Ecole Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, France
Tèl : +33 (0)2 47 36 14 14
www.polytech.univ-tours.fr

**Département Informatique**

# Projet de Fin d'Etudes

# Cahier de conception du logiciel

| | |
|---|---|
| **Projet :** | Run My Code pour la RO |
| **Emetteur :** | A. Hottin |
| **Coordonnées :** | EPU-DI<br>antoine.hottin@etu.univ-tours.fr |
| **Date d'émission :** | 06/05/2012 |

**Validation**

| Nom | Date | Valide (O/N) | Commentaires |
|---|---|---|---|
| | | | |

**Historique des modifications**

| Version | Date | Commentaires |
|---|---|---|
| | | |

# Projet de fin d'études
# Run My Code pour la RO

# Cahier de conception du logiciel

Projet encadré par Patrick MARTINEAU et Jean-Charles BILLAUT
Réalisé par Antoine HOTTIN

Mai 2012

ÉCOLE POLYTECHNIQUE DE L'UNIVERSITE FRANÇOIS RABELAIS DE TOURS
DEPARTEMENT INFORMATIQUE
64, Avenue Jean Portalis
37200 TOURS, FRANCE
Tél. +33 (0)2 47 36 14 18
www.polytech.univ-tours.fr

Dernière mise à jour : 6 mai 2012

# Sommaire

# Introduction

Ce document constitue le cahier de conception du logiciel pour le projet de fin d'études portant sur *RunMyCode pour la RO*. Ce projet est réalisé par Antoine HOTTIN et encadré par MM. Jean-Charles BILLAUT et Patrick MARTINEAU. Il est effectué dans le cadre de la cinquième année d'études au département informatique de l'école Polytechnique de l'Université de Tours.

Ce projet est effectué en collaboration avec une équipe de chercheurs de l'université d'Orléans qui est à l'origine du projet *RunMyCode*. L'objet de ce projet est de faire évoluer le projet *RunMyCode* actuel. Actuellement, *RunMyCode* est un site internet destiné aux chercheurs, permettant aux utilisateurs de partager et rendre exécutable simplement et rapidement un code source lié à une publication. L'objectif du projet *RunMyCode pour la RO* est d'adapter ce projet afin de le rendre plus facilement utilisable pour les thématiques liées à la recherche opérationnelle.

Ce document vient compléter l'ensemble de la documentation rédigée au cours de ce projet. Alors que les autres documents sont des ressources techniques destinées à décrire les éléments développés et faciliter une reprise et la maintenance du projet, ce rapport se concentre davantage sur la conduite du projet et des tâches accomplies durant le projet. Seront abordés les points suivants :

- récapitulatif de la documentation qui accompagne ce rapport ;
- description de la conduite du projet, du calendrier, des tâches accomplies et de la méthodologie employée.

# Documentation annexe

Les documents suivants, joints au présent rapport, doivent être consultés au préalable avant la lecture de ce rapport. Ils contiennent toutes les informations techniques liées au projet et à l'état de l'art et sont donc capitaux pour la bonne compréhension de ce qui sera décrit dans ce rapport :

- cahier de spécifications système : cahier des charges technique, état de l'art, description des tâches à accomplir et planning ;
- cahier de spécifications logiciel : description des solutions adoptées pour répondre aux problématiques évoquées dans le cahier de spécifications système ;
- cahier de spécifications du logiciel et de codage : document décrivant la façon dont les solutions ont été conçues et implémentées en vue d'assurer une maintenance facilitée des produits.

# Conduite du projet

Cette section constitue le cœur du rapport du projet de fin d'études. Il se concentre particulièrement sur la façon dont le projet a été conduit, les choix qui ont été faits, les méthodologies employées, les difficultés rencontrées ainsi que les erreurs qui ont pu être commises. Je vais ainsi me concentrer sur chacun des points suivants pour revenir sur les éléments cités ci-dessus :

- les méthodes employées pour anticiper les phases d'intégration, qui conditionnent totalement la méthodologie employée pour les développements ;
- le planning prévisionnel et le planning réel, suite aux difficultés rencontrées lors du projet ;
- résumé des tâches accomplies ;
- difficultés principales rencontrées ;
- technologies employées, aspect pédagogique et personnel du projet ;
- suite et reprise éventuelle du projet.

## 1. Choix de la méthode : un choix guidé par le type de projet

*RunMyCode* est un projet initié par le laboratoire d'économie dOrléans. Comme décrit dans les phases de spécifications, c'est un espace communautaire dédié aux chercheurs qui a pour objectif de permettre le partage de publications scientifiques et de codes exécutables en ligne. Outre la dimension « sociale » de ce projet, il y a également une forte composante informatique liée à des contraintes ergonomiques. En effet, l'objectif majeur est de permettre à des personnes qui n'ont pas de maîtrise avancée d'outils informatiques d'exploiter des codes produits par d'autres, sans même disposer des suites logicielles nécessaires mais simplement d'un jeu de données test.

Depuis son lancement, du moins depuis le jour où j'ai rejoint ce projet, son importance et son ambition n'ont cessé d'évoluer favorablement. Le projet *RunMyCode* n'est pas achevé et le produit n'est pas en production. Si les phases de développement « principales » sont achevées, l'évolution du projet continue. Une équipe d'informaticiens se consacre d'ailleurs quasiment à plein temps à ce projet, et un ingénieur de la société ATOS est dédié à l'évolution du logiciel.

Par ailleurs, il y a de nombreux acteurs dans ce projet, augmentant du même coup le nombre de contraintes sur la méthodologie de développement :

- l'équipe d'Orléans se consacre au développement du projet et des logiciels ;
- cette équipe est en contact avec un prestataire externe pour les prestations liées à tout ce qui concerne les grilles de calcul.

Le projet *RunMyCode pour la RO* consiste à étendre les fonctionnalités de *RunMyCode* pour que les publications liées à la recherche opérationnelle et les codes dédiés aux solvers de recherche opérationnelle puissent être mis en ligne et exploités avec la même simplicité que tout autre script, en tirant parti au maximum des ressources mises à disposition. Cependant, *RunMyCode pour la RO* possède certaines spécificités par rapport aux autres logiciels, qui ont été détaillées dans les phases de spécifications.

Ce projet de fin d'études n'est donc pas une simple évolution de *RunMyCode*. Il ne s'agit pas de reprendre simplement un projet achevé afin de le faire évoluer, de le compléter. Le développement de *RunMyCode* se poursuit alors même que des extensions doivent y être ajoutées. Les contraintes en termes d'intégration sont donc très fortes, puisque tout système développé dans le cadre de *RunMyCode pour la RO* doit anticiper des développements qui peuvent se produire sur *RunMyCode*. Ce postulat conditionne directement la méthodologie de développement qu'il faudra choisir.

- Il s'agit d'une part de développer des composants dont le principe de fonctionnent diffère de l'architecture originale ;
- mais il s'agit d'autre part de rester fidèle le plus possible aux composants qui existent déjà, l'ensemble devant former un logiciel cohérent à l'issue de la phase d'intégration.

Pour conduire un tel projet, il est nécessaire d'avoir une vue globale du projet. Il s'agit d'un projet complexe, mélangeant de nombreuses technologies (web, architectures de calcul, réseau, …). Au début du projet, j'avais envisagé une méthodologie « classique » où je pourrais disposer de toutes les ressources nécessaires au développement : sources, documentation, architecture de test. Cependant, les contraintes de confidentialité ou celles liées au développement parallèle du projet ont rendu cette approche impossible. La nécessité de développer des composants « indépendants » est apparue, tout en gardant à l'esprit que ces composants devraient être intégrés sur un système qui pourrait avoir évolué par rapport à ce que je connaissais. Il fallait identifier avec précision à quels endroits les séparations entre les sites « classiques » et les sites « RO » se faisaient dans l'architecture.

Cette phase d'identification n'a pas été précise dans un premier temps car la vue d'ensemble du projet est restée flou durant la période de lancement du projet. Néanmoins, elle a été assez précise pour identifier les tâches à accomplir, établir un planning prévisionnel et débuter une phase d'appréhension des technologies employées.

Pour conclure, le choix de la méthodologie a été fortement influencé par le type de projet dont il s'agissait. Le développement parallèle du projet, les nombreuses technologies employées, les informations parfois incomplètes dont nous disposions ont rendu inapplicables des méthodologies de développement classiques. Une méthodologie « par composant » a fini par être adoptée à l'issue de la phase d'analyse.

## 2. Planning réel et planning prévisionnel

S'il n'était pas possible d'intervenir directement sur les éléments existants et de concevoir des solutions directement liées à l'architecture en place, il demeurait possible d'établir d'ores et déjà les spécifications du système et les spécifications du logiciel, c'est-à-dire de décrire le fonctionnement d'une architecture RO idéale.

### 2.1. Planning prévisionnel

Au démarrage du projet, l'analyse du système en place, sans disposer des ressources informatiques du projet, a permis d'identifier des tâches à accomplir :

- réalisation d'interfaces « compatibles RO », capables de fournir à l'utilisateur un moyen simple de générer des données et exécuter des tâches ;
- interfaçage *RunMyCode* et *RunMyCode RO*, la méthode de soumission des jobs étant différente dans chacun de ces cas ;
- sauvegarde et restitution des résultats d'exécution ;
- mise en forme des résultats RO.

L'identification et une estimation de charge de réalisation a permis d'établir un planning prévisionnel de la conduite de ce projet :

| N° tâche | Nom de la tâche | Durée | Début | Fin | Prédécesseurs |
|---|---|---|---|---|---|
| 1 | Création des interfaces homme-machines | 8 jours | Mer 09/11/11 | Jeu 01/12/11 | |
| 2 | Mise en place du back-end RO | 4 jours | Mer 07/12/11 | Jeu 15/12/11 | |
| 3 | Interface logiciel-logiciel | 20 jours | Mer 04/01/12 | Jeu 15/03/12 | 2;1 |
| 4 | Formatage résultats | 6 jours | Mer 21/03/12 | Jeu 05/04/12 | 3 |
| 5 | Base de données stockage résultats | 4 jours | Mer 11/04/12 | Jeu 19/04/12 | 3 |

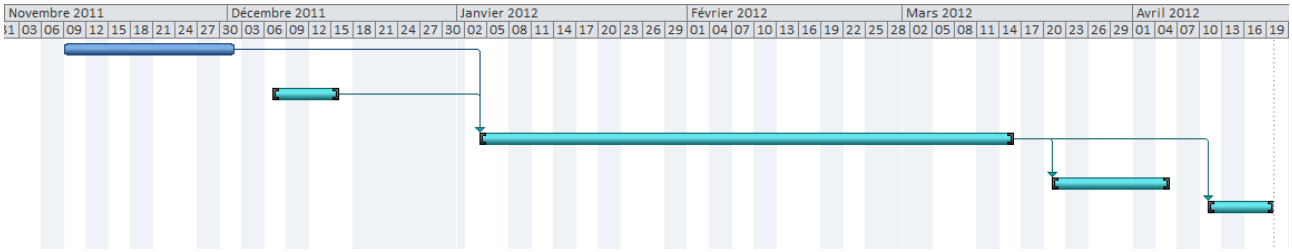


*Figure 1 - Diagramme de Gantt du planning du projet, version initiale*

Ce planning n'était volontairement que peu précis puisque lorsqu'il a été établi, je ne disposais pas encore de toutes les ressources informatiques du projet. Il m'était donc impossible d'évaluer avec précision la charge de travail que nécessiterait telle ou telle tâche. Par ailleurs, ces tâches sont davantage conceptuelles que pratiques : si chacune de ces tâches produit effectivement un résultat ou un livrable, sa description formelle n'apporte que peu d'indication sur les opérations que sa réalisation implique.

### 2.2. Planning revu à mi-parcours

Au mois de janvier, c'est-à-dire environs deux mois après le début du développement, un bilan a été fait au niveau du planning. Il apparaissait que la réalisation de certaines tâches était très laborieuse en raison du manque de ressources ou d'informations, ou parfois tout simplement de la faisabilité de la tâche. Par exemple :

- le code source du site compagnon a été mis à disposition au mois de janvier, mais la documentation n'était pas jointe aux ressources et son déploiement en local était impossible (ressources manquantes) ;
- l'architecture apparaissait trop complexe pour pouvoir être simulée en local. En effet, cette architecture complète (sites frontaux et back-end) était composé de logiciels « standards » (serveur tomcat, gestionnaire de grilles *Sun Grid Engine*) mais également de logiciels plus complexes (système de fichiers réparti *iRODS*) dont la mise en place nécessitait du matériel, voire propriétaires (file d'attente DTM).

Afin de débloquer le développement du projet, il a été décidé à partir de cette période de modifier la méthodologie employée. Ainsi :

- une approche par composant *totale* a été décidée, de sorte à développer des composants totalement indépendants et intégrables facilement, quitte à redévelopper des éléments déjà existants ;
- la mise en place d'une architecture de calcul simplifiée, mais dont le comportement est conforme à ce qui serait mis en place lors de la phase d'intégration, a été décidée.

En effet, c'est cette partie qui est le fondement du projet. Il fallait impérativement disposer d'une architecture de test pour soumettre des campagnes de tests. J'ai donc décidé de mettre en place ma propre architecture de calcul en environnement virtualisé. Cette tâche a été relativement coûteuse en temps puisqu'elle nécessitait une bonne maîtrise des systèmes d'exploitation UNIX et de certains mécanismes tels le partage de fichier, la mise en réseau, l'authentification mutualisée et les grilles de calcul.

Enfin, c'est à cette période que je suis revenu sur certains éléments qui avaient été développés au préalable. J'avais initialement proposé de mettre en place une architecture où le site compagnons réaliserait en premier lieu la génération des données, puis effectuerait autant de requêtes qu'il n'y aurait de jobs dans la campagne de tests, solution qui n'a pas été retenue par la suite au profit d'une solution ou la génération des données serait effectuée côté grille de calcul.

Pour conclure, ce bilan à mi-parcours a vu l'émergence de :

- une nouvelle méthodologie de développement, « par composant » ;
- la mise en place d'une architecture de calcul simplifiée mais fidèle à la réalité ;
- la remise en question de composants développés au préalable, comme le générateur de données ;
- la validation définitive du modèle « *RunMyCode RO* » qui consistait à encapsuler totalement la chaîne d'exécution RO dans un job « classique » au sens de *RunMyCode*, de telle sorte que l'exécution d'un job RO soit identique à l'exécution d'un job « classique » du point de vue de l'application web frontale.

Un nouveau planning prévisionnel a été réalisé :

| Nom de la tâche | Durée | Début | Fin | Prédécesseurs |
|---|---|---|---|---|
| IHM | 6 jours | Mer 01/02/12 | Jeu 16/02/12 | |
| Configuration transmission des paramètres d'instance | 4 jours | Mer 22/02/12 | Jeu 08/03/12 | 1 |
| Mise en place back-end RO | 3 jours | Mer 14/03/12 | Mer 21/03/12 | 2 |
| Commande exécution | 3 jours | Jeu 22/03/12 | Jeu 29/03/12 | 3 |
| Récupération des résultats | 3 jours | Mer 04/04/12 | Mer 11/04/12 | 4 |
| Moteur d'agrégation et de filtrage | 4 jours | Jeu 12/04/12 | Mer 09/05/12 | 5 |

### 2.3. Planning réel

Le planning suivant reconstitue finalement le déroulement des tâches effectuées depuis le lancement du projet.

| # tâche | Nom de la tâche | Durée | Début | Fin |
|---|---|---|---|---|
| **1** | **Etat de l'art, rédaction des spécifications système** | **12 jours** | **Jeu 22/09/11** | **Dim 06/11/11** |
| 2 | Recherche de documentation, prise en main des technologies | 5 jours | Lun 07/11/11 | Jeu 17/11/11 |
| 3 | Rédaction des spécifications du logiciel | 5 jours | Ven 18/11/11 | Jeu 01/12/11 |
| 4 | Développement d'un module JAVA de génération de données à partir d'un patron de données | 7 jours | Ven 02/12/11 | Jeu 05/01/12 |
| 5 | Mise en place d'une architecture de calcul test (installation VM, installation SGE, NIS, NFS, SSH, solvers …) | 7 jours | Ven 06/01/12 | Jeu 26/01/12 |
| 6 | Développement d'un module C++ de parsing de patron de données d'entrée et de génération de données | 5 jours | Ven 27/01/12 | Jeu 09/02/12 |
| 7 | Scriptage de la chaîne d'exécution RunMyCode RO, mise en place mécanismes synchronisation. | 7 jours | Ven 10/02/12 | Jeu 08/03/12 |
| 8 | Développement des interfaces JavaScript pour le site compagnon en vue de faciliter la saisie | 5 jours | Ven 09/03/12 | Jeu 22/03/12 |
| 9 | Développement d'un outil de processing des sorties du solver | 3 jours | Ven 23/03/12 | Jeu 29/03/12 |
| 10 | Rédaction documentation, tests, légères mises à niveau | 7 jours | Ven 30/03/12 | Jeu 19/04/12 |
| 11 | Intégration, tests, rédaction documentation | 2 jours | Ven 20/04/12 | Sam 05/05/12 |

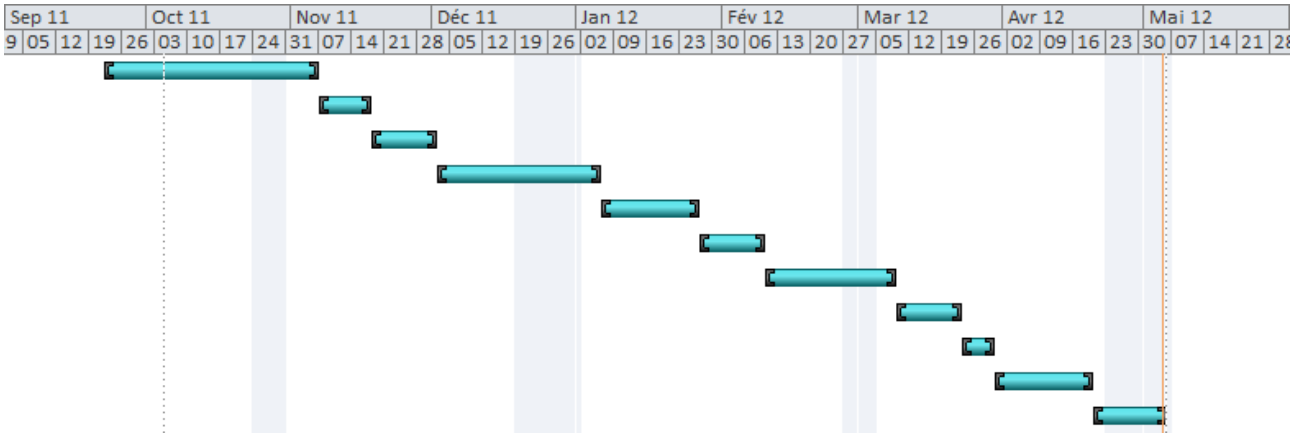


*Figure 2 - Diagramme de Gantt du planning du projet effectif*

## 3. Résumé des tâches accomplies

Cette section a pour objectif de détailler les opérations effectuées durant les tâches du planning réel énoncées dans la section précédente. Les informations techniques liées aux développements effectués dans chacune de ces tâches sont disponibles dans la documentation annexe.

### 3.1. Etat de l'art et rédaction des spécifications système

Cette tâche a consisté, avec l'aide de mes encadrants et de nos partenaires Orléanais, à analyser l'architecture fonctionnelle de *RunMyCode* et établir un cahier de spécifications système qui décrivait les tâches qui devraient être effectuées durant le projet et comment elles seraient planifiées.

### 3.2. Recherche de documentation et prise en main des technologies

Cette période a été consacrée à une prise en main des technologies qui seraient exploitées durant le projet, notamment :

- l'installation et l'exploitation du solver GLPK sur les environnements UNIX ;
- étude des possibilités qui s'offraient à moi pour réaliser des développements sur le front-office (SiteCompagnon). Mon faible niveau en J2EE m'a conduit à m'auto-former dans cette technologie tout au long du projet.

### 3.3. Rédaction des spécifications du logiciel

Cette tâche consistait à décrire de façon formelle le comportement du logiciel que nous souhaitions mettre en place. Pour rédiger ce document, je me suis appuyé sur l'architecture existante en y apportant des modifications propres à la RO.

### 3.4. Développement d'un module JAVA de génération de données

La nécessité de développer un tel programme est apparue suite à une prise de contact avec M. Billaut. L'une des spécificités de *RunMyCode pour la RO* est que l'utilisateur qui souhaite exécuter une campagne de tests ne fournit pas nécessairement une donnée, mais un patron de données, qui peut servir à générer un grand nombre de fichiers de données. J'avais initialement prévu d'intégrer ce module au sein du site compagnon afin de réaliser la génération de données en amont, d'où le choix du langage Java.

### 3.5. Mise en place d'une architecture de calcul test

L'état de l'art et la recherche de documentation m'avaient conduit à envisager un certains nombres de solutions pour l'exécution des jobs sur la grille de calcul, parmi lesquelles :

1. l'exécution séquentielle des jobs sur un nœud ;
2. l'exécution parallèle des jobs sur les nœuds de la grille de calcul et dont la synchronisation serait assurée par un module de type MPI[1] (OpenMPI), exploité grâce à une encapsulation de notre solver dans un programme qui pourrait communiquer avec les autres instances du solver ;
3. l'exécution parallèle des jobs de façon supervisée.

J'avais besoin d'une architecture de calcul test pour examiner quelles solutions étaient faisables et lesquelles ne l'étaient pas. La première option a été abandonnée car elle ne présentait pas d'intérêt : en effet, elle n'exploitait absolument pas la puissance de calcul de la grappe. La seconde possibilité a été également abandonnée, car ses conditions d'utilisations étaient trop restrictives : elle nécessitait que le solver soit open-source et que la communication internoeud soit possible. Par ailleurs, la maintenance était très compliquée. C'est finalement la dernier solution qui a été retenue, les contraintes exercées étant les plus faibles.

Pour installer une grille de calcul, j'ai décidé de déployer des machines virtuelles et réaliser une mise en réseau proche de celle d'un véritable centre de calcul. J'ai donc dû me familiariser avec les systèmes UNIX et les suites logicielles qui y sont disponibles :

- partage de fichiers ;
- authentification partagée ;
- SSH sans mot de passe avec clé pré-partagées ;
- *Sun Grid Engine*, le gestionnaire de grille de calcul ;
- ...

### 3.6. Transposition du module de génération d'instances

La solution consistant à générer les jobs en amont sur le site compagnon n'a pas été retenue en faveur d'une solution qui encapsulait les jobs RO dans un job *RunMyCode* « classique », il a fallu transposer le module de

[1] Message Passing Library.

génération de jobs dans un langage qui serait exécutable sans environnement particulier. Le choix du C++ a été retenu, étant un langage très portable et très performant qui serait à coup sûr utilisable sur tout type de grille de calcul UNIX.

La transposition en elle-même du module n'a pas été difficile mais le débogage en C++ est moins aisé qu'en Java. Par ailleurs, j'avais besoin d'une librairie capable de manipuler des expressions régulières et capable de parser des fichiers au format XML. J'ai retenu les librairies *boost* et *tinyml*. L'exploitation de ces librairies a posé un certain nombre de difficultés, notamment pour le cas de *boost* qui est une librairie très complexe. De plus, la librairie *boost* dont j'avais besoin devait être compilée et linkée séparément au programme, ce qui m'a forcé à réaliser chaque étape plusieurs fois, la première pour mon environnement de développement personnel (Windows et MinGW), la seconde pour l'environnement d'exécution final (Linux et gcc).

### 3.7. Scriptage de la chaîne d'exécution RunMyCode

Cette tâche est la tâche qui a véritablement permis de créer le prototype d'exécution RO qui était l'objectif du projet. Il s'agissait de relier chacun des composants développés au préalablement pour :

- capter les fichiers déposés sur la grille de calcul ;
- exécuter le preprocessing en appelant le module C++ qui génèrerait les fichiers de données ;
- soumettre autant de jobs GLPK qu'il n'y avait de fichiers de données en exploitant la totalité des nœuds de calcul ;
- mettre en place un mécanisme de synchronisation pour exécuter le post-processing une fois que l'ensemble des jobs serait terminé, post-processing qui permettrait de créer un fichier de résultat global.

Le mécanisme de synchronisation a été difficile à mettre en place car il ne fallait pas faire d'hypothèses qui ne seraient pas valable sur l'architecture de calcul finale. Les nœuds esclaves ne pouvant pas communiquer entre eux, la communication passerait forcément par le nœud maître. Le seul support commun entre le nœud maître et les nœuds esclaves étant le système de fichiers, c'est ce média qui a été retenu pour implémenter nos mécanismes de synchronisation.

### 3.8. Développement des interfaces JavaScript destinées au site compagnon

Pour générer notre fichier XML patron de données, la saisie s'effectuait sur le site compagnon. Il était nécessaire de fournir à l'utilisateur des outils pour le guider dans cette démarche, les syntaxes utilisées n'étant pas triviales. Afin de ne pas remettre en question le fonctionnement du site compagnon, une surcouche a été développée au-dessus des formulaires originaux du site compagnon. Cette sur couche fournit des interfaces supplémentaires qui ne se déclenchent que sur les sites RO et guident l'utilisateur dans la saisie des paramètres.

### 3.9. Développement d'un outil de post-processing des sorties

Les solvers fournissent beaucoup d'informations en sortie qui ne sont pas toujours utiles. L'objectif de cette tâche était de développer un logiciel capable, à partir d'une liste d'informations pertinentes, d'extraire des éléments du fichier de sortie global. Son fonctionnement est le suivant :

- le programme charge une liste de patterns à retenir ;
- puis il parcourt le fichier de sortie pour extraire les lignes qui matchent ces patterns.

Encore une fois, la librairie *tinyxml* a été exploitée pour définir les patterns à extraire et *boost.regex* a été exploité pour identifier les lignes intéressantes. Par commodité, ces algorithmes ont été intégrés dans le même logiciel que le pre-processing, ce qui évite de compiler plusieurs fois chacune des librairies. Le programme fonctionne ainsi en deux modes, **RunMyCodeTool –a [after]** ou **RunMyCodeTool –b [before]**. Les spécifications détaillées de ce logiciel sont données dans le cahier de conception du logiciel.

### 3.10. Tests

En attendant la phase d'intégration, une série de tests a été effectuée sur la grille de calcul de test. J'en ai également profité pour rédiger une partie de la documentation du logiciel (Doxygen pour le module C++ et cahier de conception du logiciel).

### 3.11. Intégration

L'intégration sur site a été décidée au début du mois d'avril lors d'une réunion avec Y. Stroppa. Durant les congés de pâques, cinq jours seraient consacrés à l'intégration des composants développés dans l'architecture *RunMyCode*. J'ai donc été accueilli dans le laboratoire d'économie d'Orléans. J'ai pu intégrer mes composants au niveau du site compagnon, grâce à l'aide d'Antony TONG, ingénieur ATOS consacré au développement de l'application. J'ai également pu mettre en place mon prototype d'exécution RO sur une grille de calcul de test mise en place dans le laboratoire d'économie d'Orléans. A l'issue de cette phase d'intégration, bien que certaines imperfections subsistaient au niveau de la génération des données, nous étions capables d'exécuter une demande RO complète, depuis la soumission sur le site compagnon jusqu'à la récupération du résultat sous forme de fichier PDF, en passant par la distribution des jobs sur la grille de calcul de test.

## 4. Principales difficultés rencontrées et analyse rétrospective

### 4.1. Développement parallèle

Comme énoncé en introduction de ce rapport, la difficulté principale a été de gérer le fait que le développement du projet s'effectuait en plusieurs endroits. N'ayant pas pu disposer au début du projet d'un état « final » du projet, j'ai eu du mal à identifier avec précision les tâches qu'il faudrait effectuer pour mettre en place les

fonctionnalités décrites dans le cahier de spécifications système. De plus, je n'ai disposé durant les premières phases du projet que de peu d'informations : certaines ressources informatiques mais surtout aucune documentation technique. J'ai donc rapidement dû renoncer à suivre le calendrier prévisionnel et prendre un certain nombre d'initiatives, après une légère période de latence ou je n'ai pas identifié la direction à prendre, quitte à ajouter des tâches à effectuer et en réduire d'autres. A partir du mois de janvier, le projet s'est véritablement débloqué puisque j'ai décidé de mettre en place ma propre architecture de test, ce qui m'a permis d'effectuer véritablement des développements et tests sans être trop dépendant de l'équipe orléanaise.

### 4.2. Difficultés liées aux technologies utilisées

Une autre difficulté majeure a été d'appréhender et d'acquérir une certaine maîtrise d'un grand nombre de technologies. La liste des technologies abordées durant ce projet sera effectuée dans la section « aspects pédagogiques », et certaines d'entre elles étaient nouvelles pour moi.

### 4.3. Mise à disposition des ressources

Finalement, la majeure partie de mes difficultés provient certainement d'un manque de ressources disponibles. Avec le recul, je regrette de n'avoir pas été suffisamment « insistant » pour obtenir les ressources dont j'avais besoin pour poursuivre le projet. Je regrette également de n'avoir pas pris de véritable initiative par rapport au calendrier plus tôt dans le projet.

### 4.4. Formule du projet de fin d'études

Une autre difficulté rencontrée est celle de la gestion du planning. Il faut rappeler que le projet de fin d'études se déroule tout au long de l'année, les mercredis et jeudis de chaque semaine y étant généralemet consacrés. Je considère que c'est à la fois un avantage et un inconvénient.

- C'est un avantage car si la charge de travail n'est pas directement liée à la répartition du travail, cette répartition permet une plus grande souplesse de décision ;
- c'est également un inconvénient dans le sens ou je pense que cette formule tend à réduire l'efficacité du développement. A titre tout à fait personnel, je pense qu'il est plus aisé de se concentrer sur une tâche unique sur une durée plus réduite qu'inversement. J'ai d'ailleurs déjà fait ce constat durant mes périodes de stage. En outre, une répartition sur la durée du travail n'est pas adaptée pour les projets dont le cycle de développement est parallélisé, comme c'est le cas dans *RunMyCode*.

### 4.5. Analyse rétrospective du projet

Néanmoins, au vu de ce qui a été produit et du résultat obtenu, je considère que le projet est une réussite. Toutes les fonctionnalités qui ont été décrites dans le cahier de spécifications système sont présentes dans l'architecture finale. Même s'il subsiste quelques imperfections, le système mis en place est globalement fonctionnel.

## 5. Technologies employées et aspects pédagogiques

L'aspect pédagogique de ce projet ne doit pas être oublié puisqu'il s'agit avant tout d'un projet de fin d'études. Au cours de ce projet, j'ai été amené à manipuler les technologies suivantes :

- Java 2 EE, avec HTML, CSS, JavaScript ;
- bases de données Oracle ;
- systèmes d'exploitation UNIX ;
- logiciels UNIX comme NFS, NIS ;
- programmation en script shell ;
- librairies de développement C++ : STL, boost, tinyxml ;
- environnement de développement : Eclipse ;
- gestionnaire de grille de calcul *Sun Grid Engine* ;
- système de fichiers répartis *iRODS* ;
- création de documents LaTeX automatique.

La difficulté de ce projet liée à la quantité de technologies qu'il met en œuvre constitue donc à la fois une difficulté et également un attrait, dans le sens où il m'a permis de manipuler beaucoup d'éléments et d'acquérir une expérience non négligeable dans la majeure partie de ces domaines. L'aspect du projet sur lequel je suis le plus satisfait et sur lequel j'ai le sentiment d'avoir le plus abouti est la partie grille de calcul. Je pense d'ailleurs que c'est sur les technologies liées à cette grille de calcul que j'ai acquis le plus d'expérience.

## 6. Suite du projet

A la fin du projet, un site compagnon de test est disponible sur la plate-forme de préproduction. Il est possible, sur ce site compagnon, de réaliser des demandes de jobs qui sont également exécutées sur la grille de calcul.

Tous les logiciels ayant été développés dans le cadre de ce projet l'ont été de façon générique et évolutive. Par exemple, le générateur de jobs est compatible avec tout modèle GLPK, du moment qu'il est exploité dans des conditions « normales » d'utilisation, i.e. celles décrites dans le cahier de conception. Par ailleurs, bien que seul le solver GLPK ait été mis en ligne pour le moment, tout autre solver peut être exploité moyennant les modifications suivantes :

- ajout du solver dans la liste des logiciels du MetaSite ;
- définition de la syntaxe des fichiers de données d'entrée du solver ;
- installation du solver sur la grille de calcul ;
- définition des syntaxes pertinentes dans le fichier de configuration du filtrage de la sortie de ce solver.

Aucune refonte profonde des composants n'est nécessaire pour exploiter un autre solver que GLPK. En résumé, il est possible dès maintenant de :

- mettre en ligne d'autres sites compagnons RO exploitant le sovler GLPK ;
- implémenter les traitements liés à d'autres solvers facilement.

# Remerciements

A l'issue de ce projet, je tiens à remercier chaleureusement les personnes suivantes :

- **M. Patrick MARTINEAU**, encadrant du projet, pour ses conseils et son aide précieuse tout au long du déroulement du projet ;
- **M. Jean-Charles BILLAUT**, également encadrant du projet, pour ses éléments de réponse aux questions d'ordre fonctionnel et pour ses conseils ;
- **M. Yvan STROPPA**, co-fondateur du projet, interlocuteur privilégié auprès de l'équipe Orléanaise, pour sa disponibilité, son assistance précieuse et son accueil dans les locaux de l'université d'Orléans ;
- **M. Antony TONG**, ingénieur ATOS membre de l'équipe de développement de *RunMyCode*, pour son assistance précieuse et sa disponibilité.

# Conclusion

Ce projet de fin d'études, qui consacre ma scolarité à l'école d'ingénieurs de Tours, département Informatique, aura été sans aucun doute le projet le plus conséquent et le plus difficile que j'aurai eu à conduire jusqu'à ce jour. Il s'agissait de mettre en place des solutions avancées sur une architecture très complexe, mettant en œuvre de nombreuses technologies et de nombreux acteurs. Les difficultés rencontrées sont d'ordre technique, conséquences parfois du manque d'expérience dans certains domaines ou d'un manque de documentation ou de ressources, mais sont également parfois liées à la conduite même du projet.

Ainsi, l'expérience que je retire de ce projet est à la fois technique et personnelle ; ayant appris d'une part à manipuler et mettre en œuvre de nombreuses technologies, et ayant d'autre part gagné de l'expérience en matière de conduite de projet, choses qui me serviront indéniablement par la suite.

Bien que des erreurs aient été commises et que certaines phases du projet, avec du recul, auraient pu être mieux gérées, j'ai le sentiment d'avoir au final développé une solution conforme aux attentes et suffisamment aboutie pour qu'elle soit exploitée par la suite, et éventuellement reprise pour la compléter dans le futur. Le projet *RunMyCode* est un projet très ambitieux et je suis satisfait d'avoir pu travailler sur ce projet et apporter un plus en termes de technologies sur ce projet, au demeurant très formateur.

# Table des illustrations

## Résumé :

Ce document constitue le rapport du projet de fin d'études *RunMyCode pour la RO*, réalisé par Antoine HOTTIN, DI5 (2011-2012). Ce document est joint aux documents techniques du projet : cahier de spécifications système, cahier de spécifications logiciel, cahier de conception, mais il se focalise sur la partie conduite du projet. Ce document a pour but de détailler quelles sont les tâches accomplies durant ce projet, les difficultés rencontrées, réaliser une analyse rétrospective du projet.
Durant ce projet, les thématiques suivantes ont été abordées : développement web, infrastructure réseaux, grilles de calcul.



## Mots-clés :

runmycode – RO – PFE – sun grid engine – site compagnon – LEO - GLPK


---


## Abstract :

This document constitutes the report of the end of scolarity project, which is named *RunMyCode for OR*, managed by Antoine HOTTIN, DI5 (2011-2012). This documents comes with others technical documents from this project : system specifications, software specifications, software conception, but if focuses more on the project-leading part. This documents's goal is to detail tasks which have been bone during this project, analyze difficulties of this project.
During this project, the following technologies have been used : web developpement, network infrastructure, grid engine.



## Keywords :

runmycode – RO – PFE – sun grid engine – companionsite – LEO - GLPK



Encadrants :

Jean-Charles BILLAUT
jean-charles.billaut@univ-tours.fr
Patrick MARTINEAU
patrick.martineau@univ-tours.fr

Etudiant :

Antoine HOTTIN
antoine.hottin@etu.univ-tours.fr
Département Informatique 5ème année