

**Département de génie logiciel et des T.I.**

Rapport de Laboratoire

|  |  |
| --- | --- |
| **Numéro du laboratoire** | 03 |
| **Nom du laboratoire** | Le style architectural Invocation implicite |
| **Étudiant(s)** | Jean-Simon Bonin  Xavier Duval  Martin Gingras |
| **Code(s) permanent(s)** | BONJ02098701  DUVX20048607  GINM10108708 |
| **Cours** | LOG 430 |
| **Session** | Hiver 2012 |
| **Groupe** | 01 |
| **Chargé(e) de laboratoire** | Samir Djeffal |
| **Date** | 12 mars 2012 |

Table des matières

[Introduction 3](#_Toc319253210)

[Première partie 4](#_Toc319253211)

[Deuxième partie 5](#_Toc319253212)

[Discussion 8](#_Toc319253213)

[Conclusion 9](#_Toc319253214)

# Introduction

Ce laboratoire porte sur les liens entre une architecture du style « Invocation implicite » et les implémentations orienté objet qui la compose. Dans un premier temps, nous avions à modifier une architecture existante afin de supporter quelques fonctionnalités supplémentaires tout en prenant soin de respecter ce style architectural. La deuxième partie consistait à analyser les interdépendances de cette architecture avant et après les modifications au système avec l’outil de création de matrice de dépendance Lattix LDM.

La suite de ce rapport contient quatre sections. D’abord, une section présentera une description sommaire de notre implantation pour ce laboratoire. Dans la section suivante, nous répondons aux questions de l’énoncé de ce troisième laboratoire et effectuons l’analyse architecturale du système. La troisième section est une discussion comparant les trois laboratoires sur lesquels nous avons eu à travailler cette session. Une conclusion vient clore le document en faisant un retour sur les objectifs du laboratoire.

# Première partie

**Première partie: description sommaire de votre implantation**: Les trois premiers laboratoires comportent une partie implantation et une partie analyse. Pour la partie implantation, vous devez inclure dans votre rapport une brève description de votre expérience liée à l'implantation (ce qui a été facile, ce qui l'a été moins, etc). Réservez vos commentaires de nature architecturale pour la seconde partie.

(<https://cours.etsmtl.ca/log430/> > Laboratoires > Rapports )

**TODO**

# Deuxième partie

a)

**TODO**

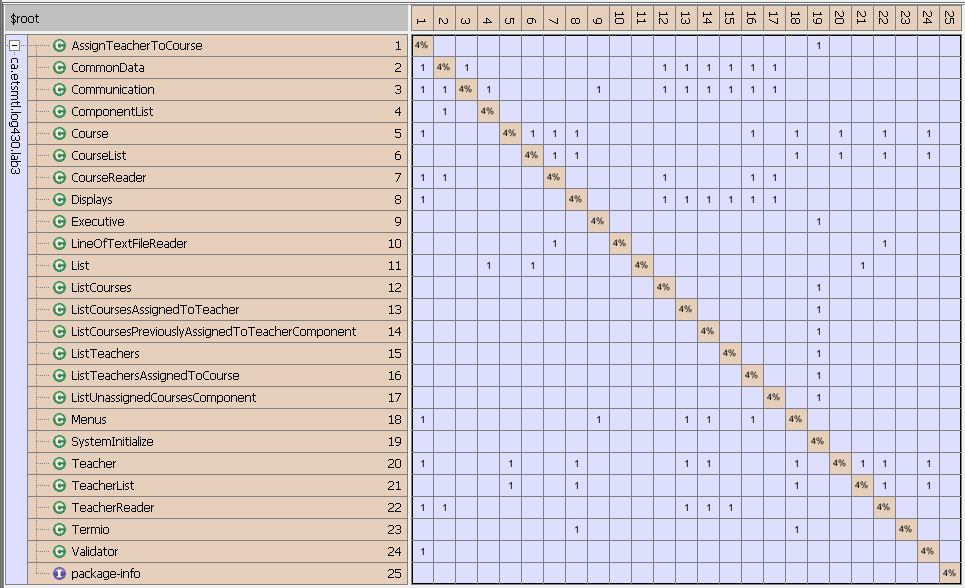
b)

Pour faire des modifications au code d’un projet dont le style d’architecture est l’invocation implicite, il faut bien comprendre le fonctionnement du patron de conception logiciel « Observer ». Dans notre cas, la classe qui est « Observable » est la classe « Communication ». Cette classe sert à écouter pour enregistrer n’importe quelle modification de l’état de l’application. Lorsqu’une modification est détectée, un message est envoyé à tous les composants qui se sont enregistré à la liste des notifications. Selon la nature de l’événement, les composants impliqués lanceront les fonctions correspondantes à la logique d’affaire de l’application.

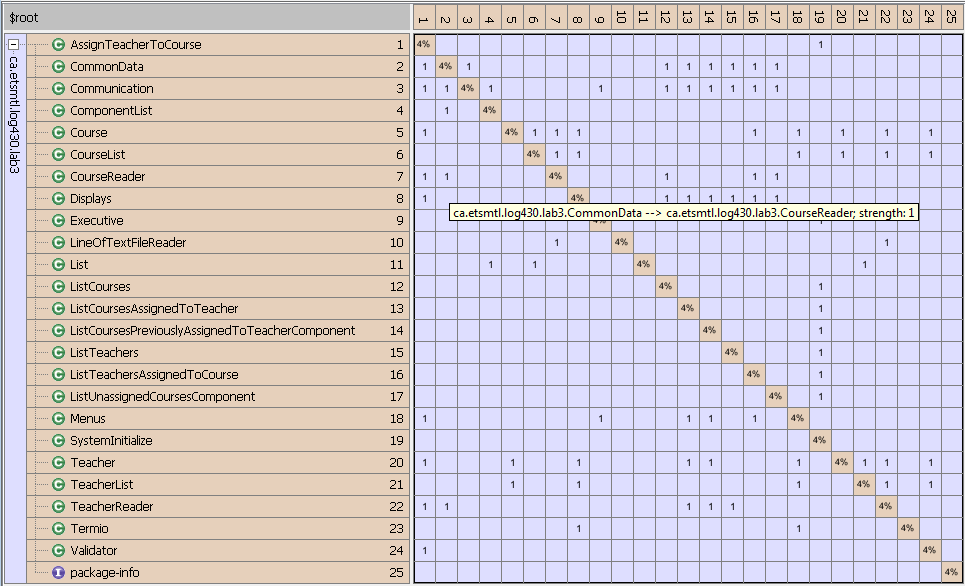
Pour respecter l’architecture d’invocation implicite, toutes nos modifications au code existant devaient respecter ce modèle. Pour ce faire, nous avons donc

c)

La matrice DSM du système original :



La matrice DSM du système modifié :



d)

Les modifications apportées à ce système sont différentes de celles du laboratoire 2 puisque dans le laboratoire 2, l’ensemble des procédures pour chacune des options du menu sont directement appelé via le « main » de l’application et l’ensemble de la logique pour chaque choix de l’utilisateur est traité à cet endroit. Pour le laboratoire 3, ces appels aux procédures sont invoqués de façon implicite puisque l’application est structurée autour de la gestion des événements. C’est-à-dire qu’au lieu de déclencher la procédure directement, on annonce la diffusion d’un événement et les autres composants qui sont abonné à cet événement démarreront la procédure qui leur est associé. De ce fait, les modifications au laboratoire 3 consistaient plutôt à créer 2 nouveaux composants pour les 2 options a ajouter et à les ajouter à la liste des receveurs qui écouteront afin de recevoir le signal associé à ces options. La logique d’affaire est donc statiquement déterminé avant la compilation pour le laboratoire 2 tandis que dans le laboratoire 3, c’est au moment de l’exécution que le couplage entre les composants est déterminé.

Certain type de changements serait particulièrement simple à intégrer à un système à invocation implicite. En effet, si par exemple nous avons à remplacer un composant par un autre, il n’y aurait aucun effet secondaire sur les autres composants car ceux-ci sont indépendants les uns des autres. De plus, nous pouvons ajouter un composant simplement en l’enregistrant en tant qu’abonné d’un événement sans aucune autre modification aux interfaces des autres composants.

D’autre part, certain autre type de changement pourraient être plus difficile à intégrer, par exemple si un changement implique que l’on doit contrôler l’ordre des réponses à un événement. Les composants qui annoncent des événements n’ont aucun control sur l’ordre des réponses reçu par les composants qui écoute pour cet événement. De plus, il peut être difficile de comprendre et de maintenir le comportement d’un composant qui annonce un événement en étant indépendant des composants qui sont enregistré à cet événement.

# Discussion

Ce laboratoire avait les mêmes exigences que le précédent en termes de fonctionnalités du système du point de vue de l’utilisateur. Seul le style architectural de l’implantation était différent. Cela a permis de comparer les deux styles afin d’obtenir une appréciation des avantages et des inconvénients des deux styles. Cependant, cela a également fait en sorte de rendre plutôt facile la première partie du travail qui concernait le code.

Outre l’adaptation au nouveau style architectural, peu de changements sont survenus dans le code que nous avions fait. Notre classe **Validator**, créée au laboratoire #2, a pu être réutilisée. L’adaptation au nouveau style s’est fait facilement. Nous n’avons pas rencontré de difficultés dans cette partie du travail. La compréhension du fonctionnement du système initial était donc la seule partie étant à la fois nouvelle et plus élevée (légèrement) en termes de niveau de difficulté. Il était néanmoins intéressant de travailler avec un style où la logique d’affaire est largement définie durant l’exécution et non statiquement définie à l’avance, avant l’exécution.

Ce laboratoire était sans doute le plus facile des trois. À notre avis, cela est largement dû au fait que nous avons pu réutiliser tout notre code du laboratoire précédent, étant donné qu’il s’agissait des mêmes exigences. Le laboratoire que nous jugeons le plus intéressant demeure le laboratoire #2. Nous sommes d’avis que les styles architecturaux étudiés aux laboratoires #2 et #3 sont applicables dans un plus grand nombre de situations que le style « Pipe and filter » qui semble plus limité et qui nécessite d’être modifié et adapté même lorsque des concepts très communs (interactivité via une interface graphique, par exemple) sont présents.

# Conclusion

Ce laboratoire avait pour objectif de nous aider à nous familiariser avec le style architectural « Invocation implicite » et de comprendre les différences fondamentales entre ce style et les autres que nous avons vu au cours des précédents laboratoires. Cet objectif a été atteint.

Nous nous questionnons toutefois sur une tâche qui a officieusement été identifiée comme obligatoire par le chargé de laboratoire : afficher les noms des cours lorsque l’on affiche les sigles pour les cours donnés dans des sessions antérieures de cette année. Il s’agit des listes de cours incluses dans le fichier de données des enseignants (**enseignantsLOG.txt**). L’application initiale ne faisait pas le lien entre ces sigles de cours et les titres des cours qui, eux, sont présents dans un autre fichier et un différent contexte d’utilisation. Les énoncés de laboratoire #2 et #3 ne mentionnent pas cette exigence et nous considérons qu’une telle modification n’est pas suffisamment triviale pour ne pas être officiellement incluse dans les énoncés.