Modélisation et Spécification – Master 2 LC
TD 1 : Modélisation en utilisant la logique

Mihaela Sighireanu
(www.liafa.jussieu.fr/~sighirea/cours/modspec/)

Le but de ce TD est d'utiliser comme formalisme de modélisation la logique du premier ordre étendue avec des opérations sur les ensembles. Ce formalisme est à la base des méthodes de spécification comme Z (voir le cours de *Génie logiciel*) et UML.

Exercice 1 : *Carrefour*

Un carrefour à deux voies orthogonales (EstWest et NordSud) est contrôlé par deux paires de feux tricolores (rouge, jaune, vert).

1. Donner une modélisation d'un tel système qui fonctionne selon la législation française (pour un feu, les états rouge, jaune et vert sont exclusifs). Essayer de bien décomposer le problème afin de réutiliser les différents prédicats écrits.
2. Spécifier quels sont les invariants de votre modèle et leur donner une preuve.
3. Que faut-il modifier pour adapter votre solution à la législation anglo-saxonne (feu jaune non exclusif avec le rouge et le vert). Que devient l'invariant du modèle ?

Exercice 2 : *Édition concurrente de programmes*

Les développeurs d'un projet ont besoin de partager leur travail, mais ils doivent s'assurer que chaque choix qu'ils font est cohérent avec les choix des autres. Des erreurs peuvent être introduites quand deux personnes travaillant sur la même partie du projet, par exemple un fichier, effectuent des changement qui sont en conflit. Il est possible d'éviter ce gendre d'erreurs en utilisant un système de gestion de documents comme SCCS (*Source Code Control System*) ou RCS (*Revision Control System*).

Voila une description informelle d'un tel gestionnaire :

- Si un utilisateur demande un document pour le modifier (*check out*) alors il peut le faire s'il a le droit de le modifier et que personne d'autre n'est en train de modifier ce document.
- Dès qu'un utilisateur a obtenu le document (*check out*) pour le modifier, tous les autres utilisateurs sont interdits de modifier ce document.

Toutefois, les utilisateurs qui ont le droit de lire le document, pourront le faire.
- Quand un utilisateur a fini de modifier un document, il doit le déposer (*check in*), ce qui permet à d'autres utilisateurs de le modifier.

Donner une modélisation de ce système en suivant les étapes ci-dessous :

1. Identifier les utilisateurs du système et comment ils interagissent avec le système. (Ceci pourrait servir à définir un diagramme d'utilisation – *use case* en UML.)
2. Donner une implémentation possible du système en spécifiant quelles sont les classes utilisées et leurs méthodes. (On peut utiliser les diagrammes de classes – *class diagram* d'UML.) Par exemple, une classe est nécessaire pour modéliser les permissions sur les fichiers ; elle doit utiliser une classe qui identifie les utilisateurs et une autre classe qui identifie les documents, etc.
3. Montrer comment les classes défines ci-dessus collaborent pour effectuer la tâche finale du système. (C'est une autre façon de spécifier en UML, nommée *collaboration diagrams*.)
4. Donner un ou plusieurs exemples d'utilisation du système (scénario) en spécifiant quels sont les messages échangés (méthodes appelées) durant une période fixée de temps. (C'est très proche des diagrammes de séquence – *sequence diagrams* d'UML, aussi connus sur le nom de MSC.)
5. Pour chaque méthode, donner, quand il est nécessaire, les pré-conditions d'application de cette méthode (C'est-à-dire le prédicat sur ses entrées et l'état interne du système).
6. Pour chaque méthode, modéliser les transformations de cette méthode sur l'état du système. (C'est une modélisation logique des automates finis – *state diagrams* d'UML.)
7. La description informelle ne dit rien sur les erreurs qui peuvent apparaître (par exemple, demande de modification d'un document en train d'être modifié, etc.). Essayer de la compléter par une modélisation formelle de ces cas d'erreur.

Correction 1 : *Carrefour*

L'état du système est caractérisé par trois booléen qui expriment l'état (allumé ou éteint des chaque couleur du feu).