Méthodes formelles: mathématiques, informatique, ou génie logiciel?

Guy Tremblay
Dépt. d'informatique
UQÀM

Septembre 1998

1

1. Plan de l'exposé

- Qu'est-ce que les méthodes formelles (MFs)?
- Quand et pourquoi des cours de MFs ont-ils été introduits à l'UQAM?
- Que contiennent les cours de MFs et comment sont-ils reçus par les étudiants? (MAT3143 et INF/MGL7160)
- Que sont les mathématiques, l'informatique et le génie logiciel?
- À quel(s) domaine(s) appartiennent les MFs?

2

2. Qu'est-ce que les méthodes formelles?

Quelques définitions:

- " [T]echniques *basées sur les mathématiques* pour décrire des propriétés [de] systèmes [informatique, pour] spécifier, développer, et vérifier [de tels] systèmes [...] d'une façon systématique, plutôt que d'une façon *ad hoc*."
  (J. Wing, 1990)
- "[Les MFs] consistent à *utiliser les mathématiques* pour le développement de logiciels. Les principales activités [...] sont:
  - l'écriture d'une spécification formelle
  - la preuve de certaines propriétés de cette spécification
  - la construction d'un programme en manipulant [...] la spécification
  - la vérification du programme [...]
  (A. Hall, 1990)

3

Place et rôle des méthodes formelles dans le cycle de vie des logiciels:

- **Analyse et spécification:**
  - Description des concepts du domaine d'application
  - Spécification du comportement du produit
  - Vérification de propriétés,
  - Animation des spécifications (prototypage)
- **Conception architecturale:**
  - Description formelle de l'architecture
  - Spécification du comportement des modules
  - Vérification de propriétés
- **Conception détaillée et codification:**
  - Dérivation formelle de programmes à partir des spécifications
  - Preuves de programmes
- **Tests:**
  - Génération de cas de tests à partir des spécifications

4

Bénéfices (?) des spécifs. formelles:

- Spécifications claires (?), précises, non-ambiguës, plus explicites
  $\Rightarrow$ Meilleure base pour la conception, pour les tests
- Base pour le développement d'outils et pour des vérifications formelles
- Réduction ...
  - du nombre d'erreurs (!?)
  - des coûts de développement (!?)
- L'*effort de formalisation* force l'analyste à clarifier, formuler, expliciter, aide à révéler des ambiguités, contradictions, éléments incomplets, etc.

5

Donc, les MFs sont basées sur l'utilisation des mathématiques, mais ...

- L'aspect clé, fondamental = spécifications;
- Les mathématiques requises pour les spécifications sont faciles: logique du 1^{er} ordre, ensembles, séquences, fonctions;
- Des mathématiques (logique) plus sophistiquées sont requises seulement pour les preuves formelles;
- Ce qui est difficile dans les MFs, c'est l'aspect *spécification* et *modélisation*, pas l'aspect mathématiques!

6

3. Quand et pourquoi des cours de méthodes formelles ont-ils été introduits à l'UQAM?

Quand? Assez récemment!

- Maîtrise:
  - 2 fois à l'UQAM (hiver 96 et 97)
  - 1 fois à l'ETS (été 98)
- Baccalauréat:
  - 4 fois (depuis l'hiver 97)

Pourquoi des MFs en maîtrise?

- Introduction de la concentration génie logiciel

Pourquoi des MFs au bacc.?

- Insatisfaction des étudiant-e-s avec MAT4130 "Algèbre et logique": trop théorique!
- Modifications de programme (aut. 95):

  Algèbre et Logique

  $\longrightarrow$

  Méthodes formelles

7

4. Quels objectifs sont visés par les cours de méthodes formelles?

Caractéristiques de la clientèle:

- Niveau maîtrise:
  - Concentration génie logiciel
  - Pour professionnels: projet en entreprise plutôt que mémoire
  - Aucun pré-requis en mathématiques

  Extrait du programme:

  > "[Ce programme] vise à accroître la performance de professionels déjà actifs dans le domaine du développement ou la maintenance de logiciels ou de systèmes-informatiques."

8

- Niveau baccalauréat:
  - *Background* assez léger en mathématiques
  - Cours pour les options "Développement de logiciels" et "Informatique répartie"

  Extrait du programme:

  > "Ce programme vise à former des analystes informaticiens spécialisés dans les applications de l'informatique à la gestion. [...] [Un tel analyste] sera en mesure de procéder à l'analyse, à la modélisation et au développement de divers types de systèmes à des fins de gestion."

9

- Éléments clés, concepts importants
  - Rôle crucial des spécifications
  - Concepts et principes du génie logiciel (abstraction, encapsulation, modularité)
  - Concepts des langages de programmation modernes (classes vs. objets, héritage, polymorphisme, langages fortement typés, etc.)

- *Slogan* = **Description is our business**
  (Michael Jackson)

- Peu d'accent sur les preuves
  - Non: preuves de programme (codification, *programming in-the-small*)
  - Oui: méthode *rigoureuse* de développement (analyse, conception)

10

Impact sur la notation choisie pour ces cours:

≠ Z ou VDM

= `Spec`

"*Software Engineering with Abstractions*" (Berzins et Luqi 91)

Caractéristiques de `Spec`:

- Approche de modélisation abstraite
- Notation ASCII
- Notion de `CONCEPT`: pour formalisation du domaine d'application
- Constructions explicites (mots-clés) pour invariant, pré/ post-conditions, transitions, exceptions, etc.
- Concepts des langages modernes: modularité, encapsulation, génériques, exceptions, héritage, itérateurs
- Différents types de modules: `DEFINITION`, `FUNCTION`, `MACHINE`, `TYPE`

11

5. Que contient le cours du bacc.?

1. `Introduction`
   - Cycle de vie des logiciels
   - Rôle des spécifications
   - Que sont les méthodes formelles?
   - Exemples d'applications industrielles

2. `Notation Spec (1`^{ère} `partie): Propositions, quantificateurs et CONCEPTs`
   - Logique propositionnelle, logique des prédicats du premier ordre
   - Quantificateurs généralisés: `ALL/SOME`, `SUM/PRODUCT`, `MINIMUM/MAXIMUM`, `UNION/INTERSECTION`, `NUMBER`
   - Notions de `CONCEPT` et de module `DEFINITION`
   - Formalisation de DERs

12

3. Notation Spec ($2^{ème}$ partie): Types abstraits

- Types de base: `boolean`, `nat`,`integer`, etc.
- Ensembles: `set`{`t`}
- Séquences: `sequence`{`t`}
- Dictionnaires: `map`{`cle, defn`}

4. Notation Spec ($3^{ème}$ partie): fonctions, machines, types

- Les différentes sortes de modules
- `FUNCTION`: racines, tris;
- `MACHINE`: piles, distributrice à café, four micro-onde, club vidéos, etc.;
- `TYPE`:
  - `TYPE` mutable = classe d'objets: piles, files, boîtes vocales, commandes Bureauphile;
  - `TYPE` immuable = collection de valeurs: piles, chaînes, nombres rationnels;

Machine = objet mais Machine $\neq$ Classe

Type abstrait immuable $\neq$ Classe d'objets

5. Méthode rigoureuse de développement

- Rigoureux $\Rightarrow$ obligations de preuve à vérifier
- Obligations liées à une spécification:
  - Peut-elle être satisfaite?
  - L'invariant est-il toujours vrai?
- Raffinement et biais d'implantation:
  - Suffisamment abstraite?
- Raffinement et simulation:
  - Peut-elle simuler (satisfaire) une autre spécification?

6. Spécifications algébriques

- Spécification de types abstraits (immuables)
- Rien à voir avec l'algèbre classique (monoïdes, anneaux, corps)
- Notation Larch (LSL vs. LCL)

7. Langages réguliers, automates finis et diagrammes de transition

- ...

Matériel développé pour le cours MAT3143:

- Notes de cours ($\approx$ 400 pages) avec de nombreux exercices
- Système interactif (sur le *web*) d'aide aux exercices: Subvention FDP 1997–98
  - Énoncé des exercices
  - Indices et/ou squelettes de solutions
  - Solutions ... avec des *filtres*:
    - Questions à choix multiples
    - Recherche de mots-clés
- Analyseur `Spec`:
  - Analyse lexicale et syntaxique, vérifications contextuelles simples, fichiers d'interface: complété
    (cours de compilation en lecture dirigée)
  - Vérification des types: Subvention FDP 1998–99

6. Que contient le cours INF/MGL7160?

Principaux éléments de MAT3143
(moins de détails, présentation plus rapide)

en plus de ...

Autres méthodes de modélisation abstraite

- VDM
- Z

Systèmes réactifs et concurrents

- Notation Lotos (algèbre de processus)
- Logique temporelle et vérification de modèle

Autres thèmes (par les étudiant-e-s)

- Interfaces personne-machine
- Spécifications formelles et prototypage
- Approche *Cleanroom*
- Mise en oeuvre de spécifications formelles par traduction dans un langage fonctionnel
- Réseaux de Petri
- Modélisation formelles des scénarios d'utilisation à l'aide de *Statecharts*

7. Comment ces cours ont-ils été reçus?

- Maîtrise: Accueil très positif
- Baccalauréat:
  - Meilleur feedback que le cours "Algèbre et logique"
  - Les étudiants trouvent la matière parfois *abstraite et difficile*
  - Approche algébrique: pas appréciée

Pourquoi difficile et abstrait?

- Richesse de la notation Spec
- Entités mutables vs. immuables
- Spécification non-opérationnelle
  ⇒ **non-exécutable**

```
TRANSITION
  ALL( x: t SUCH THAT x IN s2 ::
                s1 = *s1 - {x}  )
```

17

8. Que sont les mathématiques, l'informatique et le génie logiciel?

Mathématiques

> Étude rigoureuse des structures abstraites.
> (F. Lalonde)

> Les mathématiques consistent en l'étude, par des méthodes rationnelles numériques et symboliques, des structures dans le sens général du terme, qu'elles soient concrètes ou abstraites. En particulier, par les modèles qu'elles engendrent, les mathématiques constituent le langage universel et international des sciences.
> (G. Labelle)

18

Informatique

> Étude systématique des processus algorithmiques qui décrivent et transforment l'information: leur théorie, analyse, conception, performance, mise en oeuvre et application.
>
> La question fondamentale sous-jacente à l'informatique est "Qu'est-ce qui peut être (efficacement) automatisé?"
> (ACM/IEEE)

Génie logiciel

> L'utilisation d'une approche systématique, disciplinée, quantifiable au développement, à l'opération, et à l'entretien de logiciels; c'est-à-dire, l'application du génie au logiciel.
> (IEEE)

19

Informatique vs. Génie logiciel

> L'informatique vise à toujours mieux comprendre et étendre nos connaissances de tout ce qui touche le calcul et la computabilité (*computing*). Par contre, le but fondamental du génie logiciel est d'utiliser et appliquer ces connaissances pour construire des logiciels.
> (SWEBOK)

20

Autres différences, selon D. Parnas:

- Programme de génie:
  - Carrière = concevoir des produits que d'autres utiliseront
  - Prépare les étudiants à entrer immédiatement sur le marché du travail
- Programme en sciences:
  - Carrière = étudier les phénomènes qui intéressent les deux groupes (ingénieurs et scientifiques) et étendre nos connaissances dans ce domaine
  - Accent souvent mis à préparer les étudiants à poursuivre des études graduées

Un exemple illustrant la différence de point de vue: Logique

Mathématiques (MAT2050/4050): Symbolisation. Inférences et déduction naturelle. Sémantique, interprétation, homomorphies. Théorèmes de déduction. Théorèmes fondamentaux de non-contradiction, complétude, compacité, définissabilité.

Informatique (MAT4130): Logique propositionnelle et calcul des prédicats. Syntaxe, sémantique, validité et satisfaisabilité. Clause de Horn. Algorithme d'unification et de résolution. Introduction aux systèmes axiomatiques comme paradigmes des langages de programmation. Techniques de preuve et applications à la vérification de programmes. Indécidabilité.

Génie logiciel: Utilisation de la logique pour modéliser le domaine d'application et décrire le comportement de composants et systèmes logiciels: logique des prédicats, spécifications comportementales avec pré/post-conditions et invariants. Génération et vérification d'obligations de preuve. Utilisation de logiques modale et temporelle pour décrire les propriétés de systèmes réactifs et concurrents (sècurité et vivacité); vérification automatique de modèle.

9. Conclusion

Question:

- À quel(s) domaine(s) appartiennent les MFs?

Mes réponses:

- À l'informatique **et** au génie logiciel, pas aux mathématiques

---

La vraie question:

- Dans quelle optique doit-on enseigner les MFs dans nos cours?

Ma réponse:

- Dans une perspective génie logiciel, en mettant l'accent sur les spécifications

Deux façons d'étudier/enseigner les méthodes formelles:

Informatique: Quels sont les fondements théoriques des méthodes formelles? Comment peuvent-elles être automatisées? Quels sortes d'outils peuvent être développés et comment peuvent-ils être mis en oeuvre? etc.

Génie logiciel: Comment peut-on utiliser les méthodes formelles pour développer des logiciels fiables et sûrs? Pour quels genres d'applications sont-elles utiles? Quels sont leurs bénéfices et/ou désavantages? etc.