Camille Raymond – Christophe Gire
Damien Sendner – Thibaut Rouquette

Projet URM: Développement

University Resources Management

Résumer des problèmes rencontrés lors du développement du logiciel de gestion de réservation des salles et de la consultation de planning : URM. Vous pourrez trouver aussi un rapport sur notre méthodologie de gestion de projet.

21/03/2012

Contenu

I. Rétro-engineering: Diagramme de classe

Vous pourrez trouver en annexe un fichier « pdf » qui contient le digramme de classe généré à partir du code Java vers le logiciel « Bouml ».

Comme vous pouvez le voir, il n'y a pas de grosses différences notables entre le diagramme de classe généré par rétro engineering et celui que nous avons établi lors de la conception. Les classes de la GUI sont plus précises et possèdent les noms des méthodes Swing. Les interfaces et classes de Swing sont aussi présentes, certaines classes ont disparues, d'autres sont apparues mais le cœur de la conception est resté identique.

Nous allons maintenant vous expliciter les problèmes que l'on a rencontrés quand nous sommes passés au développement. Ceux-ci furent mineurs puisque la majorité des problèmes que nous aurions pu rencontrer ont été traités lors de la phase de conception. Par conséquent, le développement s'est bien déroulé et nous avons pu apprécier le gain tant en temps qu'en qualité de la phase de conception.

II. Problèmes rencontrés

1) PersistFactory

Un des problèmes que nous avons rencontré dans le développement que nous n'avions pas bien étudié dans la conception est la mise en place du pattern Singleton sur PersistFactory combiné au design pattern de la fabrique abstraite. Plus précisément notre problème était de savoir à quel moment il fallait créer la fabrique concrète.

Après réflexion nous avons décidé de créer la fabrique concrète dans la méthode getInstance() de la fabrique abstraite. Ainsi, si on crée un autre type de persistance, le seul endroit du code dans lequel il faut changer quelque chose sera dans cette méthode.

2) Développement de l'interface graphique

La première décision de l'équipe a été de développer les interfaces graphiques utilisateurs en utilisant l'IDE NetBeans qui permet d'implémenter automatiquement une classe à partir de glisser-déposer des widgets.

Après une génération du code peu claire et difficilement réutilisable, nous avons choisi dans un second temps de développer les interfaces de manière entièrement manuelle.

Cependant, bien que cette méthode ait été efficace pour développer des interfaces simples telles que le menu ou l'authentification, en raison des délais serrés, celle-ci n'était pas adaptée à l'implémentation de la consultation de planning ou de la demande de réservation.

Nous avons alors envisagé plus sérieusement les autres possibilités qui se présentaient à nous et nous avons trouvé Windows Builder Pro. Ce plugin Eclipse permet, à l'instar de Netbeans, de faire du développement rapide d'interface graphique. Cependant, contrairement à l'IDE détenu par Oracle, le code généré par le plugin est très lisible et structuré.

3) Connexion à la base de données

L'une des difficultés que nous avons rencontrées vis-à-vis de la persistance des données a été de trouver le driver nécessaire à la connexion à la base de données du logiciel. Il a également fallu gérer le fait que la base de données utilisée était commune afin d'éviter des modifications chaotiques. Pour cela nous avons donc décidé de copier la base de données commune sur une base de données qui nous était privée afin d'être libre de faire les modifications qu'il nous plaisait lors de nos tests.

Les vues ont été créées et sont fonctionnelles si on utilise SQL*Plus, en revanche il nous a été impossible d'afficher les types nested table sur SQLDeveloper et en Java. Nous pensons que le problème est dû au descripteur du type qui n'a pas été construit : *Echec de construction du descripteur: Unable to resolve type "CHRISTOPHE.GIRE.NT_ENSEIGNEMENT"*

III. Gestion de projet

1) Attribution des responsabilités

Nous avons attribué les responsabilités suivantes :

- Un chef de projet qui s'occupe de la coordination globale du projet.
- Un responsable conception qui s'occupe essentiellement de la répartition des tâches et du déroulement de la conception.
- Un responsable développement qui s'occupe de la répartition et du déroulement de la phase de développement.
- Un responsable base de données chargé de la répartition des tâches au niveau de la création et le remplissage de la base de données.
- Un responsable des tests chargé d'établir une stratégie de tests et de préparer sa mise en place.

Attribution:

Chef de projet	**Damien Sendner**
Responsable conception	Thibaut Rouquette
Responsable développement	Damien Sendner
Responsable BD	Camille Raymond
Responsable tests	Christophe Gire

2) Répartition du travail

a) Conception

Phase 1 : Maquette et Use Case

Nous nous sommes répartis les Uses Cases de façon à ce que la charge de travail soit égale pour chaque personne. La répartition est la suivante :

Use case - Demande de réservation	Camille Raymond
Use case - Consulter heures de service admin et enseignant	Damien Sendner
Use case - Consulter liste des demandes	Christophe Gire
Use case - Consulter planning	Thibaut Rouquette
Use case - Traitement des demandes	Damien Sendner
Use case - Authentification	Thibaut Rouquette
Use case - Consulter maquette	Thibaut Rouquette

Phase 2 : Diagramme de classe et diagramme de séquence

Dans la deuxième phase, nous avons fait en sorte que les personnes qui ont fait la maquette et le Use Case d'une fonctionnalité ne conçoivent pas le diagramme des classes et le diagramme de séquences de ce qu'ils ont fait. De cette manière, on a pu vérifier que la personne ayant fait la maquette a fourni un travail clair et compréhensible.

La répartition est donc la suivante :

Login + Menu	Damien Sendner
Consulter son planning	Christophe Gire
Demande de réservation	Thibaut Rouquette
Traitement des demandes	Camille Raymond

b) Développement

Dans la phase de développement, nous avons commencé par réaliser tous ensemble l'authentification et le menu pour que tout le monde comprennent bien tous les aspects et l'architecture de la conception mise en place.

Dans un deuxième temps, nous nous sommes séparés en deux binômes car il nous a semblé préférable de travailler à deux pour favoriser l'échange et augmenter la qualité du code (voir les études sur la programmation par binôme). Camille et Christophe ont réalisé le développement de la

consultation de l'emploi du temps, dans le même temps Damien et Thibaut ont réalisé les demandes de réservation.
Pour ce qui est du traitement des réservations, étant donné la complexité de celui-ci nous avons préféré nous concentrer sur les autres fonctionnalités et bien les finaliser que de se lancer dans son développement compte tenu du temps dont nous disposions.

Répartition :

Login + menu	Tout le monde
Consulter son planning	Christophe Gire et Camille Raymond
Demande de réservation	Thibaut Rouquette et Damien Sendner

c) Tests

Pour la conception des tests, chaque binôme a établi les tests des fonctions qu'il a développées.

3) Réunion de suivi de projet

Pour vérifier l'avancement de notre projet, nous avons effectué des réunions au moins une fois par semaine. Lors d'un changement de phase (ex : conception vers développement) nous avons effectué des réunions plus longues permettant de bien mettre en place le déroulement de cette nouvelle étape. Les décisions prises lors de ces réunions ont été le plus souvent posté sur le forum de notre gestionnaire de projet « **Team Lab** » pour que tout le monde puisse en avoir connaissance à tout moment.

4) Gestionnaire de projet en ligne : Team Lab

Après une recherche des différents outils en ligne mis à disposition pour la gestion de projet, notre équipe a retenu la plateforme **Team Lab**.
Bien que cette plateforme n'est pas parfaite, voici une liste des avantages qu'elle apporte :

- Division des responsabilités suivant les membres de la plateforme
- Création de Jalons correspondant à une activité (ex : analyse, conception, développement…)
    - Mise de délais/deadline sur les jalons
- Création de tâches que l'on associe à des jalons (ex : explication d'un use case, développement d'une fonction,…)
    - Mise de délais/deadline sur les tâches
    - Responsables affectés à chaque tâche
    - Possibilité de commenter les tâches
- Dépôt des documents en ligne
    - Classification et centralisation
    - Possibilité de consulter les pdf et les documents de la suite office
- Discussions
    - Chat : pour interagir en direct avec l'équipe
    - Forum : pour mettre des posts plus détaillés sur des sujets précis

A noter : lorsque l'utilisateur se connecte, il voit directement l'ensemble des tâches qui lui sont affectées, classées par deadline. Lors d'un retard d'une tâche, un email est envoyé à l'utilisateur et celle-ci est mise en évidence sur la plateforme (drapeau rouge).

Cet outil nous aura donc permis de gérer l'ensemble des tâches à effectuer et leurs répartition en respectant les délais prédéfinis à priori.

5) Gestionnaire de versions

Il nous a semblé important de mettre en place des outils de gestion permettant de partager le code de manière simple, de s'assurer que les différentes versions les plus à jour sont centralisées et de faciliter l'accès aux dernières versions du code pour d'éventuelles modifications.

Nous avons ainsi décidé de mettre en place un système de gestion de versions. Notre choix s'est tourné vers le logiciel client **TortoiseSVN** car il est libre et propose une interface intuitive. Grâce à son intégration dans l'explorateur Windows, un simple clic droit dans un dossier permet d'accéder à un menu permettant de mettre à jour la version sauvegardée localement et de soumettre la nouvelle version par rapport aux modifications effectuées en local.

Pour ce qui est du serveur de stockage des fichiers, nous nous sommes tournés vers Google Code, notamment car le stockage est gratuit, nous le connaissions déjà et nous utilisions également Google Document pour stocker les premiers rapports.

6) Outils de conception

Par rapport aux outils de conception, nous nous sommes d'abord orientés vers **LucidChart**, qui est un site d'édition de diagrammes collaboratif en ligne supportant le langage UML. Nous nous sommes orientés vers cette solution car elle proposait des outils collaboratifs en ligne et un stockage partagé des fichiers.

Cependant, après avoir commencé à faire les analyses pour la conception et en réalisant le nombre de classes que l'on avait, il nous a paru important de disposer d'un outil capable de générer du code automatiquement en fonction des diagrammes. C'est pourquoi nous nous sommes alors tournés vers **BoUML**, qui est un logiciel gratuit d'édition de diagrammes, qui est capable de générer du code. De plus, BoUML présentait l'avantage d'avoir des outils de gestions de classes déjà définies, nous permettant de réutiliser du travail déjà effectué, alors que réutiliser des classes dans LucidChart impliquait de les redessiner. BoUML n'est pas un outil collaboratif, mais la mise en place du gestionnaire de version nous a permis de palier à ce problème en incluant les fichiers de conception du logiciel dans les dossiers partagés.

Pour gérer les différents fichiers BoUML, nous avons d'abord fait deux fichiers séparés concernant chacun deux Use Case. Nous avons ensuite fusionné ces deux fichiers, puis traduit tous les noms en anglais car nous avions défini que la langue utilisée pour le codage serait l'anglais.

7) Outil de développement

Concernant l'outil pour développer les fonctionnalités que nous avons définies, nous avons choisi de nous tourner vers le logiciel **Eclipse**. Eclipse est un IDE très puissant et réputé être le meilleur pour ce qui est de la conception en Java. Il propose plusieurs fonctionnalités très intéressantes, comme de l'auto-complétion, une liste d'attributs et de méthodes de la classe actuelle, ainsi qu'une console intégrée. Il propose, en outre, une compilation à la volée, ce qui permet de voir très rapidement les erreurs de codage, ou encore, en laissant la souris sur une fonction, il nous en affiche la Javadoc. Une autre des raisons pour lesquelles nous nous sommes tournés vers cet IDE est que nous avions déjà eu l'occasion de nous en servir lors d'autres projets et nous le connaissons donc assez bien.

8) Outils de test

Pour effectuer les tests, nous avons utilisé la librairie **JUnit** qui nous a permis de mettre en place les tests unitaires. Cet outil nous a permis de mettre en place des tests séparés de notre code principal pour permettre plus de lisibilité et une séparation claire entre le code de l'application et le code des tests. La mise en place de ces tests plutôt dans le développement aurait pu nous faire gagner un certain temps dans le débogage. Nous y penserons la prochaine fois que nous aurons à faire un tel projet.

Nous avons aussi utilisé **ANT** qui permet sous la forme d'un fichier XML d'exécuter la compilation, la création du fichier Jar, etc. Ceci permet aussi de lancer la totalité des tests unitaires à chaque compilation, de cette façon on peut effectuer des tests de non régression car après chaque modification, on vérifie si tout fonctionne normalement.

On aurait pu utiliser un logiciel d'automatisation de tests pour effectuer les tests fonctionnels. Cependant, par manque de temps nous n'avons pas pu. De plus, sur un projet de cette taille ce n'est pas vraiment nécessaire.