A propos de UML et JML

Christine Paulin-Mohring

Université Paris Sud & INRIA Futurs

Geccoo 25 octobre 2005

Outline

Aperçu d'UML

- Support «unifié» à la conception de programmes
- Indépendant du langage d'implantation mais une approche orientée objet
- Différentes sortes de diagrammes
  - Cas d'utilisation (interaction entre le système et l'environnement)
  - Diagrammes de classes, d'objets (classes, méthodes)
  - Diagrammes d'états (cycle de vie des objets)
  - Diagrammes de séquence (tests)
- Une sémantique «informelle»
- Un langage de spécification associé OCL.

Diagramme de classes et d'objets

Diagramme de classes

Décrit les entités du système

- Nom de classe
- Attributs dans des types de base
- Opérations
- Relations entre classes avec des arités
- Notions d'héritage entre classes

Diagramme d'objets

Les différentes instances des classes qui forment le système.

Diagramme d'états

Associé à chaque classe, il décrit le cycle de vie des instances de cette classe.

Transitions

- déclenchées par des évènements
  - du système : appel de méthode
  - de l'environnement : signal, passage du temps . . .
- gardées par des conditions
- produisent des actions

Langage de spécification : OCL

- Invariants sur les classes.
- Pre et Post-conditions sur les méthodes.
- Vu comme des annotations dans les diagrammes UML.
- Collection : représentation d'ensembles finis (sets, bags, séquences).
- Opérations sur les collections : `size`, `select` ...

Besoin de formalisation

- Nécessité de préciser la sémantique d'OCL.
- Modèle d'exécution associé aux diagrammes d'états : (sélection des évènements, déclenchement des actions).
- Possibilité d'instrumenter les spécifications (simulation, vérification)

Difficultés

- Interprétation du modèle informel d'UML
- Sémantique des termes indéfinis en OCL

Quelques approches de formalisation

- Aspects statiques : le projet KeY pour JavaCard
- Aspects dynamiques : UML et ASM (Abstract State Machine)
- Aspects logiques : plongement de OCL dans HOL

Le projet KeY
Principes

`http://www.key-project.org`
Chalmers, Karlsruhe, Koblenz
The KeY tool, Ahrendt et al, J. Software System Modeling, 2005

- Intégration dans un outil de développement UML
- Diagrammes de classes UML et spécification OCL (également JML)
- Des idiomes et patterns de développement
- Logique dynamique : $< p > \phi$ la propriété ϕ est vérifiée après toute exécution du programme p.
- Démonstrateur interactif et automatique (un langage simplifié de tactiques,heuristiques)

KeY : Sémantique UML-OCL

- De nouveaux types pour chaque classe.
- Les attributs de classes (statiques) sont vus comme des constantes
- Un attribut a de type D dans une classe C est vu comme une fonction dans $C \rightarrow D$
  (on note $x.a$ pour $a(x)$).
- Une relation r dirigée d'une classe C vers D est interprétée comme un attribut ie une fonction de $C \rightarrow \texttt{Set}_D$

KeY et le sous-typage

L'héritage peut se traduire en contraintes de sous-typage sur les invariants, pre et post.

Si C_2 est une sous-classe de C_1

- Sous-typage structurel : $Inv_2 \Rightarrow Inv_1$
- Sous-typage comportemental : $Pre_1 \Rightarrow Pre_2 \wedge Post_2 \Rightarrow Post_1$

Les obligations de preuve correspondantes sont engendrées.

Logique dynamique

Appel de méthodes

Java	DL
$x = f(a_1, \ldots, a_n)$	`method-frame`(x){body}

Gestion des alias

Le programme est traduit en une suite d'affectations :

$$< loc := val >$$

loc : valeur gauche ; *val* : valeur logique (sans effet de bord)
La distinction sur les locations est retardée.

Transactions

Opérateur $[[p]]\phi$ pour un invariant après chaque opération atomique.

Vérification des programmes

- Un ensemble de pre-post $pre_i / post_i$:
  précondition : $true$
  postcondition : $\wedge pre_i \Rightarrow post_i$
- $inv(self) \wedge pre(self, \vec{x}) \rightarrow < self.m(\vec{x}); > post(self, \vec{x})$
- $inv(self) \wedge pre(self, \vec{x}) \rightarrow < self.m(\vec{x}); > inv(self)$
- `attribut@pre` est vu comme un nouveau symbole de fonction de même signature que `attribut`.
  On ajoute en pré-condition
  `x.attribut@pre=x.attribut`

KeY : exemples d'applications

- Bibliothèques Java pour les ensembles
- API JavaCard
- Authentification
- Analyse de flots d'informations confidentielles
- Calcul de restriction de vitesses de train
- . . .

Abstract State Machines

`http://www.eecs.umich.edu/gasm`
Modeling the Dynamics of UML State Machines, E. Börger, A. Caverra, E. Riccobene, ASM 2000
Aussi ASM 2003 . . .

ASM

- Notions abstraites d'états et d'opérations.
- Transitions vues comme des mises à jour gardées.

Traduction de UML vers les ASM

- Capturer les aspects dynamiques et concurrents des diagrammes
- Différentes sortes d'état (composite, concurrent . . .)
- Obtenir des outils d'analyse et de simulation

Machine d'états finis vers JML

From finite state machines to provably correct JavaCard applets. Hubbers, Oostdijk & Poll, Nijmegen

Traduire une spécification sous forme d'automate d'états fini vers des spécifications JML

- Variable `mode` représentant les différents états
- Invariant sur les états :

  `mode =` a_1 `|| ... || mode =` a_n
- Contraintes pour les transitions :

  `mode=a ==> \old(mode)=`b_1`||...||\old(mode)=`b_k
  `\old(mode)=a ==> mode=`c_1`||...||mode=`c_l

Sémantique d'OCL en HOL

A. Brucker & B. Wolff
HOL-OCL : Experiences, Consequences and Design Choices,
UML 2002

A Proposal for a Formal OCL Semantics in Isabelle/HOL,
TPHOLs 2002

- Une sémantique de OCL dans HOL (shallow embedding).
- Chaque classe est interprétée comme un record correspondant aux attributs (primitifs + associations).
- Le diagramme de classes définit l'univers (extensible).
- Les classes récursives sont traitées avec une indirection `ref`
- Un état est une fonction des références dans l'univers

Sémantique d'OCL en HOL

- Les opérations sont définies de manière relationnelle sur un couple d'états.
  $pre(x, \sigma) \wedge post(x, \texttt{result}, \sigma, \sigma')$
- Une difficulté majeure est de capturer les valeurs indéfinies.
  - Fonctions strictes, sauf opérations logiques :
    false $\wedge \perp =$ **false**.
- Définition du raffinement.
- Possibilité de mécaniser le raisonnement.

Traduction d'OCL vers JML

Translating the Object Constraint Language into the Java Modelling Language, A. Hamie, Brighton.

OCL	JML
@pre	\old
collections : sets, bags ...	classes modèles
$c \rightarrow forAll(e : T \mid p(e))$	$(\backslash forall\ T\ e; c.\mathtt{has}(e); p(e))$
$c \rightarrow collect(e : T \mid exp)$	??
T.allInstances()	??

Conclusion

- Dans quelle mesure doit-on s'adapter à UML/OCL ?
  - Sémantique partielle
  - Adaptation du langage ?
- Une chaîne cohérente de développement
- Différents niveaux d'instrumentation du modèle