

Patrons de conception prouvés

Thierry Lecomte¹ Dominique Méry² Dominique Cansell²

(1) ClearSy, 320, avenue Archimède, Les Pléiades 3 Bât A 13857 Aix en Provence, France.

(2) Loria, Université Henri Poincaré Nancy 1 .

thierry.lecomte@clearsy.com dominique.{mery,cansell}@loria.fr

Résumé

Les concepteurs peuvent aujourd'hui justifier leurs décisions de conception à partir de patrons qu'ils ont appliqués. De même que la qualité d'un code source peut facilement être jugée par un regard rapide aux normes de codage qui ont été suivies, la qualité d'une conception peut maintenant être estimée d'après les patrons utilisés. De bons outils de conception logicielle embarquent une expertise de conception, sous une forme réutilisable, de patrons instanciables.

Ces concepts sont transposables dans le domaine de la conception de système, en particulier selon le point de vue de leur vérification et de leur validation. Le raffinement est un outil d'ingénierie des systèmes corrects. Toutefois, cet outil, comme bien d'autres, nécessite une utilisation experte pour être efficace. Une telle expertise doit être cataloguée et réutilisée, quand cela est possible, sous la forme de patrons de conception. Ces patrons doivent cibler la vérification et la validation formelle, afin de fournir des outils de haut niveau dont l'usage permettrait de justifier l'atteinte d'objectifs de sécurité et de confiance. Construire de tels patrons nécessite des innovations théoriques.

Nous pensons que les patrons de conception formels ou les patrons de conception prouvés vont être amenés à jouer un rôle vital dans la mesure, basée sur la vérification, de l'atteinte d'objectifs de sécurité et de confiance. Les patrons de conception prouvés ont une caractéristique très intéressante : ils sont construits d'après un objectif de validation des objets conçus et capturent l'information orientée preuve du système en cours de construction.

Des premiers résultats ont été obtenus lors de l'étude de systèmes réactifs (INRS) et de la certification EAL5+ de politiques de sécurité de cartes à puces..

1 Introduction

On sait depuis longtemps que la modélisation d'un système complexe ne peut s'accomplir en une seule étape. Dans la pratique la démarche suivie lors d'un développement de logiciels est incrémentale et progressive. On part d'une description informelle puis peu à peu, on construit une spécification par adjonction de nouveaux éléments. Il est possible dans cette démarche de réutiliser des éléments existants ou encore d'effectuer des retours arrière afin d'intégrer les erreurs ou les oublis de spécification. Une construction ne se fait jamais en un seul jet mais plutôt par différents va-et-vient.

Le projet RIMEL (Raffinement Incrémental de Modèles événementiels), qui a débuté en 2007, concerne principalement le raffinement de modèles événementiels et la systématisation de cette technique dans le cadre d'applications ciblées notamment la conception d'algorithmes ou systèmes répartis. La systématisation de cette technique repose sur le développement de schémas conceptuels appelés patrons de développement s'appuyant sur une validation par la preuve (patrons de développement prouvés). Nous présentons dans cet article la notion de patrons de conception prouvés, utilisant la méthode B, tels que définis par [Abrial 2006].

Le chapitre 2 présente succinctement le raffinement avec la méthode B et ses applications pratiques. Le chapitre 3 décrit les travaux préliminaires concernant les patrons de conception prouvés. Nous concluons la discussion par l'énoncé des travaux qui seront réalisés dans le cadre du projet RIMEL.

2 B, le raffinement et la preuve

Le concept de raffinement, bien connu dans le cadre des méthodes formelles de conception de systèmes, traduit et rationalise une telle démarche. Il permet d'introduire graduellement les détails d'un cahier des charges et de ce fait de concrétiser les modèles afin de s'orienter vers une implémentation, tout en préservant la correction des propriétés validées par rapport aux spécifications intermédiaires. Des preuves de correction doivent être effectuées. B est une méthode qui permet de spécifier et concevoir des logiciels pour lesquels ces preuves de correction sont réalisées tout au

long du développement, en apportant au développeur la possibilité de décomposer sa spécification en éléments plus petits. Cette structuration des modèles associée au raffinement (données, algorithmes) permet d'introduire la complexité du logiciel de manière graduelle, de manière à ce que le logiciel soit prouvé pas à pas au cours de sa construction. Le modèle implantable, partie intégrante de cette modélisation, est alors prouvé et ne nécessite plus de test fonctionnel additionnel. Dans ce type de modélisation, on sépare les propriétés (le quoi) et le comportement codé (le comment), en vérifiant d'abord que les propriétés de la spécification sont cohérentes (il n'y a pas de contradiction) puis en validant le fait que le comportement codé ne contredit pas sa spécification. Par transitivité sur l'arbre de décomposition du modèle, on obtient un modèle implantable qui ne contredit pas sa spécification la plus abstraite (la racine de l'arbre de décomposition). Le formalisme de modélisation utilisé s'appuie sur la théorie des ensembles, la logique des prédicats du premier ordre et le langage des substitutions généralisées.

Pour permettre de modéliser non plus des logiciels mais des systèmes (clos) au sens le plus large du terme, le langage B événementiel a été défini et instrumenté [Rodin]. Ce langage, proche syntaxiquement du langage B logiciel, permet de modéliser des systèmes par l'intermédiaire de variables d'état, affectées de propriétés et modifiées lors de déclenchement d'événements. Un événement, de nature atomique et sans durée, est un couple condition/action : la condition est un prédicat qui, si elle est vérifiée, peut conduire à l'exécution de l'action (modification des variables d'état du système). Le déclenchement d'un événement parmi plusieurs ayant leur condition d'exécution vérifiée est non déterministe. Le B événementiel ne dispose pas aujourd'hui de l'outil de décomposition de modèles tels qu'existant dans le B logiciel.

Quelle que soit le formalisme retenu (logiciel ou système), l'activité de raffinement (transformation d'une spécification abstraite en implantation) est une activité manuelle guidée par l'utilisateur et sanctionnée par les outils de preuve : un mauvais raffinement est synonyme d'obligations de preuve qu'il est impossible de démontrer. L'exception notoire est constitué par l'outil de raffinement automatique développé par Siemens Transportation Systems et utilisé notamment pour la génération des automatismes de sécurité pour la ligne 14 du métro parisien. Cet outil, constitué d'un moteur d'application de règle et de règles de raffinement, permet, pour une typologie de modèles déterminée, de transformer automatiquement un modèle abstrait en un modèle concret. Par exemple, le prédicat d'appartenance d'un élément à un ensemble

$$X \in ENS$$

sera raffiné en utilisant une fonction booléenne associant à un chaque élément typé la valeur TRUE ou FALSE

$$f(X) = TRUE$$

Le raffinement manuel est utilisé dans des cadres variés, par exemple pour produire des modèles formels de politiques de sécurité de cartes à puce certifiées EAL5+ (selon les Critères Communs) et démontrer la validité fonctionnelle d'un système de contrôle/commande de porte palières [Sabatier].

3 Les patrons de conception prouvés

Un patron de conception est un ensemble de variables, d'événements et d'invariants qui vont permettre de modéliser une notion que l'on veut voir figurer dans le système que l'on construit. Un patron de conception aura aussi recours à des constantes et des propriétés relatives à ces constantes. Ces éléments de modèle seront directement insérés dans le modèle du système en cours de construction, après instanciation des éléments du patron de conception (variables, invariants, constantes propriétés).

Un patron de conception est prouvé lorsque les événements décrits dans ce patron vérifient les propriétés invariantes exprimées sur ces variables. De ce fait, n'importe quelle instanciation correcte du patron de conception va conduire à la production d'éléments de modèle dont la correction est acquise. Dans les faits, la démonstration de correction du modèle produit est la charge des outils de preuve, mais nous savons de fait que le modèle à démontrer est juste, ce qui est un gain significatif.

Cette approche de modélisation par utilisation de patrons de conception prouvés se distingue de l'approche faisant appel au raffinement automatique décrite précédemment. En effet, le raffinement automatique est un processus itératif, majoritairement automatisé, qui utilise un moteur de règles et des règles de raffinement « génériques ». Ce moteur de règles ne réalise pas de vérification de correction des règles qu'il tente d'appliquer. Une règle de raffinement peut aussi être mal adaptée au modèle en cours de traitement et nécessiter une réécriture spécifique, avant toute application. La validité du modèle obtenue sera déterminée lors de la phase de preuve du modèle. Cette validité peut être impossible à obtenir du fait de l'inadéquation/fausseté des règles de raffinement appliquées. Il va falloir alors revenir sur le processus de raffinement, identifier la ou les règles en cause, créer de nouvelles règles de raffinement puis relancer le processus de raffinement automatique. Cette approche est clairement moins efficace que l'approche par patrons de conception prouvés, où l'on sait de fait que les éléments de modèle produits sont corrects et peuvent être démontrés par preuve.

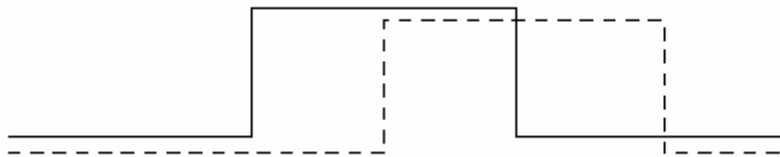
La suite de ce chapitre présente des exemples de patrons de conceptions prouvés. La démarche de création et de validation de ces patterns sera présentée ultérieurement dans un autre article.

3.1 Exemples de patrons de conception

Nous produisons ci-dessous quelques exemples de patrons de conception prouvés qui ont été élaborés dans [Abrial 2006]. Bien qu'ils aient été développés dans le cadre d'un système de sécurité à base d'automatismes, leur portée est plus large.

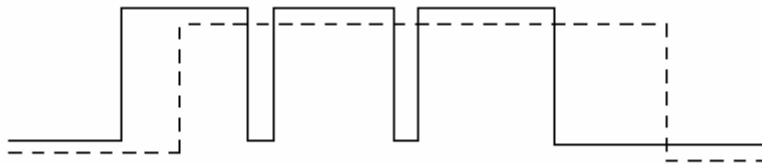
3.1.1 Synchronisation faible

Il s'agit d'un patron simple d'action et réaction simple sans rétroaction qui est présenté sur le diagramme suivant. Nous supposons avoir une action a , représentée par un trait continu, suivie d'une réaction notée r , représentée par un trait pointillé.



Nous notons que r survient après a . En d'autres termes, r passe à l'état haut après que a y soit passé. De manière similaire, r passe à l'état bas une fois que a soit passé à l'état bas.

Il est possible qu'une fois que r est à l'état haut, a passe successivement à l'état bas puis à l'état haut, éventuellement plusieurs fois, alors que r n'est pas capable de réagir à ces variations rapides : r reste à l'état haut tout le temps.



Il est aussi possible que a passe plusieurs fois à l'état haut puis à l'état bas, sans que r puisse y réagir et reste à l'état bas.



Tous ces cas vont être traités par un même modèle où nous avons une synchronisation faible entre l'action et la réaction. Un compteur appelé ca est associé à a , et un compteur cr est associé à r . Ces compteurs indiquent le nombre de fois où action et réaction sont passés à l'état haut (le nombre de fronts montants).

Le patron « synchronisation faible » est défini par 6 propriétés invariantes (0 correspond à l'état bas de a et r , 1 à l'état haut).

pat0.1:	$a \in \{0, 1\}$
pat0.2:	$r \in \{0, 1\}$
pat0.3:	$ca \in \mathbb{N}$
pat0.4:	$cr \in \mathbb{N}$
pat0.5:	$cr \leq ca$
pat0.6:	$r = 0 \wedge a = 1 \Rightarrow cr < ca$

Il est aussi défini par 4 événements correspondant au changement d'état de a et de r . Il est à noter que nos événements constituent un système causal et que donc les événements correspondant au passage simultané de a et r à l'état haut ou à l'état bas ne font pas partie de notre patron. L'événement de passage à l'état haut de a est appelé a_on , le passage de a à l'état bas est appelé a_off . La même convention est adoptée pour r .

```

a_on
when
  a = 0
then
  a := 1
  ca := ca + 1
end

```

```

a_off
when
  a = 1
then
  a := 0
end

```

```

r_on
when
  r = 0
  a = 1
then
  r := 1
  cr := cr + 1
end

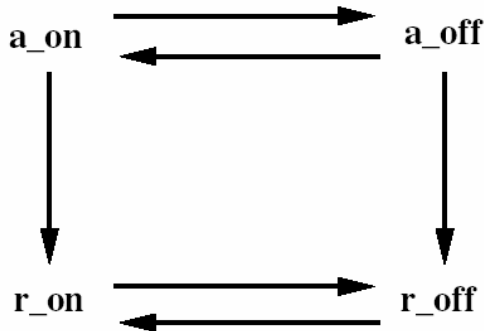
```

```

r_off
when
  r = 1
  a = 0
then
  r := 0
end

```

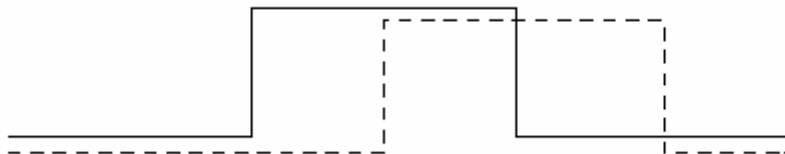
Nous obtenons le schéma de transition ci-dessous. Les flèches indiquent les relations de précédences entre événements (par exemple, l'événement suivant a_on est soit a_off , soit r_on).



Ce modèle est complètement démontré par preuve mathématique.

3.1.2 Synchronisation forte

Il s'agit d'un patron simple d'action et réaction simple avec rétroaction. Il est comparable au patron précédent, mais nous n'autorisons pas les cas décrits par les diagrammes. Le seul comportement autorisé est présenté sur le diagramme ci-dessous :



Le patron « synchronisation forte » est défini par 6 propriétés invariantes :

```

pat0_1:  a ∈ {0,1}
pat0_2:  r ∈ {0,1}
pat0_3:  ca ∈ ℕ
pat0_4:  cr ∈ ℕ
pat2_1:  a = 1 ∧ r = 0 ⇒ ca = cr + 1
pat2_2:  a = 0 ∨ r = 1 ⇒ ca = cr

```

Il est aussi défini par 4 événements :

```

a_on
  when
    r = 0
    a = 0
  then
    a := 1
    ca := ca + 1
  end

```

```

a_off
  when
    r = 1
    a = 1
  then
    a := 0
  end

```

```

r_on
  when
    r = 0
    a = 1
  then
    r := 1
    cr := cr + 1
  end

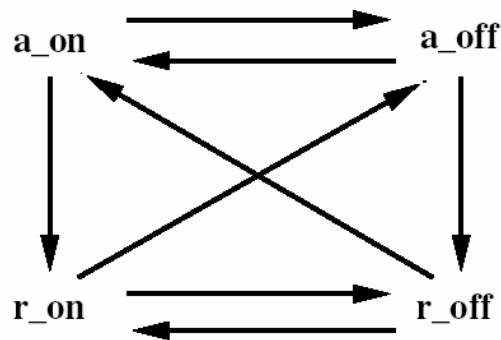
```

```

r_off
  when
    r = 1
    a = 0
  then
    r := 0
  end

```

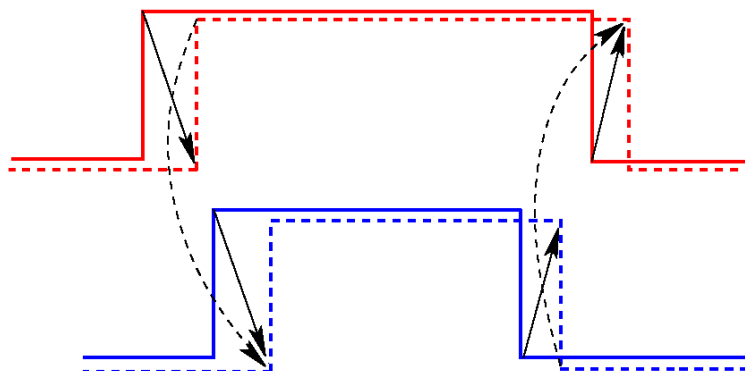
Nous obtenons le schéma de transition ci-dessous.



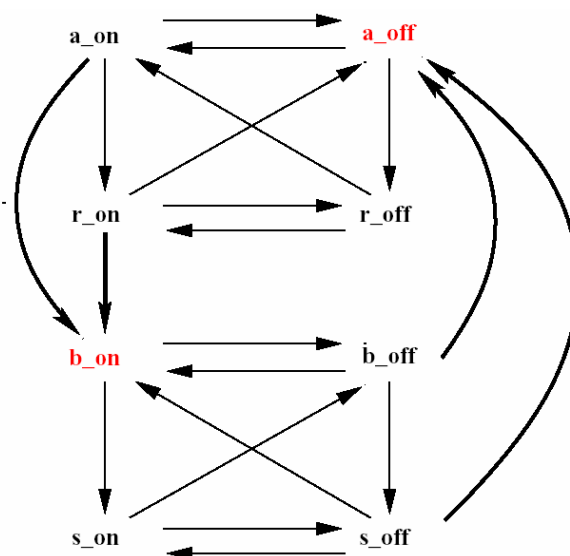
La correction de ce modèle est démontrée par preuve mathématique.

3.1.3 Autres patrons

De manière similaire, des patrons correspondant à la synchronisation faible de réactions fortes et à la synchronisation forte de réactions fortes ont été définis. Ils permettent de corréler, de manière équivalente aux deux patrons définis précédemment, deux couples d'action/réaction, notés ici a,r et b,s . Il est alors possible de définir des relations de causalité entre la réaction r (trait pointillé rouge) et la réaction s (trait pointillé bleu). La synchronisation forte consiste alors à spécifier que pour tout passage à l'état haut (resp. état bas) de a,r correspond un passage à l'état haut (resp. état bas) de b,s . La synchronisation faible spécifie que la survenue d'action/réaction a,r peut n'avoir aucune influence sur b,s .



Nous obtenons le schéma de transition ci-dessous pour la synchronisation faible de réactions fortes.



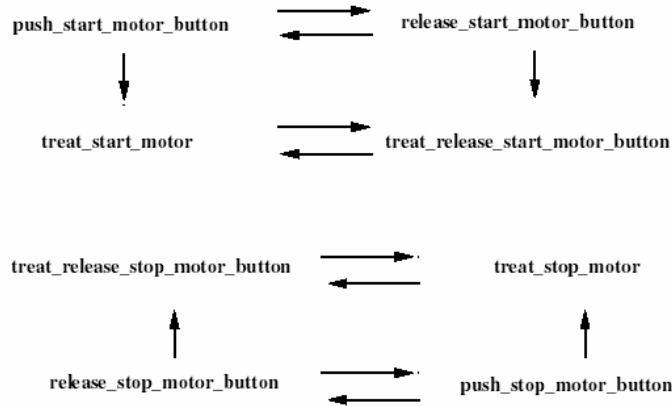
3.2 Application pratique des patrons de conception

Dans [Abrial 2006], la mise en œuvre de ces patterns est présentée dans le cas pratique d'un système de contrôle commande d'une presse [LAMY 2005]. L'analyse a posteriori de l'étude a permis de déterminer les patrons de conception présentés ci-dessus et de les prouver.

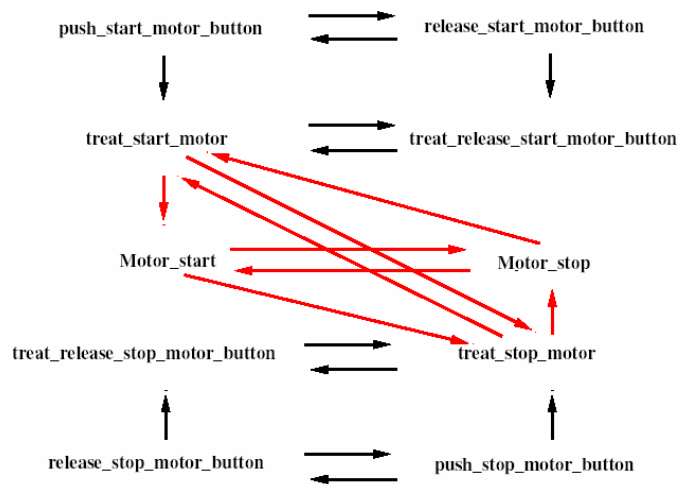
Ces patrons de conception sont appliqués pour créer de nouveaux éléments de modélisation (variables, événements) et enrichir les propriétés invariantes. Par exemple, le patron « synchronisation faible » peut être instancié avec de nouvelles variables et événements tels que décrits ci-dessous. Les paramètres du patron sont : le nom des événements correspondant au passage à l'état haut et bas de l'action et de la réaction (*a_on*, *a_off*, *r_on*, *r_off*), le nom des variables d'état représentant l'état d'activation de l'action et de la réaction (*a*, *r*), les valeurs des variables codant l'état haut et l'état bas des variables d'activation (0, 1).

<i>a_on</i>	≈	push_start_motor_button
<i>a_off</i>	≈	release_stop_motor_button
<i>r_on</i>	≈	treat_start_motor
<i>r_off</i>	≈	treat_release_start_motor_button
<i>a</i>	≈	start_motor_button
<i>r</i>	≈	start_motor_impulse
0	≈	ko
1	≈	ok

Un ou plusieurs patrons peuvent être appliqués une ou plusieurs fois pour un niveau de modèle donné, produisant autant de nouveaux éléments de modélisation. Dans notre cas, le résultat consiste en des synchronisations indépendantes ou à des synchronisations combinées (à droite).



Ils peuvent servir à enrichir des éléments de modélisation existants. Les paramètres passés lors de l’instanciation correspondent à des éléments de modélisation existants. Les événements existants impliqués dans la patron voient leur garde renforcée et leur action (substitution) complétée, alors que de nouveaux invariants sont ajoutés au modèle. Dans notre cas, le résultat consiste en des synchronisations combinées.



A l’issue des travaux d’analyse des modèles existants et de la définition de patrons de conception, la modélisation complète du cas d’étude a été reprise en s’appuyant fortement sur ces patrons et a permis d’en simplifier la conception ainsi que la preuve. En effet, la démonstration des patrons de conception, lors de leur définition, ainsi que la démonstration des modèles utilisant ces patrons de conception pour la génération du code du contrôleur ont toutes été réalisées de manière entièrement automatique avec les outils de preuve de la plateforme Rodin.

4 Conclusion

Le raffinement est une technique permettant de maîtriser la conception incrémentale d’un système mais le passage à l’échelle de cette technique nécessite des techniques permettant de réutiliser ou de rejouer des développements partiellement. La composition de tels développements est aussi un enjeu très important et une retombée sur laquelle nous pouvons nous engager. Il est important de mieux intégrer la preuve dans le processus de conception et de validation de ces systèmes. De ce point de vue, nous pensons apporter des éléments méthodologiques au travers des patrons de conception fondés sur la preuve. Dans le cadre du projet RIMEL, nous proposons de définir une bibliothèque de schémas de conception par raffinement, ainsi que développer et mettre en œuvre les outils associés (éditeur de patron, moteur d’application) qui permettront d’industrialiser cette approche. Ces développements, complétés par un cadre de développement guidé et fondé sur la preuve, devraient contribuer à réduire le coût du développement d’applications à logiciel prépondérant, dans la mesure où il suffira de rejouer et de s’appuyer sur des outils de transformation pour produire les systèmes logiciels.

Remerciements

Ce travail bénéficie d'une aide de l'Agence Nationale de la Recherche portant la référence ANR-06-SETIN-015 (2007-2009)

Références

Abrial, J.R. (2006), *Practical System Modelling. Example: a Mechanical Press Controller*

Sabatier, D. & al, Utilisation de la méthode formelle B pour un système SIL3, Lambda Mu 15

Lamy, P., Blaise, J.C. (2005). *Utilisation de méthodes formelles*, Jautomatise n°42 Septembre-Octobre 2005

Rodin. *Plateforme* <http://sourceforge.net/projects/rodin-b-sharp>

Stouls, N & al (2005). Modélisation et raffinement d'une politique de sécurité, Potesta