

Rapport de stage

Interface Utilisateur pour FlowVR

par Jérémie THOMAS

Stage de dernière année — RICM — Polytech'Grenoble

Du 16/05/2005 au 16/09/2005

Effectué à :

L'INRIA Rhône-Alpes
655, Avenue de l'Europe
Inovallée - Monbonnot
38334 Saint Ismier Cedex

Sous la direction scientifique de :

Jérémie ALLARD
Bruno RAFFIN
Nicolas TURRO

Table des matières

1 Présentation de la structure d'accueil et du sujet **5**
- 1.1 Présentation de l'INRIA . . . 5
  - 1.1.1 En bref . . . 5
  - 1.1.2 L'INRIA en quelques chiffres . . . 5
  - 1.1.3 L'INRIA Rhône-Alpes . . . 5
- 1.2 Le service SED . . . 6
- 1.3 Présentation du sujet et des objectifs . . . 6

2 Contexte technique **7**
- 2.1 FlowVR : principes . . . 7
- 2.2 Exemple d'application FlowVR : GrImage . . . 7
- 2.3 FlowVR : développement d'une application FlowVR . . . 8
  - 2.3.1 Étape 1 : description des modules à utiliser . . . 9
  - 2.3.2 Étape 2 : Choix des modules et de leur disposition : le fichier MML . . . 9
  - 2.3.3 Étape 3 : Création du réseau : le script perl . . . 10
  - 2.3.4 Étape 4 : Création du réseau : le fichier net . . . 10
  - 2.3.5 Étape 4 : Lancement de l'application . . . 10
- 2.4 Technologies et outils utilisés . . . 10

3 Plan de travail **13**

4 Réalisations éffectuées **15**
- 4.1 Présentation de FlowVR GUI . . . 15
  - 4.1.1 La fenêtre principale . . . 15
  - 4.1.2 L'éditeur de fichiers de description . . . 16
  - 4.1.3 L'éditeur de fichiers MML . . . 17
  - 4.1.4 L'éditeur Perl . . . 18
  - 4.1.5 Exécution d'une application FlowVR . . . 18
- 4.2 Architecture . . . 19
- 4.3 Organisation du travail et difficultés rencontrées . . . 21
  - 4.3.1 organisation du travail et orientation . . . 21
  - 4.3.2 Probèmes techniques rencontrés . . . 21

5 Bilan et perspectives **23**

Chapitre 1

Présentation de la structure d'accueil et du sujet

J'ai éffectué mon stage à l'INRIA Rhône-Alpes pour une durée de 4 mois dans le service Support, Expérimentations et Développement logiciel (SED). Le but du stage était de réaliser une interface simplifiant les étapes nécessaires à la réalisation d'applications distribuées.

1.1 Présentation de l'INRIA

1.1.1 En bref

L'Institut National de Recherche en Informatique et Automatique (INRIA) est un organisme public à caractère scientifique et technologique qui mène des recherches avancées dans le domaine des sciences et technologies de l'information et de la communication. Ce domaine inclut l'informatique, l'automatique, les télécommunications, le multimédia, la robotique, le traitement du signal et le calcul scientifique. L'INRIA est placé sous la double tutelle du Ministère de la Recherche et du Ministère de l'Economie, des Finances et de l'Industrie. Son organisation décentralisée (6 unités de recherche dans toute la France) lui permet de collaborer avec de nombreuses universités, grandes écoles, organismes de recherche et entreprises.

1.1.2 L'INRIA en quelques chiffres

Le budget total est de 123 millions d'euros dont 22% de ressources propres. L'INRIA compte 400 chercheurs, 500 ingénieurs et techniciens, 750 post-doctorants, stagiaires et invités, 800 doctorants, 450 chercheurs et enseignants d'autres organismes, et 200 "ingénieurs experts" (sur contrat de recherche). De plus, quatre vingt sociétés sont issues de l'INRIA, parmi elles, Ilog, aujourd'hui cotée au Nasdaq. En 2004, l'INRIA disposait de 175 brevets actifs, 120 logiciels distribués en accès libre ou commercialisés.

L'INRIA est un des membres initiateurs du consortium ObjectWeb avec, entre autre, France Télécom et Bull. Cet institut est aussi à la base du logiciel de calcul scientifique "Scilab" et du consortium du même nom.

1.1.3 L'INRIA Rhône-Alpes

L'unité de recherche INRIA Rhône-Alpes regroupe 25 équipes avec des partenaires tel que le CNRS, l'INPG, l'ENS de Lyon et bien d'autres, pour un total de 500 personnes. Ses activités sont organisées autour de cinq pôles : les systèmes communicants, cognitifs, symboliques, numériques et biologiques.

1.2 Le service SED

Le service Support, Expérimentations et Développement logiciel (SED) est une cellule de soutien aux projets de recherche menés à l'INRIA Rhônes-Alpes. Ce service est à la fois support d'expérimentations et de développements.

Dans le cadre du support à l'expérimentation, ce service est chargé :
- de la maintenance des plateformes expérimentales (robotique, clusters, réalité virtuelle)
- du soutien au développement par la mise en place d'expérimentations ou de logiciels
- du soutien à la recherche par la participation aux expérimantations

Au niveau du support au développement logiciel, le service est principalement chargé :
- de la mise en place et de l'animation d'une structure d'échange afin de favoriser le partage des connaissances entre les développeurs
- de l'aide à la conception de logiciels par l'apport de compètences techniques
- du suivi des développements et de la contribution à la formation des étudiants

1.3 Présentation du sujet et des objectifs

Le contexte de mon stage se situe autours des plateformes clusters et réalité virtuelle. La création d'applications distribuées de réalité virtuelle nécessite d'une part du matériel d'acquisition, de restitution et de calcul (tel que caméras vidéos, vidéo-projecteurs et une grappe de PC). Elle nécessite d'autre part une interface logicielle répondant aux besoins de communication en temps réel et de synchronisation entre les modules d'acquisition, de calcul et de restitution, répartis sur la grappe de PC. Cette interface logicielle, nommée FlowVR, est le fruit du travail de l'équipe MOAIS[1].

FlowVR est une bibliothèque middleware permettant de faciliter le développement d'applications de réalité virtuelle distribuées sur une grappe de PC. Le développement et l'exécution d'une application FlowVR passe par plusieurs étapes. L'application est décomposée en modules (des programmes) qui sont distribués sur une grappe de PC (un cluster) et échangent des données suivant un schéma défini par l'utilisateur (appelé le réseau FlowVR). La description des propriétés des modules ainsi que celle du réseau FlowVR utilise XML. Ces fichiers de description sont ensuite analysés et transformés par plusieurs outils pour générer une liste de commandes qui permet de lancer l'application. Au cours de l'exécution, d'autres outils permettent de modifier le réseau FlowVR ou de récupérer des informations de performance. L'utilisateur doit assimiler l'utilisation de chacun de ces outils avec leurs lignes de commandes aux options multiples et doit être vigilant quand à l'ordre dans lequel ils sont utilisés. Il en résulte de nombreuses erreurs qui compliquent l'utilisation de FlowVR.

L'objectif du stage est de développer une interface graphique pour FlowVR masquant la diversité des outils sous-jacents et guidant l'utilisateur de manière intuitive dans l'enchaînement des tâches.

Je tiens à remercier Nicolas TURRO (service SED) pour son encadrement à l'INRIA, ainsi que les membres du projet MOAIS, en particulier Jeremie ALLARD et Bruno RAFFIN, pour leur collaboration et leurs conseils tout au long de mon stage.

[1] Multi-programmation et Ordonnancement sur ressources distribuées pour les Applications Interactives de Simulation (http ://moais.imag.fr)

Chapitre 2

Contexte technique

Le logiciel développé au cours de ce stage n'étant pas autonome, je vais, dans une première partie, présenter le contexte de FlowVR. J'ai en effet du, dans un premier temps, me documenter et mener une série de tests avec FlowVR, afin de saisir son mode de fonctionnement. Dans une deuxième partie, je présenterai les technologies et les outils avec lesquels j'ai travaillé.

2.1 FlowVR : principes

Une application FlowVR est un ensemble de modules, éventuellement distribués, qui s'échangent des données. Chaque module itère dans une boucle infinie, en consommant et en produisant des données. Les modules sont des programmes (des processus), dotés de ports d'entrée et de ports de sortie pour communiquer. C'est le moteur de FlowVR qui se charge d'acheminer les données entre les modules, d'un port de sortie vers un port d'entrée. FlowVR est donc une bibliothèque de programmation d'applications, permettant de créer des modules ou de convertir des programmes existants en modules. En ce qui concerne l'échange de données, FlowVR définit un réseau abstrait avec des fonctions avancées, du routage simple à de complexes opérations de synchronisation ou de filtrage de messages. À chaque message est associé une liste d'entêtes. Ces dernières contiennent des informations permettant de router ou de filter les messages. Cette liste d'entêtes peut être routée séparement du message vers des noeuds spéciaux du réseau, chargés des règles de synchronisation. Différents réseaux FlowVR peuvent être conçus sans modifier ou recompiler les modules.

2.2 Exemple d'application FlowVR : GrImage

La plateforme GrImage[1], hébergée à l'INRIA Rhône-Alpes, est dédiée aux applications de réalité virtuelle. Elle permet d'obtenir de hautes performances grâce à ses 5 PC dédiés au calcul, ses 6 PC dédiés à l'aquisition (chacun étant relié à une caméra numérique) et ses 16 PC dédiés au rendu (chacun étant relié à un vidéo-projecteurs, constituant ainsi un affichage de 4 mètres sur 3 mètres d'une résolution de 4096x3072 pixels). Les noeuds sont reliés entre eux par deux réseaux Gigabit. Pour des applications nécessitant de gros besoins de calcul, GrImage est relié à la grappe I-cluster2[2], constituée actuellement de 104 noeuds bi-processeurs, reliés par un réseau Myrinet.

[1] http ://www.inrialpes.fr/sed/GrImage

[2] de plus amples informations peuvent être trouvées à l'adresse http ://www.inrialpes.fr/i-cluster2/

C'est sur cette plateforme que la plus importante application FlowVR fonctionne. Une chaine de 7 algorithmes de vision répartis permet d'extraire le modèle 3D d' un opérateur, aquit par des images 2D fournies par les cameras. 3 modules permettent de calculer les differents composants d'une scène virtuelle 3D. 2 modules permettent d'immerger le modèle 3D du personnage dans la scène virtuelle et de calculer les interactions en temps reel. 2 modules permettent de controller les différents paramètres de l' application via une interface QT et un joypad. L'affichage final est ensuite distribué sur les 16 PC/projecteurs en utilisant des modules de l'extension Flowvr render.

2.3 FlowVR : développement d'une application FlowVR

La figure 2.1 présente un exemple de réseau d'une application FlowVR. Cette application, Prime, est constituée de trois modules : «compute» est chargé de tester des nombres afin de déterminer s'ils sont premiers, «visu» représente les nombres premiers calculés sous forme graphique et «capture» contrôle l'application en traitant les événements provenant du clavier. Cet exemple est calqué sur le modèle MVC, qui permet de dissocier facilement en différents modules la partie `Model` (compute), `View` (visu) et `Controller` (capture).

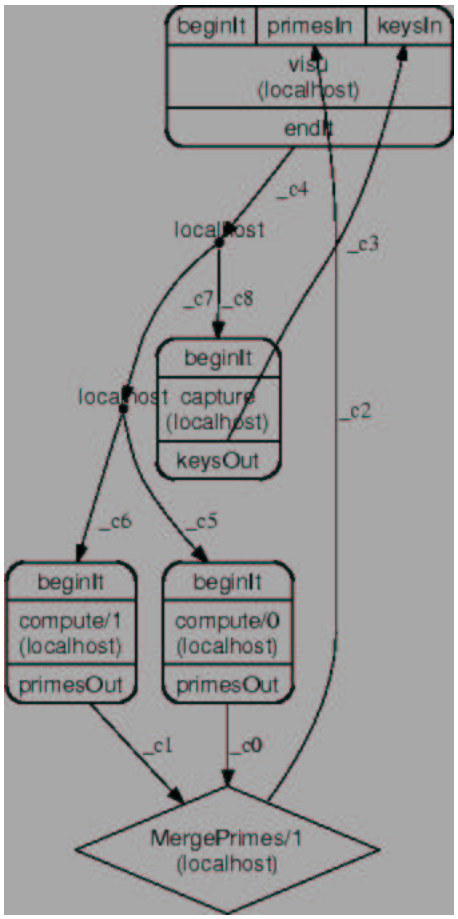

FIG. 2.1 – Un exemple d'application FlowVR : Prime

Les modules (les boîtes rectangulaires) contiennent des ports de communication (primesIn,

endIt, ...), reliés les uns aux autres. Des filtres (ici le filtre `Merge`) peuvent être ajoutés.

La création d'une application FlowVR se déroule en plusieurs étapes. La figure 2.2 présente la chaîne de développement d'une application FlowVR. Chacune de ces différentes étapes est décrite ci-dessous.

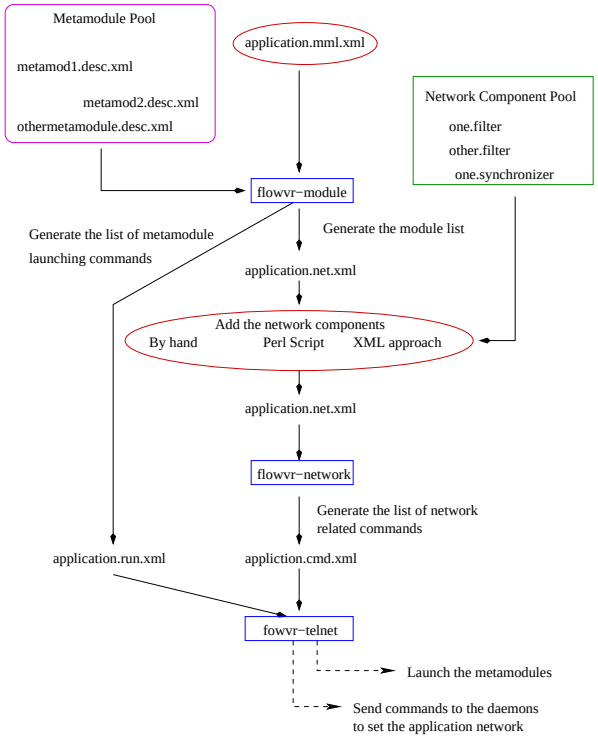

FIG. 2.2 – Étapes du développement d'une application FlowVR

2.3.1 Étape 1 : description des modules à utiliser

Comme nous l'avons dit, un module est un binaire exécutable et autonome, développé généralement en C++, à l'aide de la bibliothèque de FlowVR. A chaque module est associé un fichier de description. Ce dernier, au format XML, contient des informations décrivant le module, comme par exemple le chemin vers le fichier binaire du module, les paramètres devant être passés à l'exécutable, ou encore la liste de ses ports d'entrée et de sortie. Ce fichier est parsé au lancement d'une application FlowVR afin de définir le module à utiliser.

2.3.2 Étape 2 : Choix des modules et de leur disposition : le fichier MML

Une fois une bibliothèque de modules constituée, accompagnée des fichiers XML de description, vient l'étape de définition du fichier XML Memamodule List (MML), permettant de choisir les modules à utiliser, le nombre d'occurence de chaque module, ainsi que leur attribution sur les différentes machines de la grappe.

Une liste de module est donc définie. Pour chaque module de la liste, un identifiant lui est attribué, ainsi que le path vers son fichier de description, la liste des machines sur lesquelles le module sera lancé, les paramètres de lancement du module, etc.

2.3.3 Étape 3 : Création du réseau : le script perl

Il est maintenant nécessaire de relier les différents ports des modules entre eux. Bien que cela ne soit pas indispensable, on écrit souvent un script perl qui permet de définir les éventuels filtres à utiliser, et de relier les ports entre eux. Le fait d'utiliser un langage de haut niveau (ici perl) apporte beaucoup de souplesse, grâce à ses structures conditionnelles et itératives. Il est par exemple facile de relier tous les ports de sortie des modules d'aquisition aux ports d'entrée des modules de calcul, par une simple instruction d'itération. Le script sera donc toujours correct si on souhaite l'utiliser ultérieurement, dans une autre application pour laquelle il n'y aura pas le même nombre de modules.

2.3.4 Étape 4 : Création du réseau : le fichier net

Ce fichier XML représente les connexions entre les modules. Il peut être tappé à la main, ou généré automatiquement par la commande `flowvr-module` à partir du fichier MML, des fichiers de description des modules et du script perl.

2.3.5 Étape 4 : Lancement de l'application

Une fois le réseau défini, il est possible de lancer l'application. Pour cela, il faut que, sur chaque noeud, soit lancé un démon FlowVR (commande `fowvrd`). Ce démon, contrôlé par un langage de commandes basé sur XML, permet de contrôler chaque noeud de l'application.

Un contrôleur doit alors être défini. C'est un module spécial, lancé sur un des noeuds de la grappe, qui est chargé de lancer les processus sur les différents noeuds et d'envoyer des commandes de contrôle aux démons. `flowvr-telnet` est un contrôleur bas-niveau, lisant sur l'entrée standard les commandes de lancement (run) et celles de contrôle de l'application (command). Les fichiers de lancement (run.xml) et de contrôle (command.xml) à envoyer sur l'entrée standard sont au format XML. Ils peuvent être générés respectivement par `flowvr-module` et `flowvr-network`.

2.4 Technologies et outils utilisés

Les étapes précédentes, indispensables à la mise au point d'une application FlowVR, sont assez fastidieuses. Elle demandent à l'utilisateur de la rigueur et une parfaite connaissance de la grammaire des fichiers XML. Afin de simplifier la création d'une application FlowVR, il a été décidé de créer une interface permettant de masquer les quatre étapes définies ci-dessus. L'application graphique, nommée FlowVR GUI, est capable, à partir de modules compilés, de simplifier la tâche de l'utilisateur dans la création des fichiers de description, du fichier Metamodule List, du script perl ainsi que de l'accompagner dans le lancement et le contrôle de l'application.

FlowVR GUI présente deux avantages principaux. Il permet d'une part l'accès à FlowVR à des utilisateurs moins expérimentés, en masquant la complexité des outils sous-jacents. D'autre part, dans le cadre du développement d'applications complexes, il propose un système intuitif de gestion des modules : les modules sont configurés simplement et rapidement, et l'affichage matriciel de l'éditeur MML permet d'avoir une représentation graphique de la répartition des modules sur la grappe.

Cette interface pour FlowVR, développée en C++, utilise la bibliothèque graphique QT[3]. La gestion des fichiers XML (parsing et création) est effectuée en utilisant la bibliothèque Ti-

[3]Environnement de développement, complet et multi-plateformes (voir http ://www.trolltech.com)

nyXML[4]. Cela permet à l'application d'être facilement portable, même si cela n'est pas une des priorités (FlowVR fonctionnant actuellement uniquement sous Unix/Linux).

Le développement du logiciel s'est donc effectué sur une machine Linux, avec des logiciels open source : le code source développé est sous licence GPL version 2, tout comme FlowVR. Il utilise des bibliothèques open source, et a été compilé avec g++.

Les thechnologies utilisées sont relativement matures dans l'entreprise : Linux est largement adopté dans l'unité de recherche. C++ est répandu dans le développement sur grappe, grâce à sa philosophie objet d'une part, mais aussi grâce à sa rapidité d'exécution, idéale pour des applications temps réel. Quant-à QT, c'est l'environnement utilisé à l'INRIA pour le développement d'interfaces utilisateurs.

Les outils utilisés pour le développement sont les suivants :

- KDevelop : c'est un environnement de développement, très connu dans le monde de l'open source. QT et CVS peuvent s'y intégrer, pour offrir une plateforme complète de développement.

- De nombreux outils de QT, en particulier :
  - QT Designer qui permet de créer rapidement des interfaces graphiques pour QT. Les interfaces utilisateurs résultantes sont au format XML (.ui), et sont supportés par l'ensemble des plateformes compatibles avec QT.
  - QMake : cet outil développé par Trolltech permet de générer des Makefiles. Cela est intéressant lors du développement sur des plateformes hétérogènes, les Makefiles étant souvent spécifiques à une configuration.

- CVS : Concurrent Version System permet de gérer le code développé par les différentes personnes travaillant sur FlowVR. Ce système facilite le travail en équipe, pour partager notre travail et tester celui des autres.

- Wiki : espace Web commun dans lequel chaque développeur FlowVR peut proposer, informer et critiquer. Des réunions hebdomadaires nous permettent d'autre part de discuter de l'avancement de FlowVR.

[4]Petite bibliothèque C++ légère de gestion XML (http ://www.grinninglizard.com/tinyxml)

Chapitre 3

Plan de travail

Le développement de FlowVR GUI fût itératif (développement de la fenêtre principale, puis des éditeurs, les uns à la suite des autres), mais avec beaucoup d'interraction de la part de l'équipe de FlowVR.

En effet, FlowVR propose de nombreuses possibilités, qui ne peuvent pas toutes être représentées sous forme graphique. Nous avons donc due, à de nombreuses reprises lors des réunions, faire de nouveaux choix, proposer de nouvelles solutions et revenir sur des décisions prises.

J'ai donc, au cours de ces quatre mois, due revenir sur certaines parties, et reconcevoir des choses que j'estimais la veille comme finies. Je peux tout de même donner un plan de travail, approximatif, des quatre mois de développement de FlowVR GUI :

- Semaine 1 et 2 : découverte de FlowVR et test d'exemples.
- Semaine 3 et 4 : découverte du travail existant, définition de l'architecture, discussions en réunion des besoins et des contraintes. Découverte des outils de développement.
- Semaine 5 : Création de la fenêtre principale, et d'une esquisse des différents éditeurs (les deux éditeurs XML – de description de fichiers et de liste de métamodules – et l'éditeur perl).
- Semaine 6 à 8 : création de l'éditeur de fichiers de description.
- Semaine 9 à 11 : création de l'éditeur de liste de métamodules.
- Semaine 12 : création d'une première version de l'éditeur de fichier perl, en mode texte.
- Semaine 13 : intégration des différents éditeurs et développement du mode d'exécution d'applications FlowVR.
- Semaine 14 à 16 : Finitions, corrections de bugs, prise en compte des remarques des autres membres de FlowVR, rédaction du rapport.

Chapitre 4

Réalisations effectuées

4.1 Présentation de FlowVR GUI

FlowVR GUI se décompose en quatre parties. Les paragraphes suivants détaillent chacunes de ces parties.

4.1.1 La fenêtre principale

La fenêtre principale est lancée au démarrage du programme. C'est à partir de cette fenêtre qu'une application FlowVR peut-être crée ou ouverte.

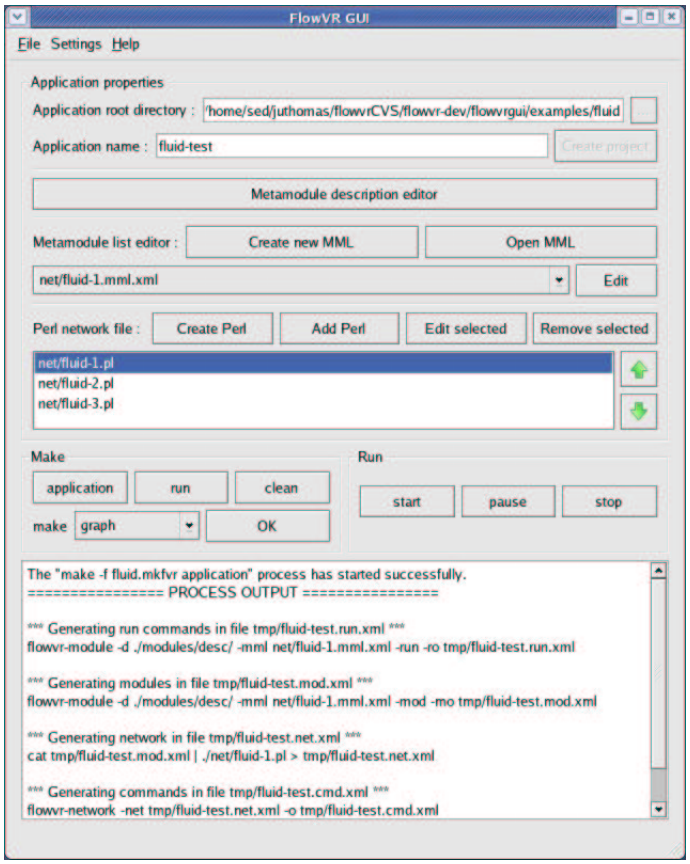

FIG. 4.1 – Fenêtre principale de FlowVR GUI

Les informations nécessaires au lancement d'une application FlowVR ont été regroupées dans un Makefile, constitué d'une entête contenant un certain nombre de variables, et de commandes permettant de générer automatiquement les fichiers XML *run*, textitcommand, *net* et de lancer l'application.

La fenêtre principale lit les variables présentes dans le Makefile, les affiche et permet à l'utilisateur de les modifier. On trouve comme variables le nom de l'application FlowVR, la liste des répertoires dans lesquels sont sauvegardés les fichiers de description des modules à utiliser, le chemin vers le fichier *Metamodule List* à utiliser, le chemin vers le script perl définissant le réseau, et enfin la liste des actions disponibles pour l'application (génération des fichiers XML *net*, *run* et *command*, lancement de l'application, affichage du réseau sous forme graphique, etc.

C'est aussi à partir de la fenêtre principale que l'on accède aux différents éditeurs.

4.1.2 L'éditeur de fichiers de description

L'éditeur de fichiers de description est un éditeur de texte XML. Il permet d'ouvrir des fichiers de description, d'en créer, et d'en sauvegarder. Cet éditeur est décomposé en deux parties.

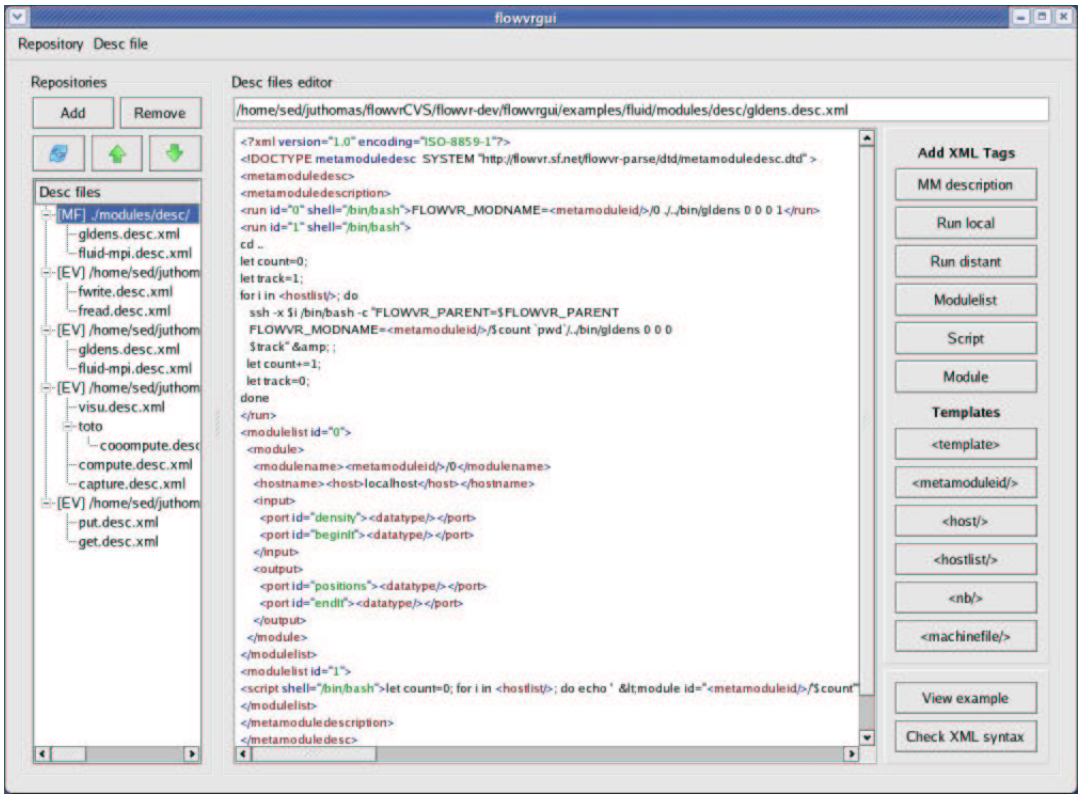

FIG. 4.2 – Fenêtre de l'éditeur de fichiers XML de description des modules

La partie de gauche est un explorateur de fichiers desc. L'alogrithme l'implémentant parcourt de manière récursive des répertoires, et liste les fichiers de description existants, sous forme d'arbre (dans un objet QT `QListView`). Les répertoires parcourus sont ceux contenus dans la variable DESC_PATH du Makefile, puis ceux présents dans la variable d'environnement FLOWVR_DESC_PATH. Dans le cas ou plusieurs fichiers de description ayant le même nom

seraient reconnus lors du parcours, seul le premier sera affiché dans la liste, et donc éditable.

Ceci fût imposé, FlowVR utilisant le même type de parcours. L'ordre des répertoires des variables DESC_PATH et FLOWVR_DESC_PATH étant important, l'interface est dotée d'un moyen de réorganiser les répertoires.

La partie droite de l'éditeur permet d'éditer les fichiers de description. Afin de simplifier l'édition, un module de coloration syntaxique a été ajouté, ainsi que des boutons permettant de créer une structure type de fichier de description. Il est possible de consulter un exemple, ou encore de vérifier si le document créé respecte bien la grammaire DTD par le bouton «check syntax» qui appelle l'outil nsgmls (éditeur par défaut, qui peut être configuré dans les préférences de FlowVR GUI).

4.1.3 L'éditeur de fichiers MML

L'éditeur de liste de métamodules permet de choisir les modules à utiliser dans l'application. Un tableau permet d'attribuer à chaque noeud de la grappe (host) le nombre d'occurrences du module qui seront exécutés.

Aprs avoir évalué de nombreuses possibilités, nous avons opté pour cette forme qui présente plusieurs avantages, en particulier l'affichage d'un maximum d'informations dans la même fenêtre et la visualisation intuitive et immédiate de la répartition des modules sur les noeuds.

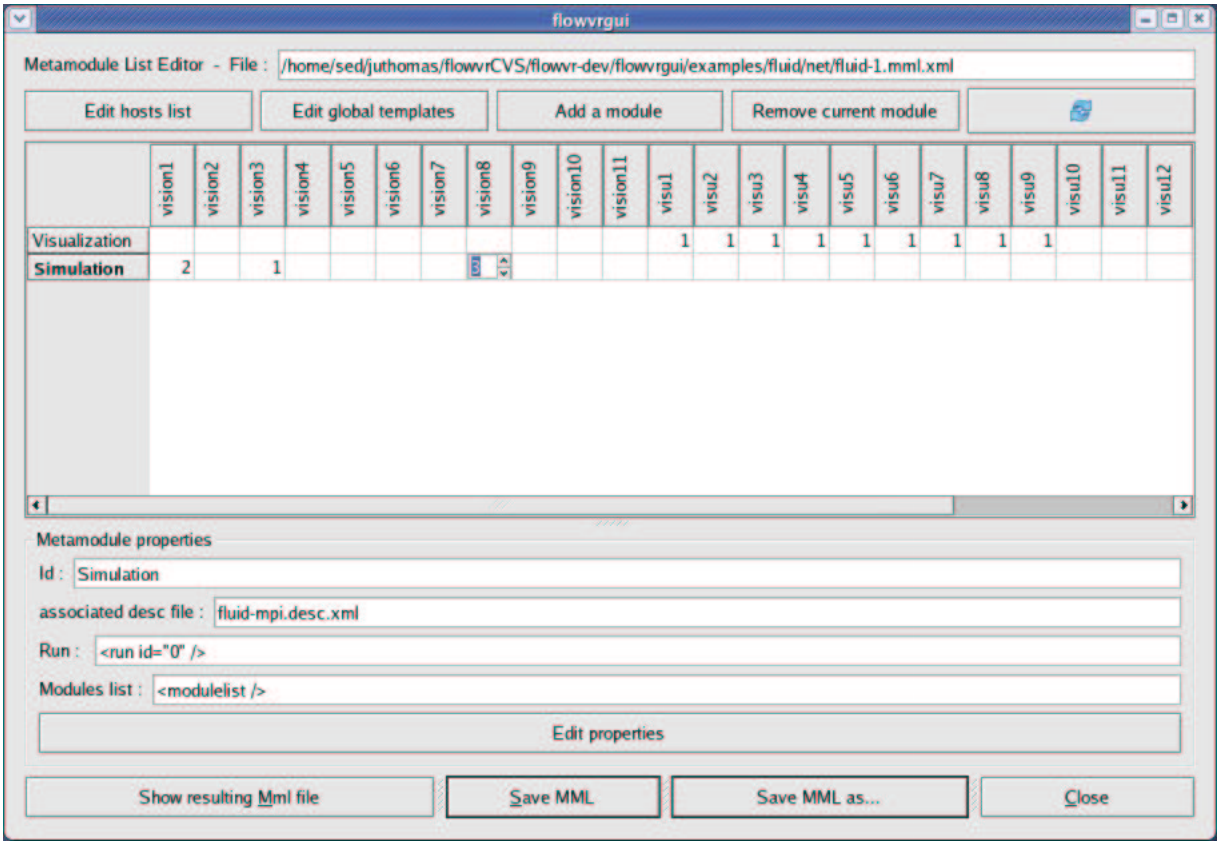

FIG. 4.3 – Fenêtre de l'éditeur de liste de métamodules

La liste des noeuds de la grappe se trouve dans un fichier texte. Ne dépendant que de la configuration de la grappe, et non de l'application, elle est sauvegardée avec les variables de

flowvrgui dans le fichier ~/.flowvrgui/hosts.list, commun à toutes les applications FlowVR. Il est cependant possible de l'éditer en cliquant sur «Edit Host List».

Utilisation des templates

Dans les fichiers XML de description de modules, des templates peuvent être définis[1]. Ce sont des balises XML, interprétées au lancement de l'application FlowVR. Afin de déterminer si un template «myTemplate» du fichier Desc doit être utilisé, le fichier MML doit contenir une balise template «myTemplate», avec un attribut id dont la valeur correspond à celle déclarée dans le fichier desc. La valeur du template peut éventuellement être définie, et servira alors de paramètre.

Les templates peuvent être définis dans le MML à plusieurs endroits. Une définition à la racine de <metamodulelist> considèrera le template comme global. Une définition se trouvant entre les balises <metamodule id="toto"> et </metamodule> considèrera le template comme local au module *toto*.

Il est possible de définir les templates globaux par le bouton «Edit global templates». Les templates locaux peuvent quant-à eux être définis en éditant les propriétés d'un module (bouton «Edit properties»).

Il est possible d'ajouter des modules ou d'en supprimer en cliquant respectivement sur «add module» et «remove module». Les propriétés des modules peuvent être modifiées en cliquant sur «Edit properties».

Une attention toute particulière a été portée à l'interface de cet éditeur, qui, bien qu'elle ne soit pas courante, se devait de rester intuitive.

4.1.4 L'éditeur Perl

L'éditeur perl permet de créer des scripts définissant les liaisons entre les différents ports des modules, ainsi que d'ajouter d'éventuels filtes, noeuds de routage ou encore synchroniseurs. Les possibilités offertes par un langage de programmation étant très vastes, il est difficilement envisageable de proposer une interface permettant de générer un script perl, en le masquant totalement.

C'est pour cela que nous avons opté dans un premier temps pour un éditeur de texte simple, permettant d'ouvrir, de créer et d'enregistrer un fichier perl. Nous avons cependant proposé une possibilité d'éditeur sous forme graphique, capable, au moins partiellement, de créer les connexions entre les modules. La création de ce mode d'édition est une des possibles évolutions de FlowVR GUI. Elle est détaillée dans le dernier chapitre «Bilan et perspective».

4.1.5 Exécution d'une application FlowVR

Une fois les étapes précédentes accomplies, FlowVR GUI est en mesure de lancer l'application.

Le fichier Makefile de l'application contient une variable action. C'est par cette variable que FlowVR GUI connait l'ensemble des actions possibles sur l'application. Les actions courantes sont `application`, `run`, `clean`, `graph` et `show_graph`.

L'action `application` permet de créer les fichiers nécessaires au lancement de l'application. Les commandes `flowvr-module` et `flowvr-network`, appelées par `make application`, peuvent

[1] une description précise des templates peut être trouvée dans la documentation de FlowVR

alors générer les fichiers `.run.xml`, `.mod.xml`, `.net.xml` et `.cmd.xml` à partir des fichiers de description, du fichier MML ainsi que du script perl[2].

L'application peut alors être lancée par l'action run (c'est-à-dire la commande `make run`). Il faut qu'un démon `flowvrd` soit lancé sur tous les noeuds de la grappe sur lesquels vont être exécutés des modules.

Il est possible de générer le réseau et de le visualiser en exécutant respectivement les commandes `make graph` et `make show_graph`. Enfin, en envoyant sur l'entrée standard de `flowvr-telnet` «pause», «start» ou «stop», l'application peut être suspendue, redémarrée ou arrêtée.

Dans FlowVR GUI, ces différentes commandes sont disponibles dans la fenêtre principale. Elles sont executées dans des processus indépendants (via la classe QProcess de QT). L'interface peut écrire sur l'entrée standard du QProcess pour contrôler l'application. Quant-aux sorties standard et d'erreur, elles sont redirigées vers l'interface, dans une zone de texte (le QTextEdit de la fenêtre principale).

4.2 Architecture

FlowVR GUI est constitué de nombreuses classes. Ces dernières peuvent être classées dans deux catégories :

- les classes de l'interface, héritant de celles de QT. Elles sont chargées de présenter l'information sous forme graphique, dans des fenêtres.
- les classes du modèle, représentant une application FlowVR. Elles sont capable de lire, de créer et de modifier les différents fichiers d'une application (le Makefile, les fichiers XML de description, de liste des modules, le script perl, etc...). Ce modèle ne contient aucun élement graphique, et pourrait être adapté à une autre interface.

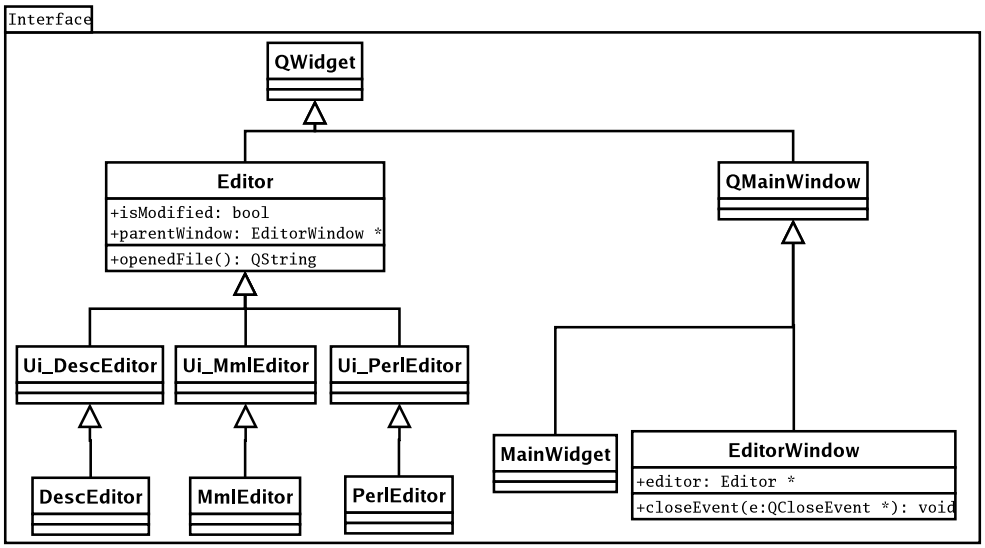

FIG. 4.4 – Diagramme des principales classes de l'interface

Le diagramme de classe de la figure 4.4 représente les principales classes de l'interface graphique.

[2] voir la documentation de FlowVR pour de plus amples détails

Les différents éditeurs de l'application, ainsi que le `MainWidget` (qui est un éditeur de Makefiles), héritent de la classe `Editor`. Le fait que ces classes soient des `QWidget` permet de les placer dans n'importe quel type de conteneur, comme par exemple des onglets, un assistant, ou des fenêtres. C'est ce dernier choix que nous avons retenu. Chaque instance de la classe `Editor` est placée dans un objet de la classe `EditorWindow`.

Le diagramme de classe de la figure 4.5 représente les principales classes du modèle.

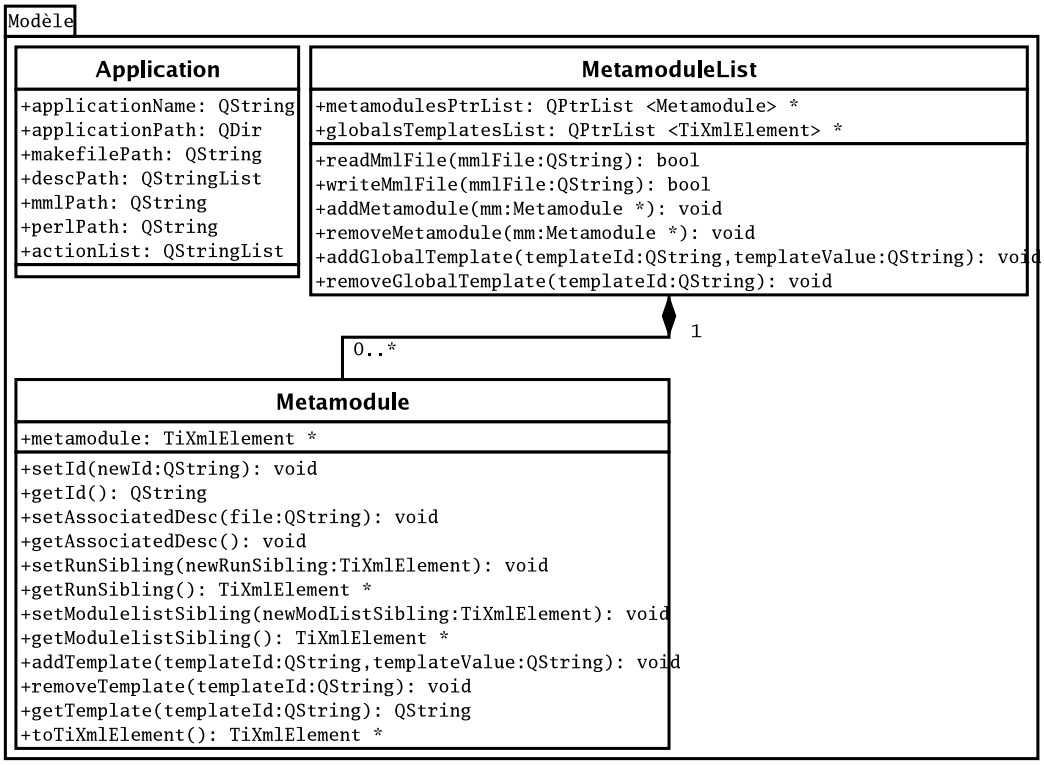

FIG. 4.5 – Diagramme des principales classes du modèle

La classe abstraite `Editor` contient des méthodes et des propriétés communes à tous les éditeurs. Cette architecture permet a FlowVR GUI de *savoir* si un fichier est en cours de modifications dans un des éditeurs. Un événement de fermeture (`QCloseEvent`) peut alors être accepté ou refusé, en consultant la valeur de la variable booléenne `isModified` de l'éditeur.

Une classe `Application` permet de sauvegarder les différents attributs d'une application FlowVR. Des méthodes permettent de lire et d'écrire un Makefile, et d'accéder aux données présentes dans l'entête du Makefile (comme le chemin vers l'application et vers les fichiers MML, perl, Desc, le nom de l'application, etc.).

L'éditeur de fichiers de listes de modules est un peu particulier : contrairement aux autres éditeurs, qui sont des éditeurs de texte, l'éditeur de fichiers MML a une forme bien différente d'un éditeur de fichiers XML : l'assocation des modules sur les noeuds de la grappe est représentée sous forme matricielle, et des formulaires permettent de saisir et de modifier les valeurs des modules. Il a donc été nécessaire de créer les classes `MetamoduleList` et `Metamodule`, qui sauvegardent respectivement les données propres à une liste de module et aux modules eux-même, et qui convertissent ces données en XML.

Chaque module est représenté dans FlowVR GUI par une instance de la classe `Metamodule`. Ces différentes instances sont stockées dans un objet de la classe `MetamoduleList` (dans une liste de pointeurs). Des procédures de la classe `MetamoduleList` permettent de gérer les templates globaux à tous les modules de la liste, ou encore d'écrire et de lire des fichiers MML au format XML.

4.3 Organisation du travail et difficultés rencontrées

4.3.1 organisation du travail et orientation

FlowVR GUI n'est pas une application commune. N'étant utile que dans un contexte bien déterminé et pour une utilisation précise, il n'existe pas d'application semblable. Nous avons donc dûe innover totalement, sans pouvoir nous baser sur l'interface et les fonctionnalités offertes par des applications semblables.

Mon travail c'est en parti basé sur celui d'un autre stagiaire, Petter Enes, qui a travaillé sur le même sujet de mars à juin 2005. Son travail a permis de définir le contexte d'utilisation de l'application (à quel public s'adresse FlowVR GUI, quelles fonctionnalités doivent être présentes, etc.). Il a d'autre part proposé une première version de l'interface de la fenêtre principale.

Je me suis donc inspiré de son travail pour concevoir l'application. Cela m'a permis de me concentrer sur l'architecture, et de proposer en réunion plusieurs versions des différents éditeurs de FLowVR GUI. Le travail en équipe a ici joué un rôle important, car j'ai pu compter sur les développeurs de FlowVR pour faire des propositions et des remarques sur mon travail. Plusieurs phases de tests ont été nécessaires, ainsi que des réunions, afin de définir les fonctionnalitées à implanter et l'IHM qui sera la plus adaptée et la plus intuitive pour réaliser les différentes tâches nécessaires à la création d'une application FlowVR. Le travail avec l'équipe de FlowVR a donc été indispensable, leur connaissance du projet FlowVR leur permettant de définir les besoins auxquels doit répondre FlowVR GUI.

Un choix à souligner est la licence de FlowVR, et donc de FlowVR GUI. Le fait que la licence retenue fût la GPL a eut un impact sur la conception du logiciel. En effet, développer un logiciel libre permet de profiter du travail des autres. Nous avons donc pu récupérer certains modules, comme par exemple la classe KXESyntaxHighlighter, qui permet d'ajouter la coloration syntaxique à un champ de texte. Bien que pour cette permière version de FlowVR GUI nous ayons opté pour un éditeur XML simple (celui de l'éditeur de fichiers de description), il sera possible d'en implémenter un plus évolué par la suite, en s'inspirant voir en récupérant certaines fonctionnalités d'éditeurs XML existants sous licence GPL. Développer un logiciel libre permet donc d'économiser du temps de développement.

4.3.2 Probèmes techniques rencontrés

Au cours du développement de FlowVR GUI, certains aspets techniques ont nécessité du temps de développement ou des recherches particulières, qu'il est intéressant de détailler.

QT présente de nombreux avantages. C'est une bibliothèque complète, dans laquelle les classes sont organisées en différents groupes, couvrant de nombreux besoins lors du développement d'une application (on trouve des groupes de classes permettant la conception d'une

interface graphique, d'accès au réseau, de parseur XML, de traitement d'images et de sons, etc.).

Cette bibliothèque est cependant en constante évolution, et présente encore quelques lacunes. Il arrive donc souvent que des méthodes courantes fassent défaut, ce qui contraint le développeur à les créer ou à appeler plusieurs méthodes successivement, ce qui crée un code moins optimisé.

On peut par exemple citer l'impossibilité d'afficher le texte d'un `QLabel` verticalement. Ceci fût nécessaire dans l'éditeur de fichiers MML, pour tableau (`QTable`) d'association des modules sur les noeuds de la grappe. Il a été nécessaire de créer une image contenant du texte, de lui appliquer une rotation, puis de créer un icône à partir de cette image.

Un autre exemple est l'utilisation d'une *spin box* (classe `QSpinBox`) dans un tableau (`QTable`). Cela n'est pas possible par defaut dans QT, les seuls éléments pouvant être mis dans un tableau (les `QTableItem`) étant du texte, des *combo box* et des *checkbox*. La classe nécessaire a donc due être créée.

On notera d'autre part l'absence d'une classe représentant les entiers, qui pourrait fournir des méthodes de haut niveau, comme la conversion d'un `QString` en entier.

J'aimerais aussi citer une autre fonctionalité intéressante, qui a été couteuse en terme de temps de développement. Il s'agit de l'affichage des fichiers de description dans une arborescence. Le seul *widget* permettant de trier des données sous forme d'arbre est la classe `QListView`. Les différents éléments de cette liste, les `QListViewItem`, ne sont pas adaptés à l'affichage de fichiers. Il a donc été nécessaire d'étendre la classe `QListViewItem` pour ajouter certaines propriétés propres aux fichiers (comme leur chemin), la possibilité de réorganiser les répertoires, suivant leur type (par la réimplementation de la méthode *compare*), etc. D'autre part, plusieurs algorithmes récursifs ont été implémentés pour parcourir les répertoires et reconstituer l'arborescence dans le `QListView`.

Le fait que QT propose ses propres types offre de nombreux avantages, comme de nombreuses méthodes de haut niveau permettant de manipuler les données. C'est par exemple le cas pour les chaînes de caractères (`QString`), les fichiers (`QFile`), les listes (`QStringList`, `QPtrList`, etc.), etc. Cependant, cela peut poser des difficultés lorsque l'on utilise QT avec d'autres bibliothèques. Ce fût le cas dans FlowVR GUI avec la bibliothèque TinyXML. De nombreuses conversions successives ont été nécessaires pour que les deux bibliothèques puissent utiliser les même données.

Chapitre 5

Bilan et perspectives

Le bilan de ce stage est très positif. Il m'a d'une part apporté une bonne méthode de travail, et il m'a d'autre part fait progresser techniquement, par la découverte et l'utilisation de nombreuses technologies.

Le travail en équipe, qui fût la méthode adpotée pour le développement de FlowVR GUI, m'a permis de bien cerner les besoins, ainsi que la problématique à laquelle doit répondre l'interface. C'est grâce à cette méthode que nous avons pu aisement sélectionner la solution idéale parmis l'ensemble des propositions émises en réunion. Le travail en équipe m'a aussi permis d'avoir de nombreux retours sur mon travail, des conseils de conception et des astuces techniques de personnes expérimentées, me permettant ainsi de gagner un temps précieux.

Sur le plan technique, le stage fût là aussi très profitable. J'ai découvert de nombreuses technologies (CVS, QT, développement distribué, ...) et j'ai pus améliorer mes compétences en développement et en ingénierie du logiciel (C++, IHM, architecture, ...).

L'objectif de ce stage, qui fût le développement de FlowVR GUI, est atteint. La version actuelle du logiciel permet d'effectuer les différentes tâches nécessaires à la création et à l'édition d'une application FlowVR.

Cette première version pourra prochainement évoluer, afin d'automatiser d'avantage certaines tâches et de proposer plus d'outils, simplifiant ainsi la création d'une application. On peut par exemple citer une évolution possible de l'éditeur de fichiers de description en véritable éditeur XML, représentant le fichier en cours d'édition sous forme d'arbre, et par exemple capable de vérifier, en temps réel, la syntaxe du fichier. Quant-à l'éditeur de scripts Perl, une version graphique utilisant la bibliothèque OpenGL pourrait être créée. Les modules, représentés en trois dimentions, pourraient être reliés par des cliques de souris, des filtres pourraient alors être ajoutés de cette même manière, etc.

FlowVR GUI quitte à la fin de mon stage la branche développement de FlowVR, pour passer dans la branche production. Il pourra alors être testé à plus grande échelle, et devenir un des élements essentiels de FlowVR.

Bibliographie

[1] Lecointre L Limet S. Melin E. Raffin B. Robert S. Allard J., Ronan G. Gouranton V. *FlowVR : Coupling Distributed Codes for High Performance Iteractive Applications*. Laboratoire ID, LIFO, CNRS/INPG/INRIA/UJF, Université d'Orléans, 2005.

[2] *Le site Web de FlowVR*. Laboratoire Gravir, Laboratoire ID, flowvr.sourceforge.net, 2005.

[3] Boyer E. Raffin B. Allard J., Ménier C. *Running Large VR Applications on a PC Cluser : the FlowVR Experience*. Laboratoire Gravir, Laboratoire ID, CNRS/INPG/INRIA/UJF, 2005.

[4] *Online QT Reference Documentation*. Trolltech, http ://doc.trolltech.com, 2005.

[5] Fogel K. Bar M. *Open Source Development with CVS - 3rd Edition*. Paraglyph press, 2003.

[6] Baudoin M. *Apprends LaTeX*. ENSTA, 1997.