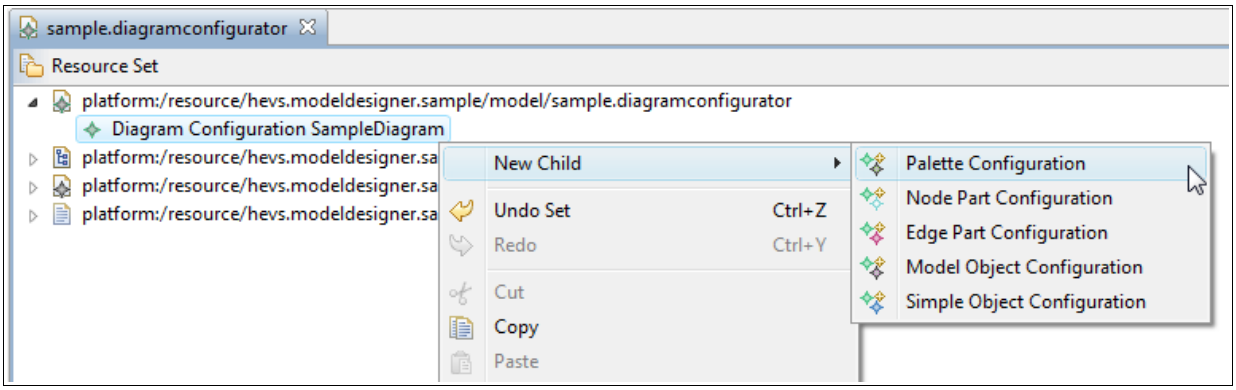
Here is the kind of configuration you should enter :





Graphical Elements Implementation

Now, you can start implementing the contents of your graphical editor.

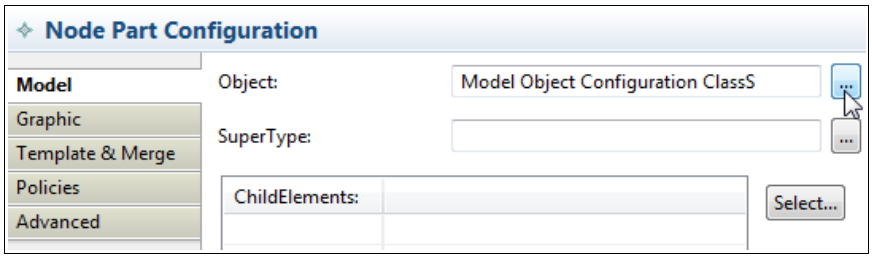


Model Obect

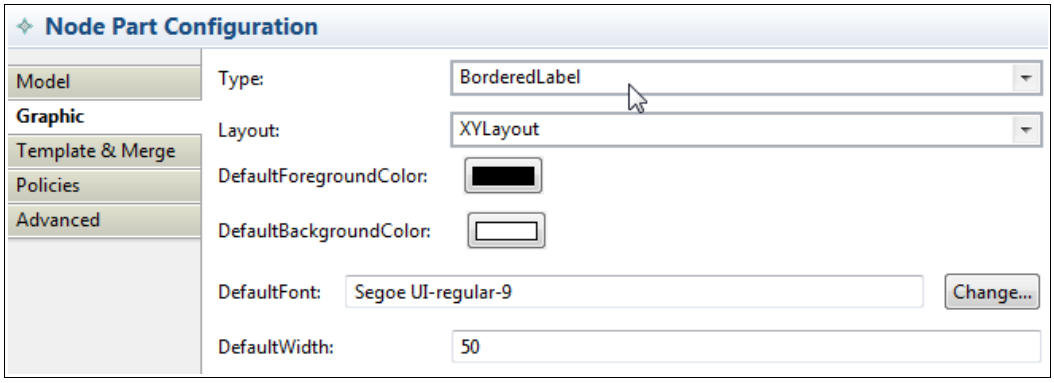
First you must add a Model Object Configuration for each model element that you want interact with the graphical editor, even if that element will not be graphically represented.

Node Part

Then you add a Node Part Configuration for each element you want to be graphically represented. You must specify wich Model Object will be related to the Node.



Going through the left tabs, you can specify the properties of the graphical node like the figure that will represent the node in the graphical editor.

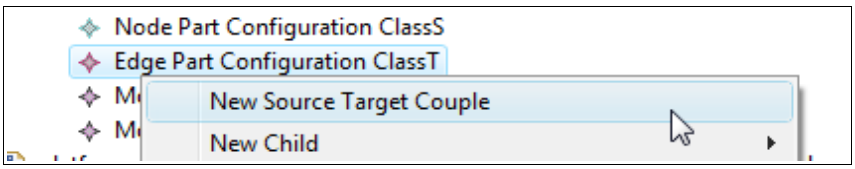


Edge Part

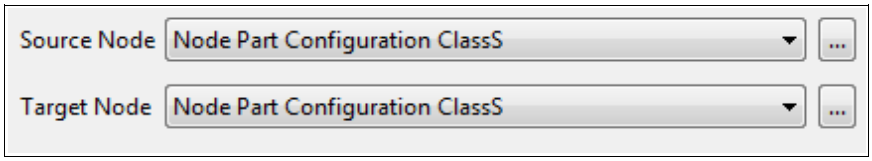
The references (association lines) are made with the Edge Part Configuration. You must also specify what Model Object will be represented by the edge.

Source Target Couple

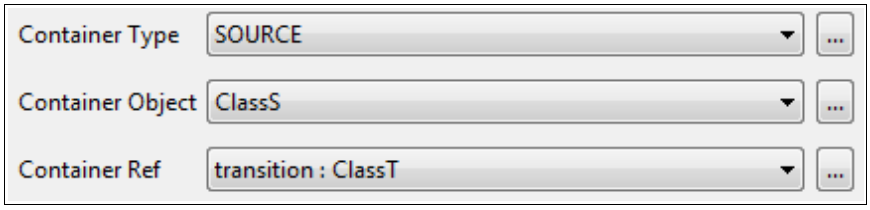
The Edge Part must also contain a Source Target Couple. This property defines wich Node and wich property of that Node will be linked with the Edge.



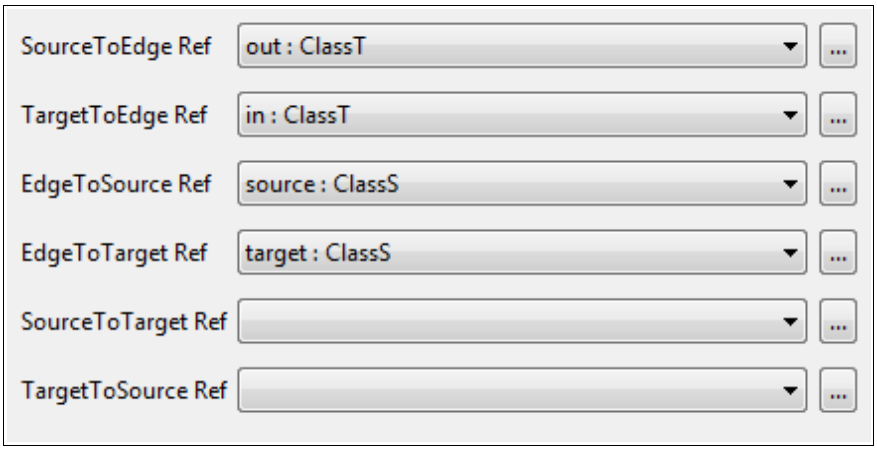
In this example the Source Node is the same as the Target Node, but usually you'll have different Node types.



You must also specify wich property contains the instance of the Edge.

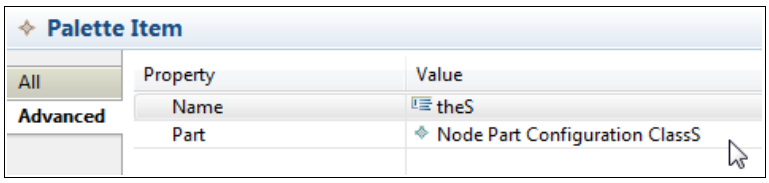


Finally, you choose wich relation will the Edge affect.



The Palettes and their childs

Once you configurated all the graphical objects, you must create a Palette Configuration. This Palette will let you define Categories by adding a child Palette Category. The Palette Categories can contain Palette Items. These Palette Items are used to create graphical parts. Actually, they let you create the graphical Parts that you have defined before.



Now, we have this structure.

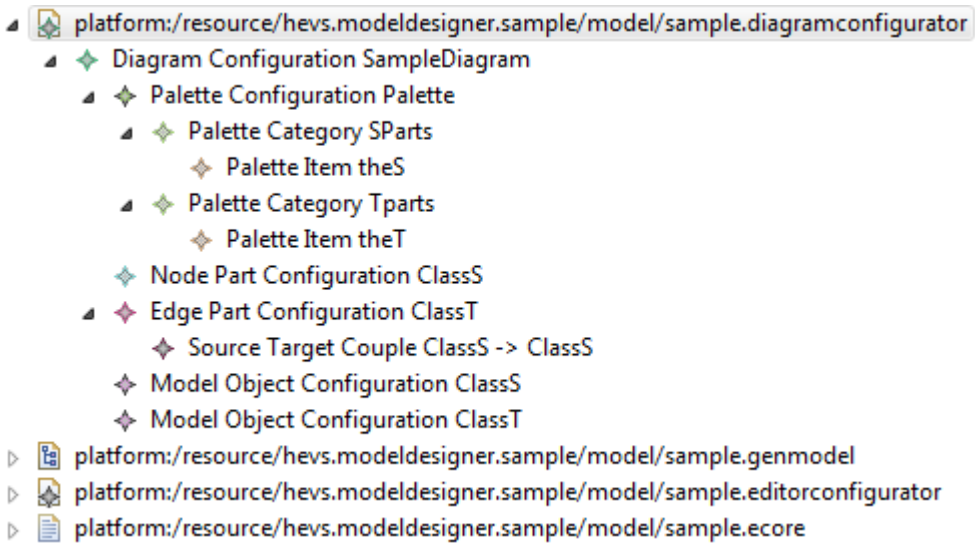
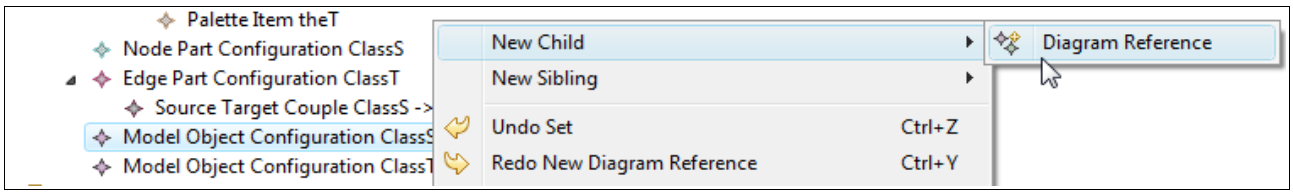
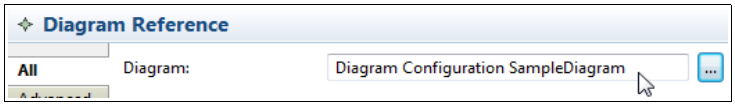


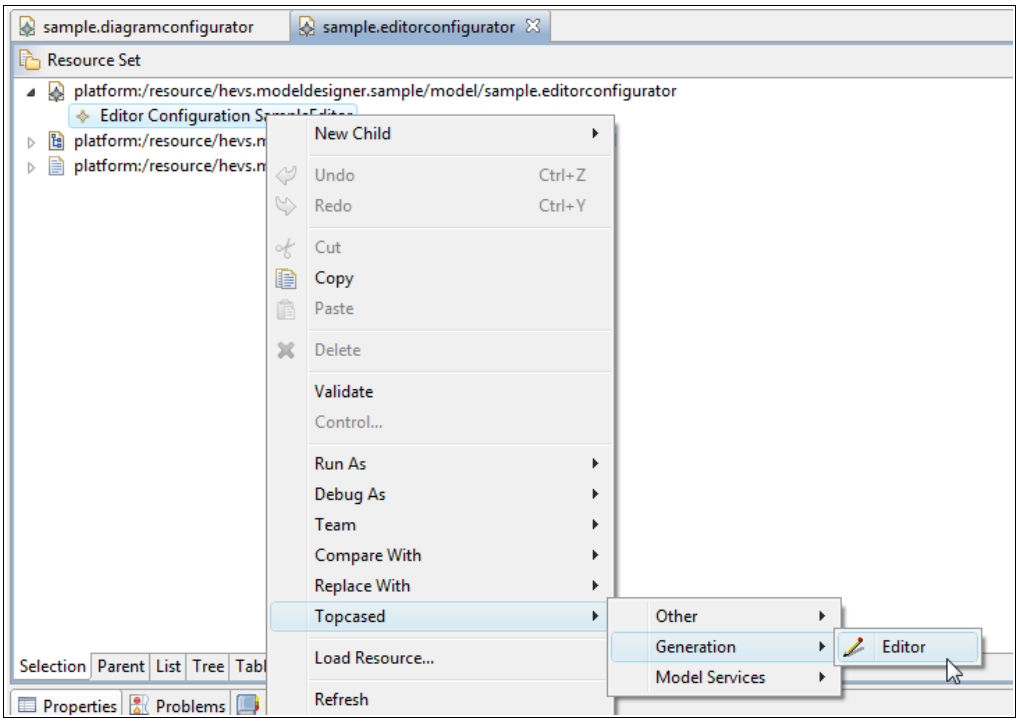
Diagram Reference

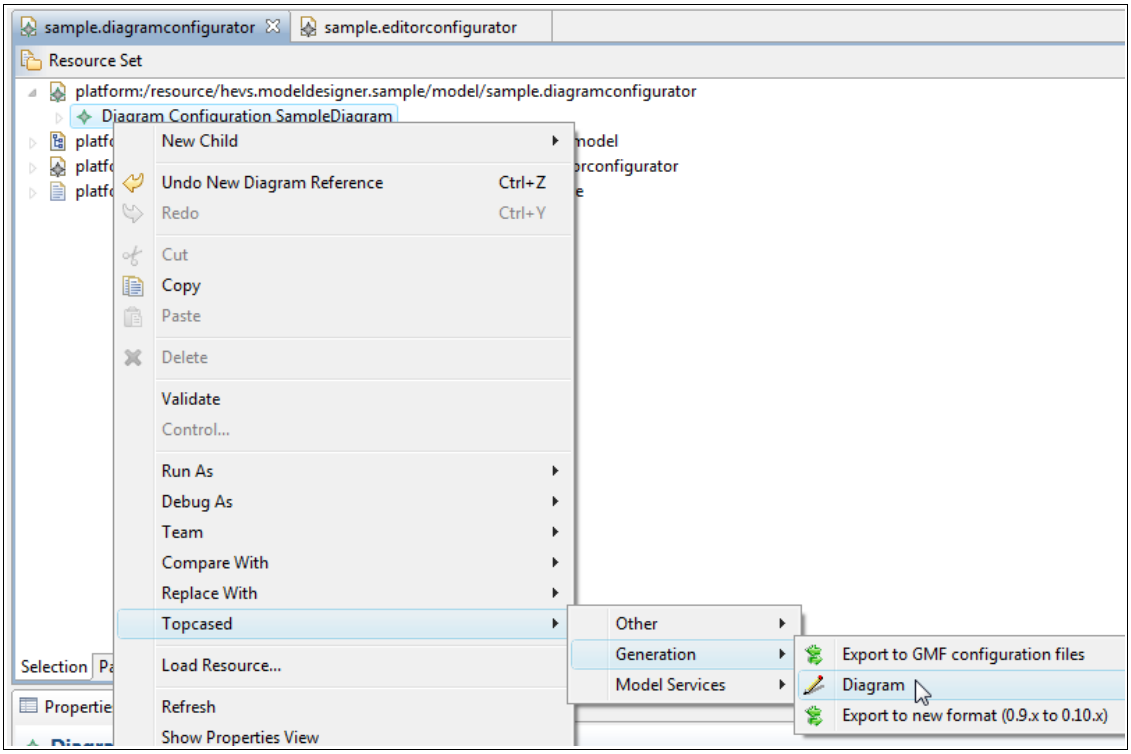
Something is still missing. The Diagram Reference. This part specifies wich Model Object will contain the Diagram.



After adding the Diagram Reference, you'll be able to generate your Graphical Editor.







Don't forget the generation of the editorconfigurator part. before doing the generation of the diagramconfigurator.

You'll surelly get some errors after generating the graphical editor.

Use the help from your IDE to resolve the issues.

Most of time it's a wrong return type, so just cast to get it right.
Check also the the imports.
Once you've got no more errors, you got one last thing to correct.

Go to file :

src/hevs7modeldesigner/modeler/sample/wizards/SampleDiagramsPage.java

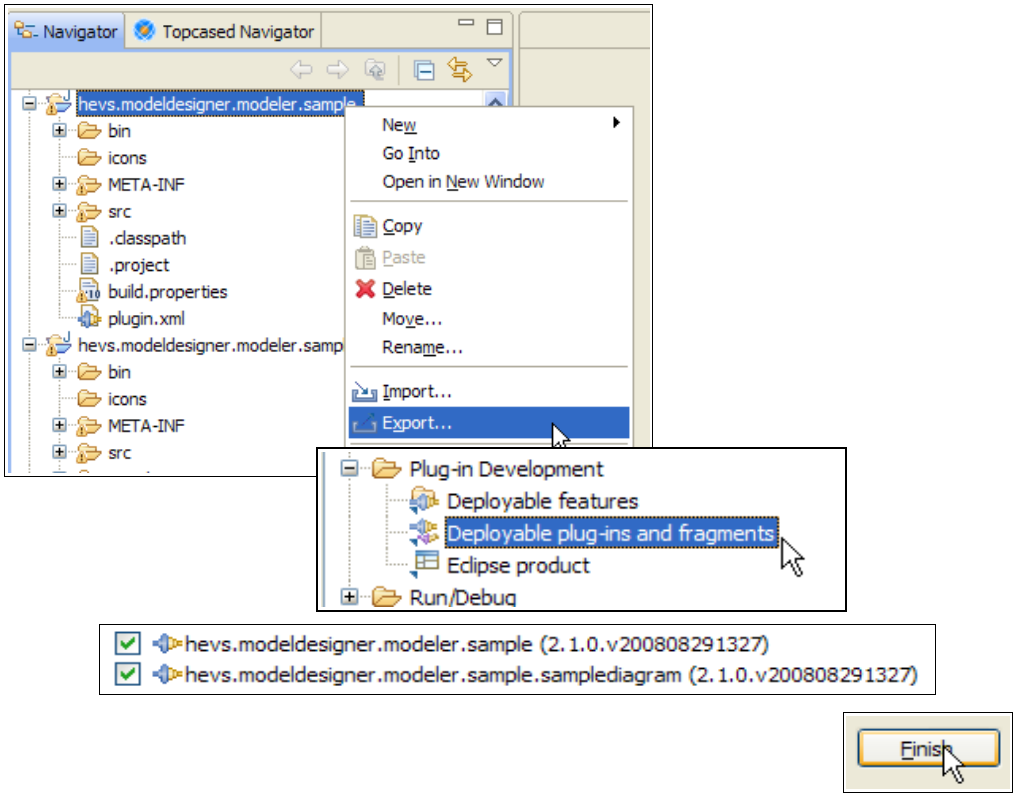
This file is part of the generated code from the editorconfigurator.

At the end of the file search for this function and insert the missing line (the return line).

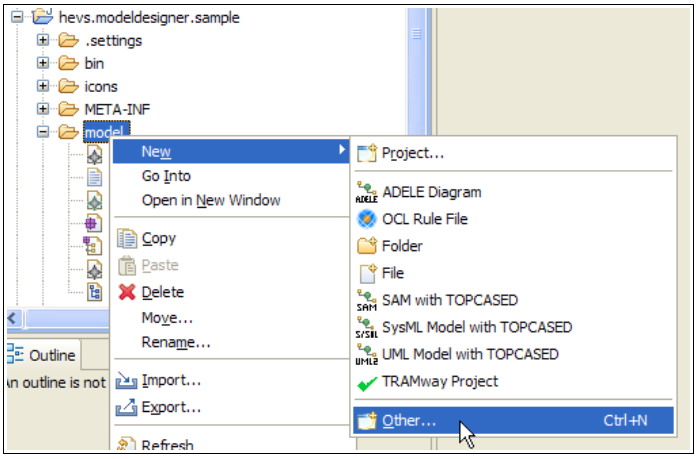
```
/**
 * @see org.topcased.modeler.wizards.DiagramsPage#getDefaultTemplateId()
 * @return String
 * @generated
 */
public String getDefaultTemplateId() {
  // TODO return the corresponding ID of the default template
  return "hevs.modeldesigner.modeler.sample.samplediagram.templates.samplediagram";
}
```

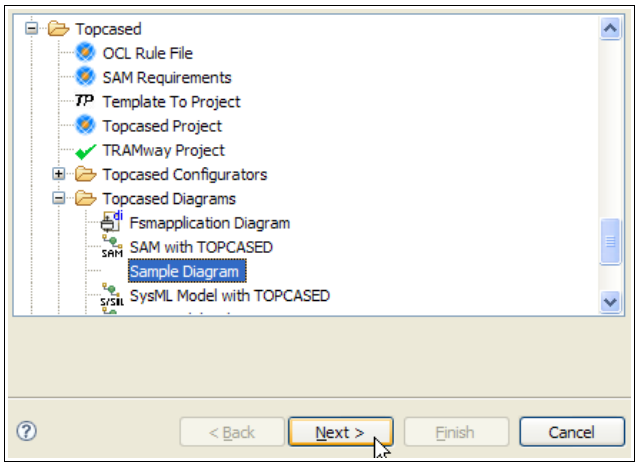
Save your modifications.

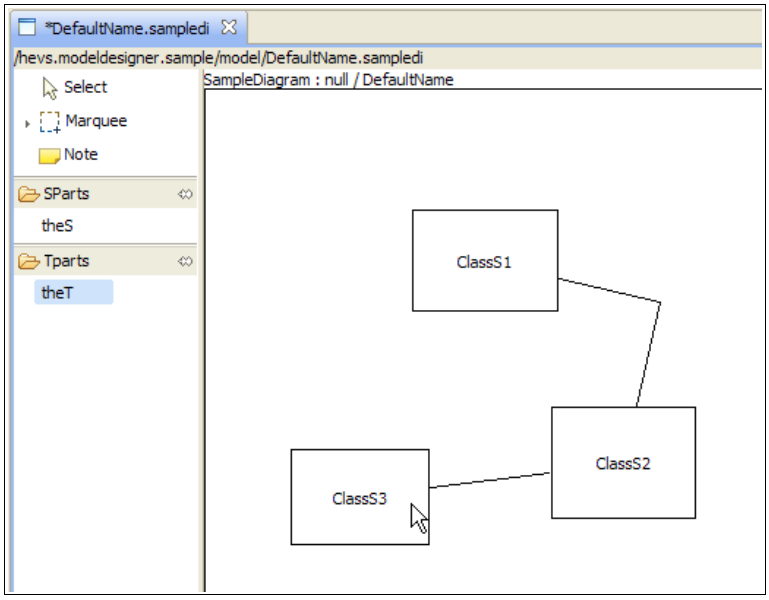
You can now export your plugins.



Once the export finished, you must restart Topcased to be able to use your new Graphical Editor.







END