Projet de fin d'études
DESS ISI

Rapport de projet

Réalisation d'un outil permettant de définir des composants de langage conformes au modèle OFL

Laboratoire I3S
Projet OCL
2000 route des lucioles
Les algorithmes, bâtiment Euclide B
BP 121 – 06903 Sophia Antipolis CEDEX

Auteurs

Olivier Cotto
Roland Pellegrin

Encadreurs

Pierre Crescenzo
Philippe Lahire

Résumé du projet :
L'application qui devra être réalisée se nomme OFL-Meta et constituera un élément d'un nouvel environnement de programmation.
Le but de ce projet est de contribuer à la réalisation d'une interface graphique qui permette à un méta programmeur de créer, modifier ou détruire des OFL-Composants pour constituer un langage. Tous les paramètres et toutes leurs valeurs sont prédéterminés dans la définition du modèle. En pratique, l'application ressemblera à un explorateur Windows qui travaillera sur des chaînes de caractères plutôt que sur des fichiers.

Année 2001/2002

Sommaire

I. Présentation du contexte du projet

OFL (Open flexible Languages) est un modèle méta-objet qui décrit la sémantique opérationnelle, le comportement des langages de programmation à objets les plus courants tels Java, C++, Eiffel, ...

Le principal objectif du modèle OFL est d'offrir au méta-programmeur un moyen simple de créer un nouveau langage ou de modifier le comportement d'un langage existant. En effet, traditionnellement, la méta-programmation est une tâche difficile et fastidieuse. OFL met donc à la disposition du méta-programmeur trois concepts lui permettant de décrire son langage :

- les *concepts-descriptions* ont la capacité de générer des *composants-descriptions* qui figurent les méta-classes du langage,
- les *concepts-relations* permettent de décrire des *composants-relations* qui formeront les méta-relations du langage et
- les *concepts-langages* qui représentent des *composants-langages* (un par langage décrit) dont la principale fonction est de rassembler et composer les composants-descriptions et composants-relations du langage.

Par exemple, le composant-langage Java est formé, entre autres, des composants-descriptions Class, Interface et Array et des composants-relations Extension, Implementation et Aggregation.

Pour créer un OFL-composant, le travail du méta-programmeur est facilité par la présence d'un système de paramètres au sein du modèle. Au lieu d'écrire du code, forcément complexe, il s'agit donc de spécifier un type de comportement. Par exemple, pour créer chaque composant-relation, le méta-programmeur doit répondre aux questions :

- La relation définit-elle une utilisation ou une importation ?
- La relation admet-elle le polymorphisme ?
- Si oui, dans quel(s) sens ?
- Quelle est sa cardinalité maximale ?
- Peut-elle être circulaire, répétée ?
- Possède-t-elle une relation inverse ?

Puis le modèle prend en compte la valeur de chacun des paramètres ainsi instituée et génère le composant correspondant. Le méta-programmeur définit donc, suivant ce principe, tous les composants-descriptions et composants-relations de son langage.

Si son but est plutôt de modifier la sémantique opérationnelle d'un langage existant, il lui faut choisir précisément quel comportement il souhaite adapter et modifier la valeur des paramètres correspondants.

Quatre outils logiciels sont en cours de réalisation pour mettre en oeuvre les principes décrits par OFL au sein d'un environnement de programmation basé sur le modèle. Essentiellement graphiques, ils permettent au méta-programmeur de définir les composants du langage (OFL-Meta, le sujet de ce rapport) puis au programmeur de les utiliser (OFL-ML) pour générer ensuite du code exécutable (OFL-Parser). Ces outils font usage des formalismes UML et XML et d'un système de persistance (OFL-DB).

Le projet que nous présentons ici, est la réalisation de l'outil logiciel OFL-Meta, celui qui va permettre de définir les composants du langage.

II. Cahier des charges

A. Contexte

Afin de donner la possibilité à tout un chacun de pouvoir utiliser le modèle OFL au sein d'un environnement de programmation, le développement de quatre outils logiciels a été lancé dans le cadre des Projets de Fin d'Etude des étudiants 3ème année et DESS de l'ESSI (Ecole Supérieure en Sciences Informatiques). L'outil OFL-Meta fait partie de cette suite logicielle et ce rapport va donc en présenter les différentes spécifications.

B. Prise de connaissance

- Connaissance du sujet
  La première phase du projet est la prise de connaissance avec le modèle OFL.
  Pour cela, une réunion a eu lieu le 18 octobre avec M. Crescenzo, encadreur du projet, afin d'expliquer le thème du projet.

- Définition des objectifs
  Les objectifs ont été définis lors de la première réunion, à savoir réaliser un composant logiciel nommé OFL-Meta. Celui-ci doit permettre de créer, modifier ou supprimer graphiquement des OFL-composants qui sont des instances d'OFL-concepts. En d'autres termes, il offre la possibilité de décrire la sémantique opérationnelle d'un langage qui sera utilisé pour la conception d'une application. Les OFL-composants qu'il manipule peuvent être stockés en utilisant divers formalismes standards comme XML ou MOF.
  La réalisation de l'outil doit être homogène avec les autres composants logiciels de manière à ce que l'ensemble de ces quatre composants puisse ne former par la suite qu'un même outil logiciel.

- Limites du sujet
  A la mi-novembre 2001, les limites du sujet ont été étendues puisque les classes du modèle OFL, déjà réalisées par M. Lahire, devaient être intégrées au projet. Ces classes avaient été conçues dans le but de fonder les premières briques au projet. Elles n'ont finalement pas été retenues car peu adaptées à notre projet.

C. Compréhension du modèle OFL

Le développement de l'outil OFL-Meta nécessite de bien cerner le sujet et ses limites. Dans cette optique et étant donné que la création du modèle OFL a fait l'objet d'une thèse par M. Crescenzo, la compréhension s'est faite en plusieurs étapes :

- Lecture de la partie de la thèse de M. Crescenzo concernant les points jugés importants.
- Rédaction de toutes les questions portant sur des points sombres, qui nécessitent d'être éclaircis
- Débats et questions avec les encadreurs du projet.

D. Choix de développement, implémentation

- **Conception**
  La norme UML étant reconnue et standardisée, nous avons retenu celle-ci pour la partie modélisation de l'application OFL-Meta.
  Les diagrammes qui seront réalisés dans le cadre du projet seront des diagrammes de classes.
  Le support de modélisation est Rational Rose Enterprise Edition V 7.1

- **Langage de programmation**
  D'un commun accord avec les encadreurs du projet, le langage que nous avons choisi comme support de développement de l'application OFL-Meta est Java.
  Ce choix se justifie pour plusieurs raisons :
  -Portabilité : l'application doit pouvoir tourner aussi bien sous MS Windows que sur stations type Unix ou linux
  -Facilité de programmation par rapport à un langage comme C++.
  Le support de développement est Visual Café Enterprise Edition V 4.5

- **Package graphique**
  La librairie graphique que nous avons choisie pour réaliser la partie graphique du logiciel est Swing. Cette librairie est en en effet très répandue, son utilisation est aisée et la documentation qui s'y réfère abonde.

- **Documentation**
  Les classes écrites sont commentées suivant la convention définie dans le Javadoc-tool disponible sur le site de Sun http://java.sun.com/j2se/javadoc/writingdoccomments/index.html

III. Travail effectué

A. Prise de connaissance

La première phase du projet a été la prise de connaissance du projet. Celle-ci s'est principalement faite au cours de la deuxième réunion pendant laquelle M. Crescenzo a présenté le modèle OFL et ses composants : des OFL-concepts (concept langage, concept relation, concept description), des OFL-atomes (langage, relation, description, objet…), des OFL-composants (un-langage, une-relation-généralisation, une-relation-agrégation…) et des OFL-instances.

B. Compréhension du modèle OFL

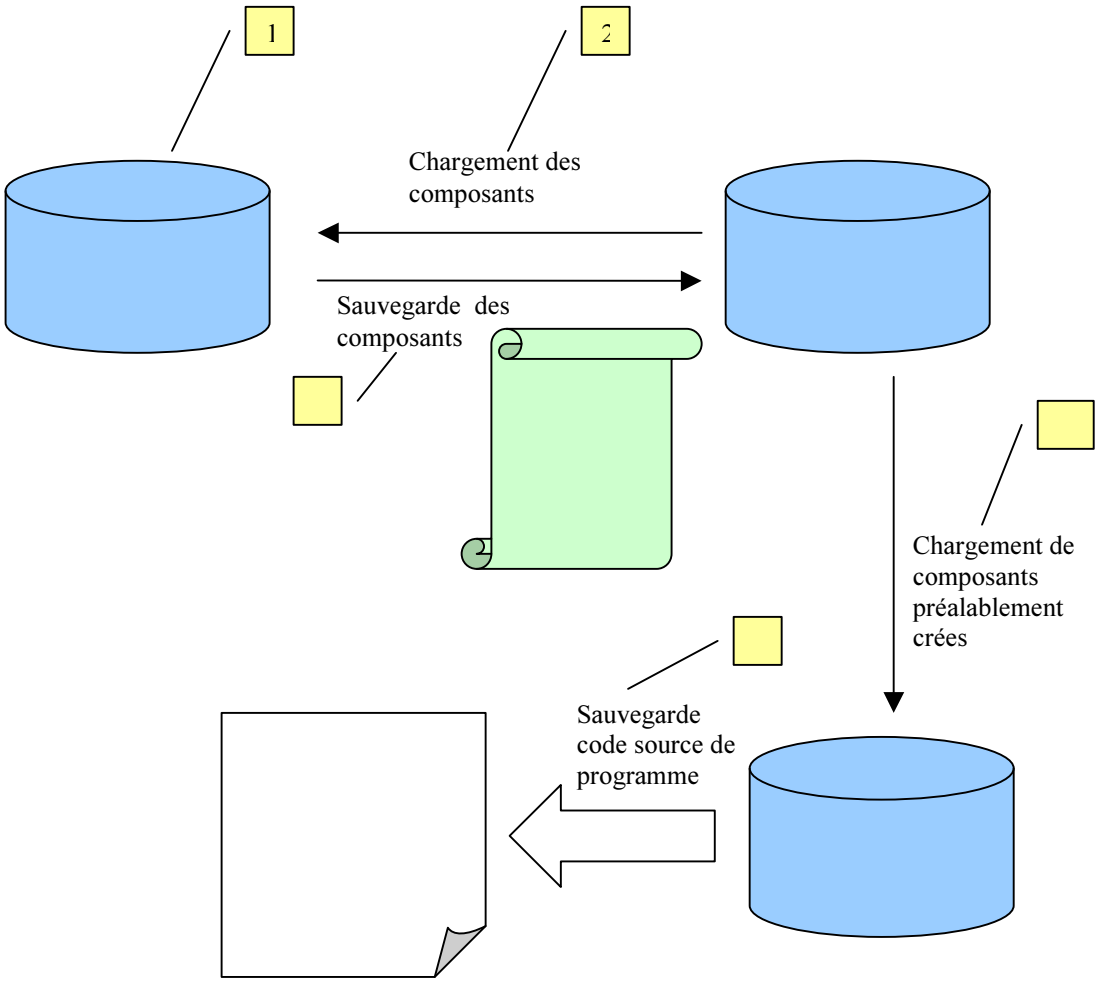
En plus de la deuxième réunion au cours de laquelle M. Crescenzo a expliqué de manière générale le modèle OFL, la compréhension de celui-ci est passée par la lecture d'une partie de la thèse de M. Crescenzo car celle-ci a servi de référence tout au long du projet.
Cette tâche a demandé une grande attention de la part des participants du projet car les concepts sont nouveaux et on se place à un haut niveau d'abstraction.
A la suite de celle-ci, une réunion a été organisée de manière à clarifier tous les points qui le méritaient et voir ensemble comment chacun des trois groupes allait interagir avec les autres.

C. Interaction et coopération entre les trois groupes d'étudiants affectés au projet.

Le projet OFL a été scindé en trois parties, chacune d'elle étant assurée par un groupe :

- OFL-Meta pour la réalisation de l'interface permettant de définir les OFL-composants (langage, relation, description)
- OFL-ML pour la réalisation de l'interface permettant de se servir, de dessiner et de mettre en relation des OFL-composants
- OFL-DB pour la sauvegarde de ces composants.

Le schéma page suivante illustre le rôle du module OFL-Meta au sein de l'environnement de programmation.



OFL-Meta permet de définir des composants (langage, description, relation) en donnant des valeurs à certains paramètres. (1)

Ces composants peuvent ensuite être sauvegardés, au moyen de OFL-DB, dans un fichier au format XML (2) et ils peuvent également être rechargés si l'on souhaite apporter des modifications à un langage précédemment sauvegardé. (3)

Les composants ainsi crées sont ensuite utilisés par le module OFL-ML qui les charge afin de pouvoir offrir au programmeur la possibilité d'utiliser ces composants (définis dans OFL-Meta). (4)

Le logiciel OFL-ML permet ensuite à l'utilisateur de créer une application et de générer son code source (5).

De manière à pouvoir gérer au mieux cette interaction nécessaire aux trois groupes, de nombreuses consultations et mises au point ont été requises, notamment au sujet de la partie OFL-DB qui concerne la sauvegarde des modèles et qui suscitait au départ des interrogations quant au format de sauvegarde.

D. Conception et modélisation UML de l'application OFL-Meta

1. Contexte initial

Avant de commencer à développer l'application OFL-Meta, nous avons réalisé une modélisation de celle-ci selon la norme UML. Cette modélisation a été faite dès le début du projet de manière à ne pas perdre de temps et avoir une première maquette navigable de l'application OFL-Meta. C'est la modélisation qui avait été initialement présentée dans le dossier de spécifications et qui nous a permis d'évoluer à partir d'une première maquette OFL-Meta.

2. Evolution

Conformément a ce qui avait été mentionné dans le dossier de spécifications, nous avons décidé en décembre d'intégrer la structure de données écrite par M. Lahire, afin de prendre en compte toutes les composantes du modèle OFL. Celle-ci se présentait sous la forme de 138 classes. Ces classes ne constituaient en fait que le squelette du modèle car elles étaient seulement compilables et n'avaient jamais été testées.

Cependant, il s'est avéré que nous avons très rapidement bloqué sur cette structure, ceci pour plusieurs raisons :
D'abord le nombre des classes, 138 au total, ce qui se révèle très long à assimiler et intégrer.
Ensuite le plus difficile a été de comprendre le rôle de chacune des classes : la seule documentation sur les classes était la thèse de M. Crescenzo, il était donc extrêmement difficile d'avancer rapidement tout en se plongeant dans la thèse pour trouver l'information.

Suite à cela, nous avons donc discuté au cours de la réunion du 17 janvier 2002 avec nos responsables à propos de nos difficultés à progresser dans l'intégration de ces classes. Apres une mise au point commune, nous avons décidé de repartir ensemble sur une nouvelle structure simplifiée. Le squelette de cette structure a donc été défini au cours de cette réunion.
A la suite de celle-ci, nous l'avons complétée puis renvoyée à nos encadreurs ainsi qu'aux autres participants.
Cette structure a finalement encore été modifiée par M. Lahire et M. Crescenzo et c'est à partir de celle-ci que nous avons ensuite pu évoluer et ajouter les autres briques spécifiques à chacun des 3 projets.

3. Diagrammes de classes réalisés

Les différents diagrammes de classe que nous avons conçu puis fait évoluer au cours du projet, sont présentés ci-après.

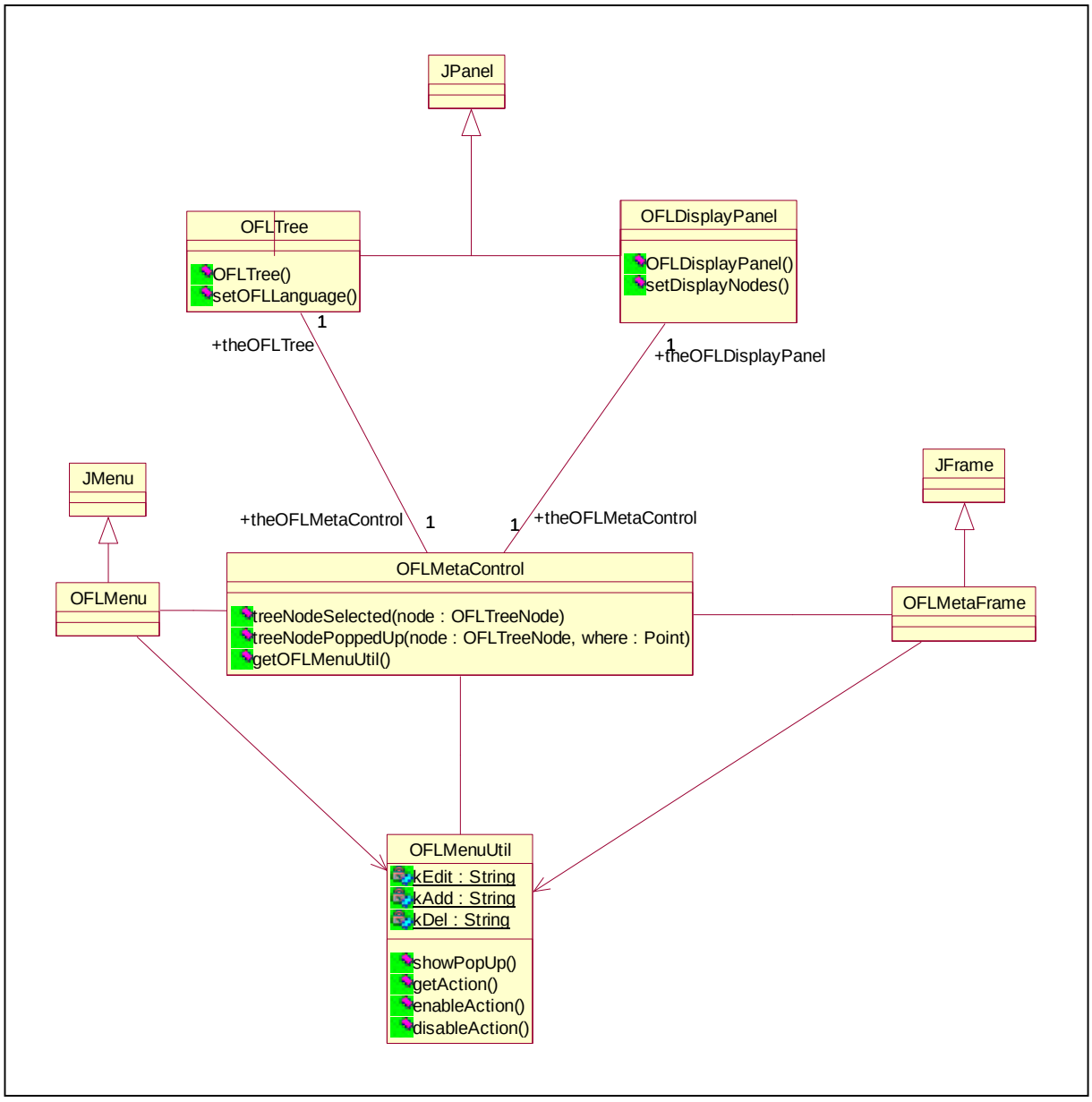


Figure 1 Diagramme de classes général

OFL-MetaControl : Classe principale du programme, tous les évènements de gestion des composants sont centralisés dans cette classe (contient également le « main »).
OFLTree : Classe chargée de la représentation graphique de la structure de l'arbre.
OFLDisplayPanel : Classe chargée de l'affichage dans la partie droite de l'application des valeurs des différents paramètres correspondant au nœud sélectionné.
OFLMenu : Classe de gestion et d'affichage de la barre de menus.
OFLMenuUtil : Classe de services pour les menus popups.
OFL-MetaFrame : Classe dérivant de javax.swing.JFrame, fenêtre de l'application.

Diagramme de classe « OFLTree »

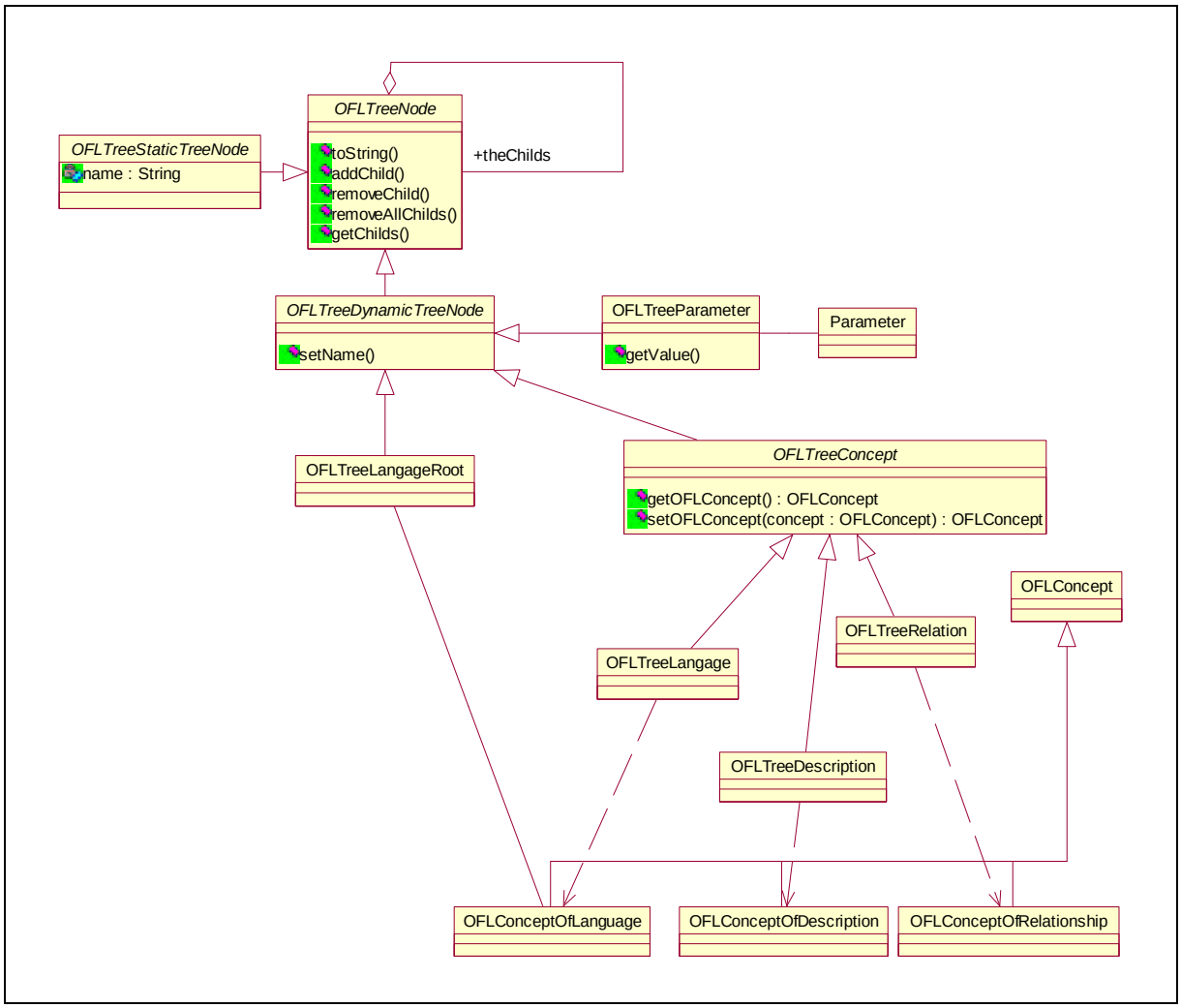


Figure 2 - Diagramme de la structure de données des noeuds de l'arbre de composants

Nous pouvons voir ici que toutes les classes de la structure de données des nœuds stockés dans l'arbre héritent de l'Interface OFLTreeNode.
Cette dernière définit les méthodes de représentation (toString()) ainsi que les méthodes de gestion de hiérarchie d'objets.

On peut également remarquer sur ce schéma le lien entre les nœuds dynamiques de l'arbre (OFLTreeDynamicTreeNode) et les composants de la librairie OFL (OFLConcept …).

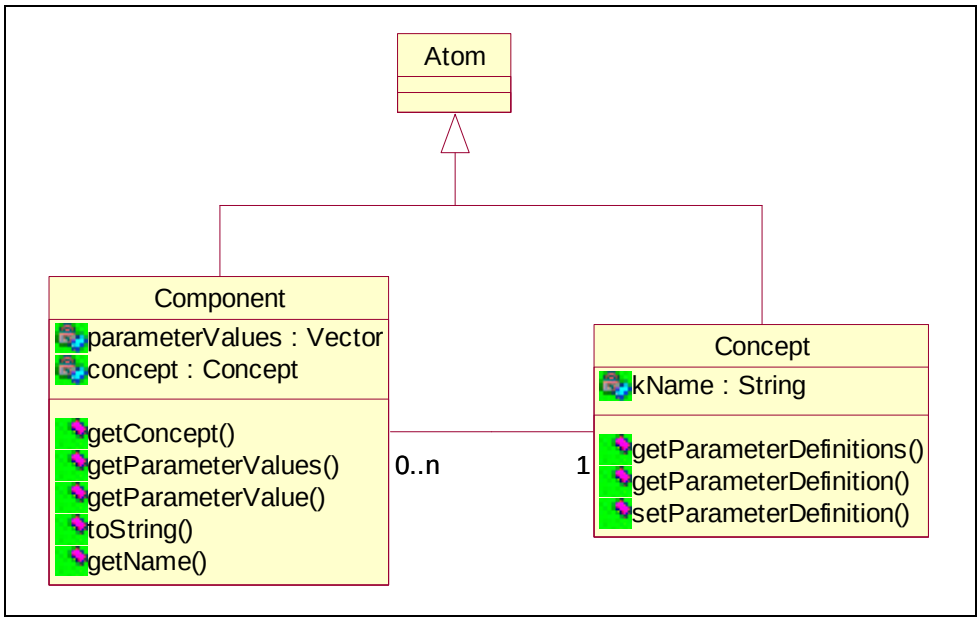


Figure 3 - Diagramme de classes "Atom"

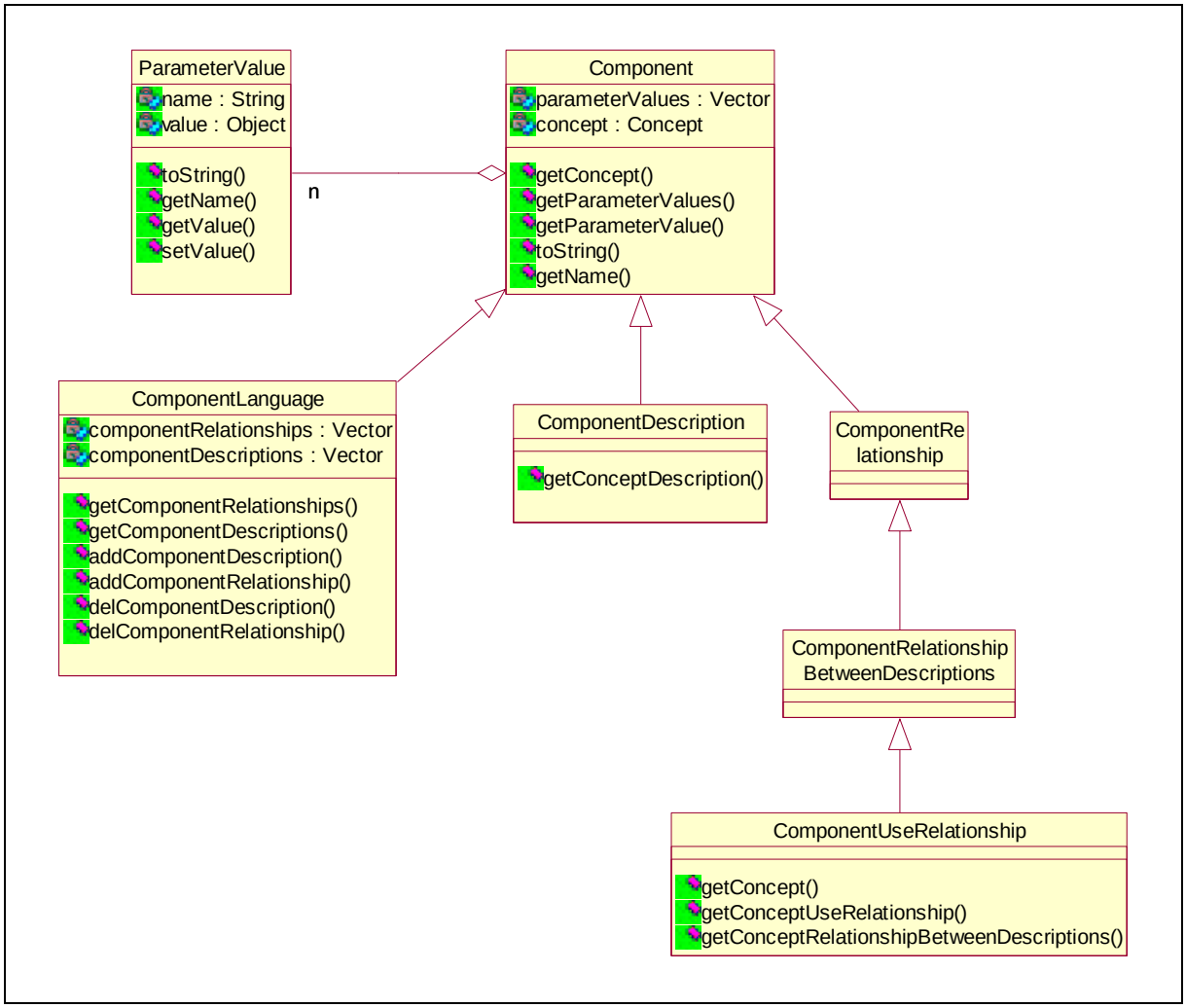


Figure 4 - Diagramme de classes « composants »

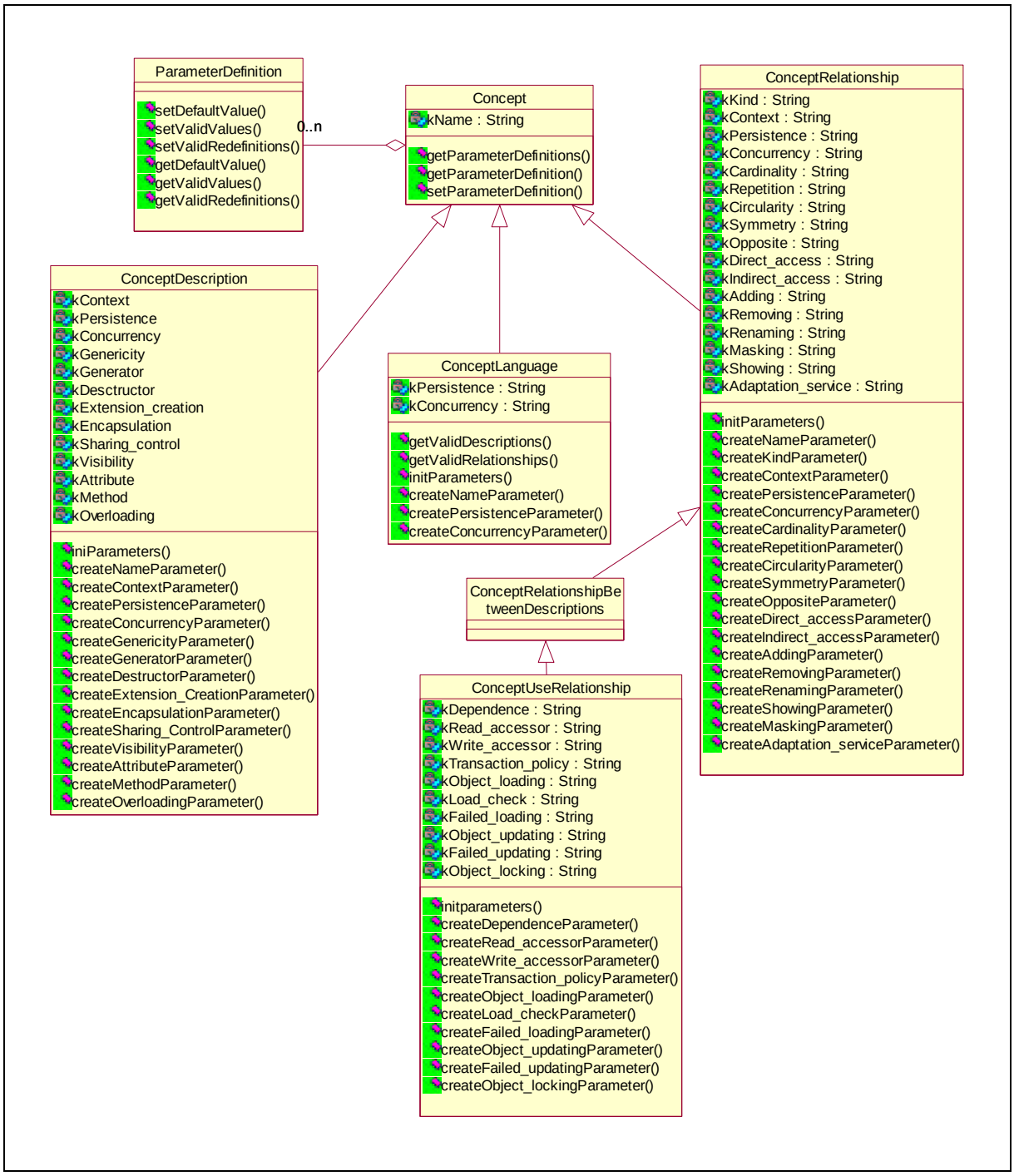


Figure 5 - Diagramme de classe « concepts »

IV. Le logiciel OFL-Meta

A. *Présentation générale*

L'application OFL-Meta a une interface utilisateur similaire à l'éditeur de registre que l'on trouve communément sous Windows, avec cependant plus de fonctionnalités pour l'édition des paramètres.

Elle se décompose ainsi en trois zones :

- La barre de menus en haut de la fenêtre
- L'arborescence des composants sous forme d'arbre, dans la partie gauche de la fenêtre
- Les paramètres et leur valeur sous forme de tableau, dans la partie droite de la fenêtre.

L'application permet de redéfinir le comportement d'un langage en attribuant des valeurs à des paramètres associés aux composants du langage.
Pour cela, le langage, ses composants, ses paramètres et ses redéfinitions sont organisés au sein d'une structure arborescente hiérarchisée. Celle-ci se présente de la façon suivante :

Le premier nœud se nomme « OFL-Meta », c'est la racine de l'arbre. Il possède deux fils

- Un nœud portant le nom du langage crée. Celui-ci possède trois fils correspondant aux trois composants OFL
  - Un noeud « Langage component » qui possède un fils
    - Un nœud portant le nom du langage qui possède deux fils
      - Un nœud « Parameters » qui possède autant de nœuds fils que de paramètres
      - Un nœud « Redefinitions » qui possède deux fils
        - ✓ Un nœud « Descriptions components »
        - ✓ Un nœud « Relationships components »
  - Un nœud « Descriptions components » qui contient autant de nœuds qu'il y a de composants-descriptions. Sous chacun de ces nœuds, on trouve un fils
    - Un nœud « Parameters » qui possède autant de nœuds fils que de paramètres
    - Une nœud « Redéfinitions »
  - Un nœud « Relationships components » qui contient autant de nœuds que de composants-relations. Sous chacun de ces nœuds, on trouve un fils
    - Un nœud « Parameters » qui possède autant de fils que de paramètres
- Un autre nœud représentant les librairies disponibles

Par défaut, au lancement de l'application, un langage nommé « TestLang » est crée. Celui-ci contient déjà les descriptions « Class » et « Interface » et les relations « Agregation » et « association ».
Ces descriptions et relations peuvent évidemment être modifiées ou supprimées et l'utilisateur peut également en créer d'autres.

La capture d'écran page suivante illustre les explications données précédemment.

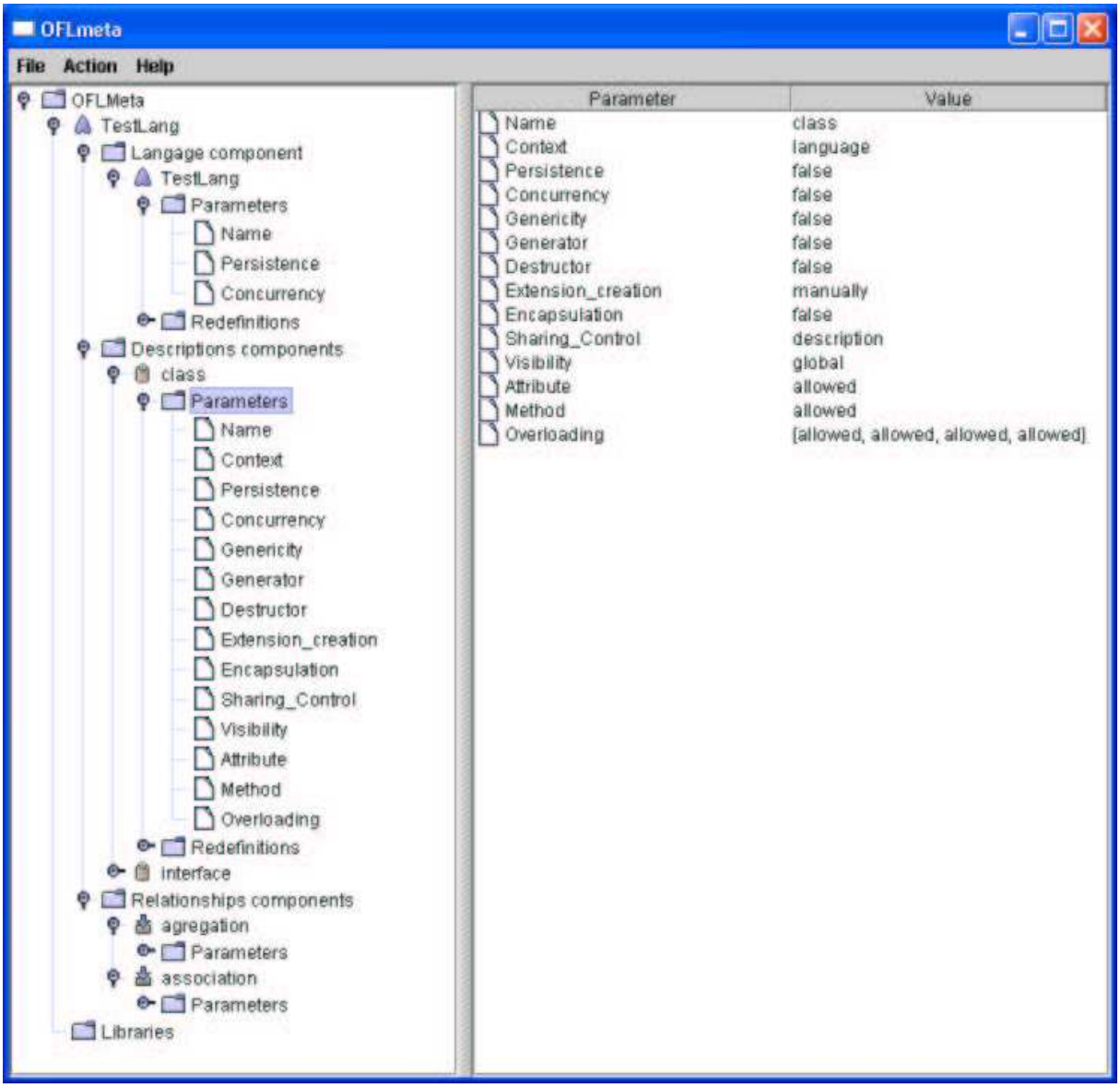
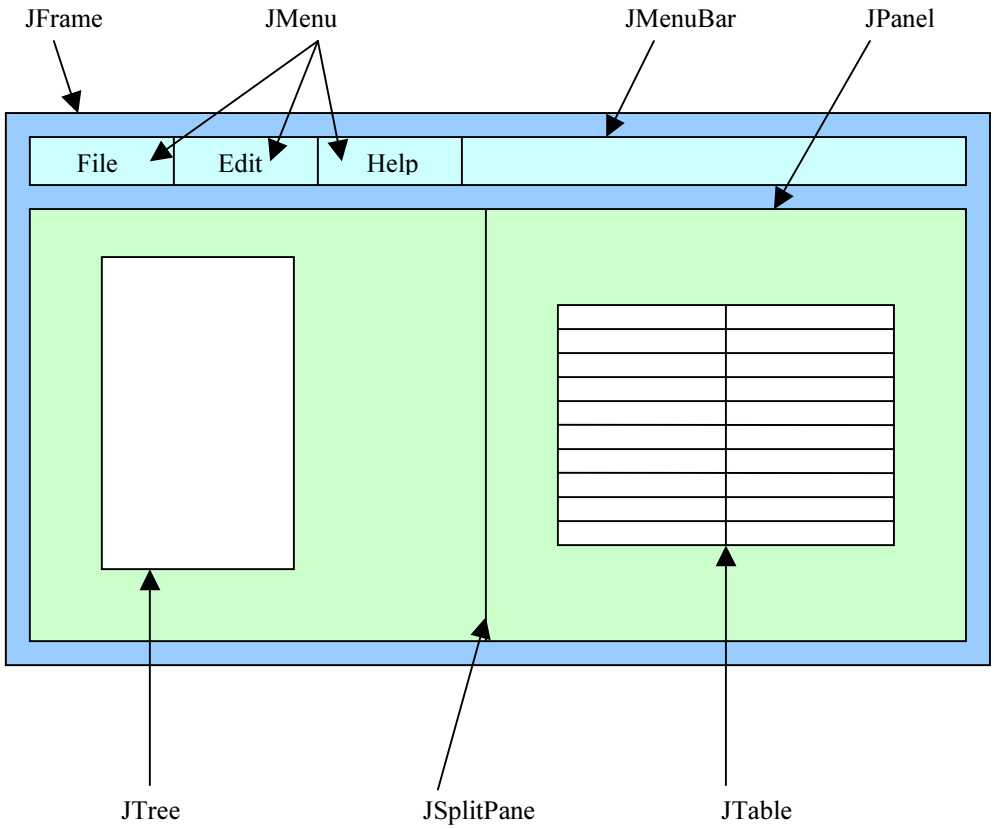


Figure 6 Vue globale de l'application

On peut schématiser les choix d'implémentation généraux de l'application de la façon suivante :



La fenêtre principale du logiciel OFL-Meta est une **JFrame**. Celle-ci joue le rôle de « container » et accueille donc d'autres composants JavaSwing.
Elle est ainsi composée de deux entités : un **JMenuBar** et un **JPanel**.

- Le composant **JMenuBar** est lui-même composé de trois **JMenu** : « File », « Edit » et « Help ». Ces trois **JMenu** étant composés eux-mêmes de plusieurs sous menus de type **JMenuItem**
  « File » → « New », « Load », « Save » et « Import »
  « Edit » → « Edit », « Add » et « Deletete »
  « Help » → « About »
- Le composant **JPanel** est lui-même composé d'un **JSplitPane** à son tour composé d'un **JTree** et d'un **Jtable**.

B. Fonctionnalités

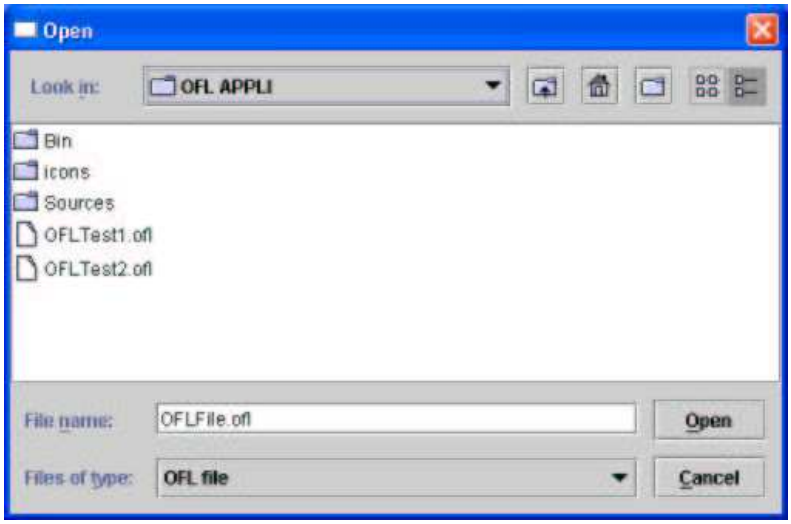
Menu « File »

Sous-menu « New »

Permet la création d'un nouveau langage. A terme, l'utilisateur se verra proposé un « wizard ». Il pourra, par une suite de menus, donner des valeurs à des paramètres qui lui seront présentés et l'application générera ensuite le langage nouvellement défini.

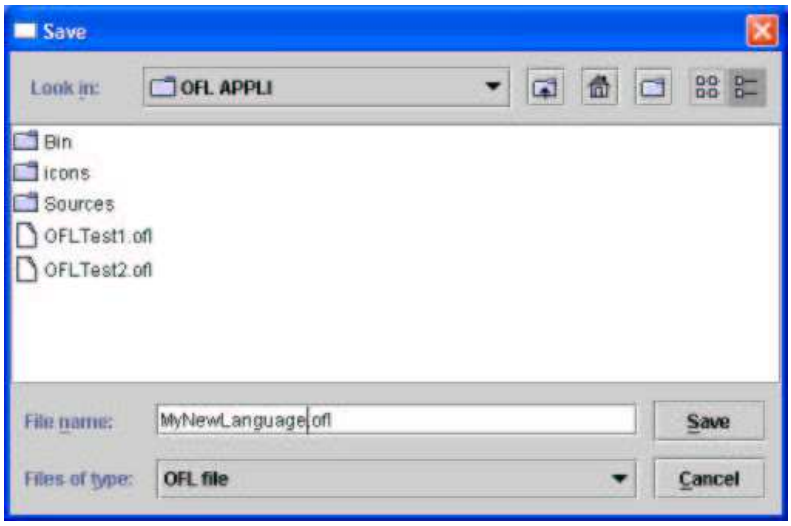
Sous-menu « Load »

Permet de charger un langage à partir d'un fichier au format XML.
Cette fonctionnalité est réalisée au moyen d'une fenêtre de type « JFileChooser » qui permet à l'utilisateur de naviguer dans l'arborescence et de choisir le fichier au format XML correspondant au langage qu'il souhaite charger.



Sous-menu « Save »

Permet la sauvegarde d'un langage sous forme de fichier XML.
Cette fonctionnalité est réalisée au moyen d'une fenêtre de type « JFileChooser » qui permet à l'utilisateur de naviguer dans l'arborescence et de nommer le fichier au format XML correspondant au langage qu'il souhaite sauvegarder.



Sous-menu « Import »

Permet d'importer un langage dans une librairie.

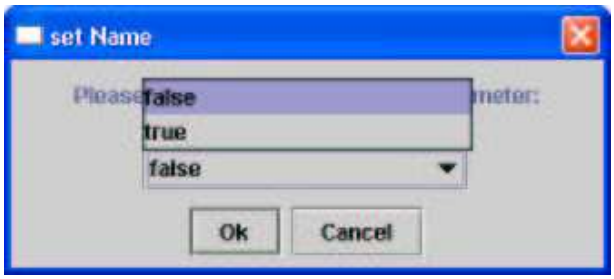
Menu « Edit »

Sous-menu « Edit »

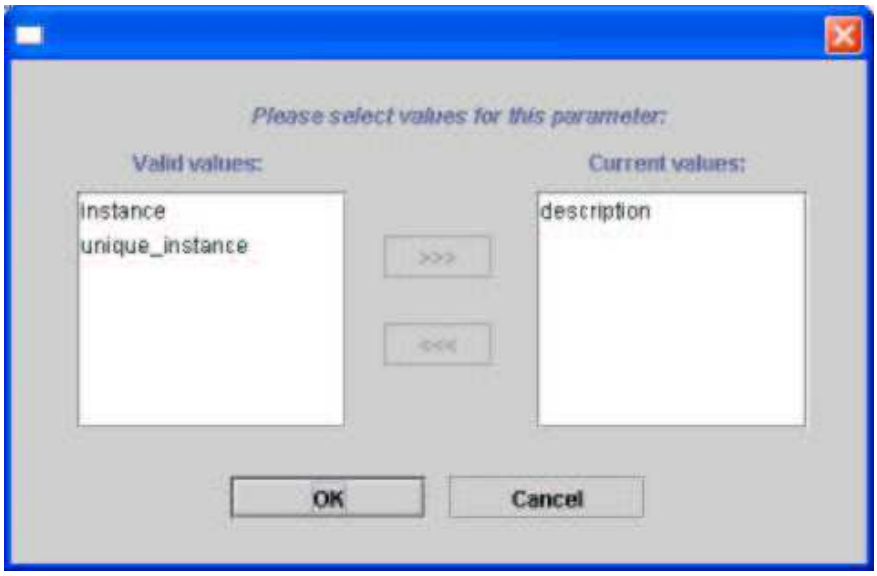
Permet suivant le contexte d'éditer un paramètre, c'est-à-dire lui donner une valeur ou une suite de valeurs, ou bien de renommer un langage, une relation ou une description.
Plusieurs cas se présentent suivant le type du paramètre à modifier

Renommer :

Sélectionner une valeur parmi une liste de valeurs possibles :



Sélectionner plusieurs valeurs parmi un ensemble de valeurs possibles :



Sous-menu « Add »

Permet d'ajouter, suivant le contexte, un concept description ou un concept relation au langage courant

Sous-menu « Delete »

Permet de supprimer, suivant le contexte, un concept description ou un concept relation au langage courant

Menu « Help »

Sous-menu « About »

Donne des informations sur la version du logiciel et les auteurs

V. Planning passé

ID	Tâche	Durée	Début	Fin
1	Premier entretien avec M. Crescenzo	1 jour	18/10/2001	18/10/2001
2	Prise de connaissance avec le sujet	10 jours	18/10/2001	31/10/2001
3	Réunion: exposé de M. Crescenzo sur le modèle OFL	1 jour	24/10/2001	24/10/2001
4	Réalisation du diagramme de classes	14 jours	25/10/2001	13/11/2001
5	Développement de la première maquette	14 jours	25/10/2001	13/11/2001
6	Réunion: présentation de la première maquette	1 jour	14/11/2001	14/11/2001
7	Développement de la deuxième maquette	10 jours	15/11/2001	28/11/2001
8	Réunion: présentation de la deuxième maquette	1 jour	28/11/2001	28/11/2001
9	Compréhension et déboguage des classes du modèle OFL	4 jours	29/11/2001	04/12/2001
10	Réunion: éclaircissement sur le diagramme de classes à intégrer au projet	1 jour	05/12/2001	05/12/2001
11	Réalisation du dossier de spécifications	7 jours	06/12/2001	14/12/2001
12	Examens	4 jours	17/12/2001	20/12/2001
13	Vacances de Noêl	12 jours	21/12/2001	07/01/2002
14	Déboguage et tests des classes du modèle	11 jours	08/01/2002	17/01/2002
15	Réunion	1 jour	17/01/2002	17/01/2002
	Réalisation du nouveau diagramme de classes, début de refonte de l'application	19 jours	18/01/2002	06/02/2002
16	Réunion	1 jour	07/02/2002	07/02/2002
17	Développement d'une nouvelle version d'OFL-Meta basée sur la nouvelle structure de données	14 jours	08/02/2002	21/02/2002
18	Réunion	1 jour	22/02/2002	22/02/2002
19	Développement OFL-Meta : intégration des nouvelles fonctionnalités (icones, menus contextuels, redefinitions)	20 jours	23/02/2002	14/03/2002
20	Réunion	1 jour	15/03/2002	15/03/2002
	Fusion partielle des projets OFL-ML et OFL-Meta, intégration du « save » permettant de chargeant et sauvegarder des composants du langage	2 jours	16/03/2002	17/03/2002
22	Examens	14 jours	18/03/2002	29/03/2002
	Fusion finale des projets OFL-ML et OFL-Meta, le « save » et le « load » fonctionnent, correction de bugs, rédaction du rapport final.	10 jours	30/03/2002	03/04/2002
	Réunion	1 jour	04/04/2002	04/04/2002
23	Réunion de bilan et de préparation à la soutenance	1 jour	09/04/2002	09/04/2002
	Préparation à la soutenance finale	1 jour	10/04/2002	10/04/2002
24	Soutenance finale	1 jour	11/04/2002	11/04/2002

Afin d'avoir un suivi très précis de l'avancement de notre projet, nous avons convenu avec nos coordinateurs M. Crescenzo et M. Lahire que des réunions auraient lieu en moyenne tous les 15 jours/ 3 semaines maximum.
Depuis le début du projet, nous avons donc effectué quantité d'entretiens afin d'avoir un bon suivi de celui-ci et d'être toujours certains de suivre la bonne voie.
Voici donc le détail de ces réunions :

- 18 octobre 2001 : premier entretien avec M. Crescenzo au laboratoire I3S.

- 24 octobre 2001: Réunion de prise de connaissance du sujet du projet.

- 07 novembre 2001: Présentation d'une première interface composée des principales fonctionnalités de l'application mais non encore implémentées. Celle-ci ne gère donc pas encore le modèle OFL.

- 21 novembre 2001 : Présentation d'une interface gérant la hiérarchie des classes OFL tel que réalisées lors de la phase initiale de modélisation.
Au cours de cet entretien, les encadreurs ont jugé que l'application prenait forme, et ils ont donc exprimé le souhait que soit maintenant prises en compte dans l'application les classes déjà réalisées du modèle OFL.

- 05 décembre 2001 : Précision sur le sujet quant aux classes du modèle OFL. Mise au point sur la charge de travail demandée étant donnée que le diagramme de classes est conséquent, 138 classes à assimiler, tester et débuguer.

- 17 janvier 2002 : Explications, éclaircissement et définition des objectifs par rapport au nouveau diagramme de classes qui doit être fait. Décision d'abandonner l'intégration des 138 classes et mise au point d'un nouvelle structure plus adaptée à notre projet.

- 07 février 2002 : Validation du nouveau diagramme de classes

- 22 février 2002 : Validation, mise au point, ajout de nouvelles fonctionnalités à l'application

- 15 mars 2002 : Présentation d'une nouvelle version de l'application. En attente que les « load » et « save » fonctionnent correctement (bugs relevés).

- 04 avril 2002 : Présentation de la première version exploitable d'OFL-Meta : les « load » et les « save » fonctionnent.

- 09 avril 2002 : Réunion de bilan et de préparation à la soutenance : correction du rapport du Projet de Fin d'Etudes, correction des transparents powerpoint de la présentation du projet.

- 11 avril 2002 : Soutenance finale.

VI. Conclusion personnelle

Ayant déjà eu plusieurs expériences professionnelles, nous avons choisi ce projet car nous souhaitions élargir notre expérience au domaine de la recherche.

Ce projet constitue en effet pour nous notre première expérience dans le monde de la recherche et il nous donne un aperçu de cet environnement même si la réalisation de celui-ci ne se fait pas au sein du laboratoire I3S mais dans les locaux de l'ESSI.
Nous avons ainsi découvert les différences qui existent entre le monde professionnel et le monde de la recherche, notamment le fait que les objectifs évoluent au long du processus de développement.
Contrairement à ce qu'il en aurait été pour un projet en entreprise pour des raisons de coût/délai, nous avons en effet pu remodeler notre application a plusieurs reprises car le modèle OFL avait changé suite à une réunion avec nos encadreurs.

Bien que ce projet ne s'inscrive pas dans un contexte professionnel, nous avons eut une approche client-prestataire avec nos encadreurs et les étudiants des autres projets.
Nous avons en effet organisé des réunions d'avancement régulières pendant lesquelles nous avons abordé l'avancement de notre projet, les problèmes rencontrés, les moyens de les résoudre, mais aussi les dates prévisionnelles de fusion des différents projets, ainsi que la définition précise des interfaces de communication entre ces projets.
La communication avec les autres acteurs des projets OFL a été un point très important pour nous, car il nous a permis de progresser sans cesse mais également de faire progresser les autres en apportant nos idées, et nous pensons que la communication inter-collaborateurs sera l'un des points clés qui assurera notre réussite dans un milieu professionnel.

La partie conception de ce projet nous a également tenu a cœur car c'est la première fois que nous avons commencé un projet par une modélisation sous Rational Rose. Nous avons tout d'abord commencé par traduire les besoins de nos encadreurs par une modélisation UML que nous avons ensuite suivie pour le développement en Java.
De ce fait, le travail en binôme nous a été facilité car le modèle UML nous a servi de référence.
Néanmoins, les objectifs du projet ayant beaucoup évolué au cours des deux derniers mois, nous avons quelque peu délaissé ce modèle pour pouvoir respecter nos contraintes de temps. L'idéal serait aujourd'hui de mettre à jour ce modèle pour continuer à le faire évoluer sainement.

Enfin, cette expérience dans le domaine de la recherche a été pour nous une expérience concluante car nous étions plus libre dans nos choix de conception et d'implémentation, et nous avons également pu améliorer nos connaissances du langage Java et de sa librairie graphique Swings.

Pour finir, nous espérons que la réalisation de ce projet contribuera à accélerer la mise à disposition de ce modèle au public.

VII. Bibliographie

La thèse de M. Pierre Crescenzo qui nous a servi de référence pour la compréhension de l'approche OFL.
http://www.crescenzo.nom.fr/publications/these-2001-12.pdf

Le site Web de M. Crescenzo, dans lequel on peut trouver des informations relatives à l'hyper-généricité
http://www.crescenzo.nom.fr/

Le site Web de M. Lahire qui contient également de nombreuses publications sur l'hyper-généricité.
http://www-iutinfo.unice.fr/~lahire/

Le site Web de L'équipe qui a bien voulu nous accueillir
http://www.i3s.unice.fr/ocl/

Un tutorial très intéressant pour démarrer et faire évoluer une application en Swing
http://java.sun.com/docs/books/tutorial/uiswing/index.html

Page à laquelle nous nous sommes sommes référés pour l'écriture de la documentation des classes.
http://java.sun.com/j2se/javadoc/writingdoccomments/index.html

Enfin notre site de référence pour la programmation Java (API disponibles)
http://java.sun.com/products/jdk/1.2/docs/api/index.html

Quelques ouvrages que nous avons également trouvé très utiles :

Titre : **The JFC Swing Tutorial: A Guide to Constructing GUIs**
Auteurs: Kathy Walrath, Mary Campione
Editeur:
Année parution: 1999

Titre : **Modélisation Objet avec UML**
Auteurs: Pierre-Alain Muller, Nathalie Gaertner
Editeur: Eyrolles
Année parution: 2001

Titre : **Construire une étude cas XML**
Auteurs: Jean-Christophe Bernadac, François Knab
Editeur: Eyrolles
Année parution: 1999