Université de Nantes
Faculté des sciences et Techniques
M2 ALMA - Architectures Logicielles

Gestionnaire de dossier
Méthode B

Franck Boncler
Muriel Cadiot
Marie Lenogue
Julien Ouvrard

Décembre 2015

Table des matières

1 Introduction

Un logiciel se doit d'être sûr et correct à tout moment de son exécution. Pour cela, les développeurs que nous sommes, sont tenus de certifier de l'exactitude du système en cours d'exécution. Le méthode B permet donc de modéliser le comportement de ces logiciels de façon abstraite.

C'est dans cette optique que, pour ce projet, nous devions développer un système de gestion de dossier générique grâce à la méthode B. Pour ce faire, il a tout d'abord fallut définir les ensembles de données qui caractérisent notre machine. Ensuite, nous avons définit les interactions entre ces différents ensembles, ainsi que les différentes contraintes sur ceux-ci.

A la fin de ce rapport, vous trouverez un diagramme de Gantt indiquant la répartition des différentes taches en fonction du temps.
Le code de notre machine se trouve sur Github : TheForceBWithYou

2 Analyse et spécification

Le but de ce projet était de réaliser le système de gestion des dossiers clients d'une entreprise. Pour cela, nous avons définit plusieurs ensembles nécessaires à la création de notre machine :

dossiers
ensemble des dossiers

employes
ensemble des employés

dates
ensemble des dates de création ou de modification des dossiers

LIBELLE_TRAITEMENT
ensemble fini des libellés possibles pour un dossier

ETAT_TRAITEMENT
ensemble fini des états possibles pour un dossier

Nous avons ensuite définit plusieurs relations entre ces ensembles :

proprietaire
elle définit la relation entre un dossier et l'employé qui en est propriétaire. Elle relit l'ensemble **dossiers** à l'ensemble **employes**.

date_creation
elle définit la relation entre un dossier et la date à laquelle il a été créé. Elle relit l'ensemble **dossiers** à l'ensemble **dates**.

modification_etat
elle définit l'état d'un dossier à une date donnée. Pour cela, elle relit les ensembles **dossier** et **dates** à l'ensemble **ETAT_TRAITEMENT**.

traitement_lib
elle definit la relation entre la date de modification et l'intitulé de la modification pour un dossier. Elle relit les ensembles **dates** et **employes** à l'ensemble **LIBELLE_TRAITEMENT**.

traitement_etat
elle définit la relation entre la date de modification et l'état de la modification. Elle relit les ensembles **dates** et **employes** à l'ensemble **ETAT_TRAITEMENT**.

historique
cette relation permet de sauvegarder les modifications effectuées sur un dossier par un employé à une date donnée. Elle relit l'ensemble **dossiers** aux ensembles **dates** et **employes** afin d'obtenir des données dans les ensembles **ETAT_TRAITEMENT** et **ETAT_TRAITEMENT**

Toutes ces fonctions ont été mise en place afin de répondre au mieux aux fonctionnalités demandées par le sujet dans le traitement de dossiers. Nous avons essayé de simplifier au mieux ces actions en favorisant les fonctions aux nombres réduits de paramètres, afin de limiter l'effet de fonctions "en chaîne". Il est certain que nos choix sont négociables.

Nous avons également ajouté des contraintes sur la machine.
La première contrainte permet de s'assurer que l'historique d'un dossier sera limité. Pour cela nous avons défini une constante **max_op** qui représentera le nombre maximal de modifications possibles pour un dossier. Le cardinal de l'ensemble des historiques sera donc inférieur à cette constante multipliée par le nombre de dossiers. Lors de la modification d'un dossier, il faudra s'assurer que le nombre de modifications effectuées sur le dossier ne dépasse pas celui défini par **max_op**.
La deuxième contrainte permet de s'assurer que les fonctions **traitement_lib** et **traitement_etat** ont le même domaine de définition, c'est à dire que pour tous les couples **(date, employe)** il existe un **LIBELLE_TRAITEMENT** et un **LIBELLE_ETAT**

Nous avons ensuite définit plusieurs méthodes nécessaires au fonctionnement de notre machine : une méthode de création de dossier, une méthode d'ajout d'une modification et une méthode permettant de modifier l'état de traitement d'un dossier.

3 Développement

Lors du développement, nous avons donc écrit ces méthodes :

La méthode de création d'un dossier, qui prend en paramètre un employé et une date. Cette méthode permet la création d'un nouveau dossier, la définition de sa relation propriétaire, la création de sa date de création dans l'ensemble des dates, et l'enregistrement de ces modifications dans l'historique.
Dans les préconditions nous vérifions que les types des paramètres sont bien **EMPLOYE** et **DATE** et que la cardinalité de l'ensemble historique est bien inférieure au maximum d'opérations par le nombre de dossiers qui existent.

La méthode **tracerOperation** qui permet d'ajouter une modification à un dossier. Elle prend en paramètre un dossier, un employe, une date, un libellé et un état de traitement. Cette méthode va enregistrer la modification de l'état du dossier dans l'historique.
Afin que cette méthode respecte les contraintes, nous lui avons ajouté une précondition. Cette précondition permet de s'assurer que le nombre de modifications maximum pour ce dossier n'a pas été atteint. Si celui-ci a été atteint, la modification ne doit pas avoir lieu.

Pour finir, nous avons écrit la méthode de changement d'état d'un dossier qui prend en paramètre un dossier, une date et un etat de traitement et va venir modifier l'état de ce dossier dans l'historique.
Cette méthode ne possède pas de contraintes hors type des variables car elle se contente de modifier l'état d'une modification.

4 Preuve

Nous avons ensuite utilisé le système de preuve d'atelier B pour faire la preuve de notre programme. Cependant nous ne sommes pas parvenu à prouver toutes les O.Ps [1] générées. Il reste toujours un certain nombres d'O.Ps non prouvées.

Au fur et à mesure de nos preuves, nos modifications du code initial entraînaient l'apparition de nouvelles O.Ps à prouver. Nos tentatives de preuves se sont montrées infructueuses.

5 Conclusion

Nous avons réussi à implémenter une machine sans erreur, mais nous n'avons pas réussi à prouver toutes les O.Ps générées. Lors du développement, nous avons du modifier certaines contraintes que nous avions définit dans une première phase de programmation de la machine. En effet, nous nous sommes rendus compte que celles-ci n'était pas adaptées au sujet. Cela a entraîné une modification du code en conséquence, regenérant un plus grand nombre d'O.Ps à prouver.

Faute de temps, et de connaissances approfondies en Méthode B, nous n'avons pu finir pleinement ce projet.

1. Obligations de Preuve

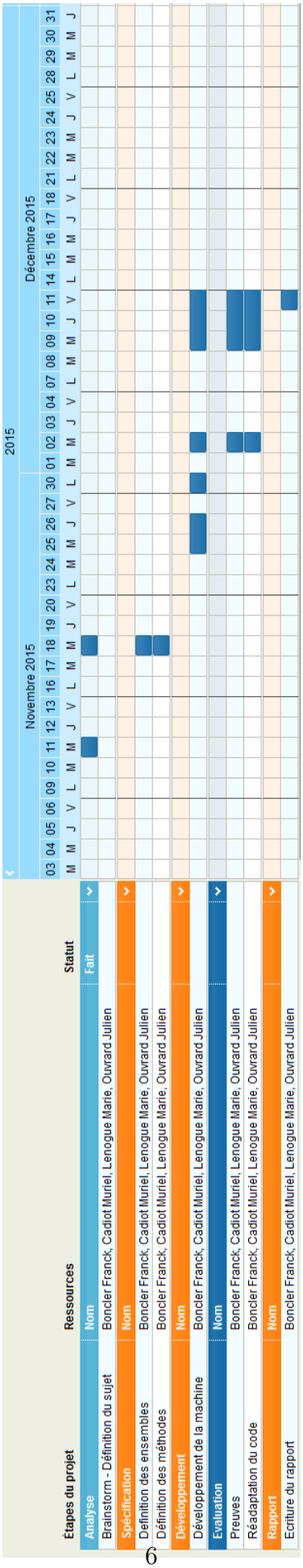

FIGURE 1 – Diagramme de Gantt