J3eA, Journal sur l'enseignement des sciences et technologies de l'information et des systèmes,
Volume 4, Hors-Série 2, 26 (2005)
DOI : http://dx.doi.org/10.1051/bib-j3ea:2005726

Programmation de systèmes réactifs et temps réel ludiques

L. Zaffalon [1], M. Vinckenbosch [1] et P. Bréguet [2]

[1] Laboratoire d'informatique industrielle
École d'ingénieurs de Genève (EIG/HES-SO)
4 rue de la Prairie
CH-1202 Genève, Suisse

[2] Laboratoire temps réel
École d'ingénieurs du Canton de Vaud (EIVD/HES-SO)
1 route de Cheseaux
CH-1401 Yverdon, Suisse

Programmation de systèmes réactifs et temps réel ludiques

L. ZAFFALON*, M. VINCKENBOSCH, Laboratoire d'informatique industrielle, Ecole d'ingénieurs de Genève (EIG/HES-SO), 4, rue de la Prairie, CH-1202 Genève.
zaffalon@eig.unige.ch , vinckenbosch@eig.unige.ch
P. BREGUET, Laboratoire temps réel, Ecole d'ingénieurs du Canton de Vaud (EIVD/HES-SO), 1, route de Cheseaux, CH-1401 Yverdon.
pierre.breguet@eivd.ch

Résumé :

L'une des principales activités des laboratoires d'informatique industrielle [24] (EIG) et temps réel [25] (EIVD) de la Haute école spécialisée de Suisse occidentale (HES-SO) est constituée par la formation dans les domaines de la programmation concurrente et temps réel d'une part et de la programmation de systèmes réactifs d'autre part. Dans ce contexte, une approche méthodologique alliant des moyens ou des objets tant logiciels que matériels, pouvant être considérés comme ludiques, a progressivement été mise en œuvre avec succès auprès des étudiants.
Des simulateurs et des maquettes de trains ont ainsi été développés. Ils permettant à tout étudiant de concevoir des applications concurrentes et temps réel de nature complexe. Quant à l'enseignement plus spécifique des systèmes réactifs synchrone, il est notamment basé sur l'usage de briques LEGO® Mindstorms™. Plus récemment, des développements ont porté sur la mise en œuvre de platines de jeu de type *Game Boy* Advance. Il est à relever que la plupart des développements sont également le fruit de travaux d'étudiants, dans le cadre de projets de laboratoire ou de diplôme.

Mots clés : programmation concurrente et temps réel, programmation synchrone, systèmes réactifs, systèmes embarqués, simulation.

1 INTRODUCTION

La formation d'ingénieurs HES (Hautes écoles spécialisées / *Fachhochschulen* / *University of Applied Sciences*) est de type universitaire. Mais elle présente la particularité d'être orientée vers la pratique. Par ailleurs, une partie des étudiants qui fréquentent les cursus de formation est issue de la formation dite *duale* (études partiellement en entreprise). En soulignant ces deux faits, il en découle la nécessité d'amener progressivement les étudiants vers des réalisations pratiques finalisées.

Fig. 1 : Simulateur de maquette de trains (type C).

Sans perdre de vue les objectifs fondamentaux de la formation, nous avons pu constater que l'usage de sujets ou de supports de nature quelque peu ludique avait un impact non négligeable, à la fois sur la motivation des étudiants et sur la productivité méthodologique. Cette dernière est d'autant plus importante que les langages ou les environnements de programmation sont également appropriés. Il va de soi que les étudiants sont aussi amenés à développer des applications plus arides.

2 SIMULATEURS ET MAQUETTES DE TRAINS MINIATURES

La conduite de trains miniatures constitue un excellent paradigme [8]. Il permet de mettre en œuvre les divers concepts inhérents à la programmation concurrente et temps réel (exclusion mutuelle, gestion des ressources, des priorités, prévention des interblocages, etc.). Dans ce contexte, un simulateur de maquettes (fig. 1) [7] et diverses maquettes réelles (exemple, fig. 2), ayant la même topologie, ont été développés. Le simulateur est écrit en Ada 95 [22] avec *OpenGL* pour la partie graphique [11]. Ada 95 est le formalisme privilégié pour le cours.

Par cette approche, l'étudiant est à même de tester ses programmes, d'abord en simulation, puis sur la maquette réelle correspondante. A cet effet, les paquetages de spécification (*interface*) du simulateur et ceux de toute maquette réelle, également écrits en

Ada 95, sont identiques. Seuls les corps des paquetages (*implémentation*) diffèrent, selon l'exécutif temps réel (embarqué) utilisé: ETS Pharlap [2] [16], MaRTE OS [20] ou *Open Ravenscar* [1]). D'autre exécutifs temps réel embarqués sont en cours d'évaluation, ainsi qu'une mise en œuvre de Java.

Fig. 2 : Maquette de trains miniatures (type C).

Le passage au réel est primordial, il permet de mettre en évidence des problématiques qui sont liées au développement croisé de logiciel enfoui/ embarqué (avec des exécutifs temps réel distincts). Aspects que l'étudiant ignore ou très fréquemment oublie et que le simulateur ne peut mettre en évidence.

Les étudiants apprécient cette illustration, non seulement du bon déroulement des déplacements, mais aussi des inévitables collisions qui reflètent des erreurs de conception ou de réalisation de leurs algorithmes. A noter qu'un éditeur destiné au simulateur de maquettes de trains [19] permet à tout étudiant de définir de nouvelles topologies, mais pour lesquelles il n'existera pas forcément d'équivalent réel.

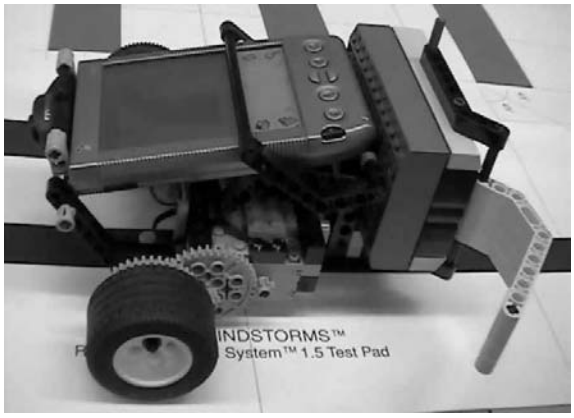

Fig. 3 : Robot mobile piloté par un assistant personnel avec module de vision (comm. infrarouge [18]).

3 ROBOTS LEGO

Les briques LEGO® Mindstorms™ (fig. 5) permettent de mettre en œuvre des notions et des concepts informatiques liés à des disciplines diverses de l'informatique.

- *Algorithmique et programmation*, avec Ada Mindstorm [10] ;

Fig. 5 Brique RCX.

- *Programmation de systèmes*; il est possible de remplacer le *firmware* de base RCX fourni par le groupe LEGO et d'utiliser un exécutif (noyau) multitâche préemptif [5] [17]. Ecrit principalement en C, ce noyau comporte: un ordonnanceur à système de priorités multiples, des sémaphores POSIX 1003b, une gestion dynamique de la mémoire et diverses bibliothèques. Cet exécutif remplace le système d'origine, même si à tout instant il est possible d'utiliser le *firmware* de base, étant donné qu'il se trouve en mémoire ROM. Il est accompagné de bibliothèques permettant le contrôle des moteurs, du port de communication infrarouge et des différents capteurs fournis avec l'ensemble.

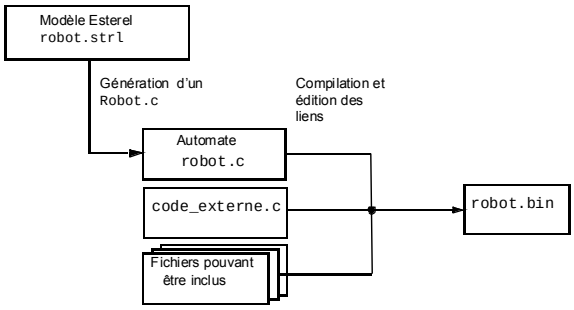

Fig. 4 : Schéma de principe de production de code exécutable.

- *Systèmes réactifs*; c'est notamment le cas dans *notre enseignement* [23]. Que ce soit à partir d'une modélisation synchrone, basée sur les *SyncCharts* (fig. 7) [3] ou sur Esterel (fig. 6) [6], il est possible de générer un automate en C/C++. Dès lors, en utilisant conjointement l'exécutif décrit ci-dessus [4] [15] (fig. 4), il est possible de réaliser des applications réactives

impliquant de tels robots mobiles (fig. 3). Tout comme il est possible d'utiliser les briques de base RCX (fig. 5) pour réaliser d'autres genres de systèmes réactifs (monte-charges, convoyeurs, simulation de systèmes de production, etc.).

```
module ROBOT:
input  CAPT_PRESS_1, DIXIEME;
output VIT_MOT_A: integer,
       VIT_MOT_C: integer,
       DIR_MOT_A: integer,
       DIR_MOT_C: integer;
output LCD: integer;
sensor REFLEXION: integer;
…
signal AFF_SEC: integer in
 [%  Gestion moteurs:
 loop
    await CAPT_PRESS_1;
    emit VIT_MOT_A(VIT_NORMALE);
    emit VIT_MOT_C(VIT_NORMALE);
    emit DIR_MOT_A(AVANT);
    emit DIR_MOT_C(AVANT);
    …
  end loop;
||%  Gestion minuterie:
  every 10 DIXIEME do
     SECONDES := SECONDES+1;
     emit AFF_SEC(SECONDES)
  end every
||% Gestion écran LCD, en alternance:
 loop
    abort
       sustain LCD(?AFF_SEC)
    when 5 DIXIEME;
    abort
       sustain LCD(?REFLEXION) %  Degré.
    when 5 DIXIEME
 end loop
]
…
```

Fig. 6 : Gestion de capteurs et moteurs (extrait Esterel).

A relever que les environnements de *programmation synchrone* utilisés (Esterel, SyncCharts) permettent à la fois la validation par la simulation interactive, la co-simulation (avec des émulateurs) ainsi que la vérification (*model-checking*) du modèle en développement.

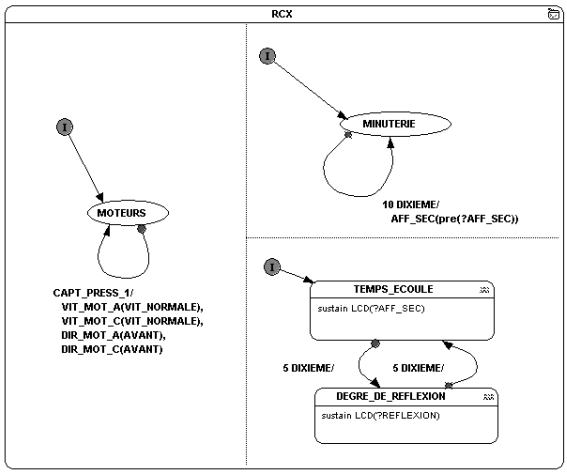

Fig. 7 : Contrôle partiel du module RCX (extrait SyncChart).

4 PLATINE DE JEU GAME BOY ADVANCE

Des applications réactives de divers niveau de complexité peuvent également être exécutées sur des platines de jeu de type *Game Boy Advance* (fig. 8).

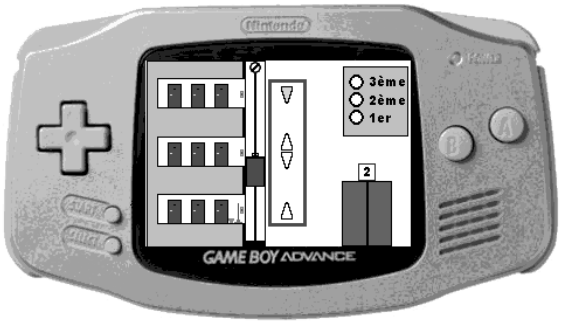

Fig. 8 : Platine GBA.

Une telle platine constitue une bonne plateforme. En effet, elle dispose de plusieurs interfaces: écran couleur, générateur de sons, connexions avec d'autres platines, touches et boutons. De plus, des émulateurs sont disponibles. Une fois le modèle validé et mis au point avec l'émulateur, la version embarquée du code peut être aisément générée et chargée [13].

Des travaux préliminaires ont été réalisés avec l'environnement Esterel Studio [9] et de projets sont en cours avec SCADE [9] (fig. 9). Ce dernier environnement véhicule un formalisme également synchrone, mais à flot de données [12]. Il est largement utilisé dans le développement de logiciel embarqué pour Airbus.

D'un point de vue didactique, nous pensons qu'il est pertinent d'illustrer et de mettre en œuvre des approches de conception distinctes. Il en résulte une meilleure compréhension des concepts sous-jacents

5 CONCLUSION

Les simulateurs, contrairement aux maquettes de trains miniatures, offrent l'intérêt d'être largement diffusables. Il en résulte un gain de temps fort appréciable dans la réalisation des projets, qui doivent néanmoins être validés sur maquette réelle.

Les platines de jeu et, dans une moindre mesure, les robots mobiles constituent des équipements bon marché, que les étudiants possèdent fréquemment. En tout état de cause, ce sont des équipements abordables pour équiper un laboratoire afin de réaliser des applications pouvant être considérées de nature industrielle.

Nous avons pu vérifier que la caractéristique ludique de ces objets comporte sans aucun doute un effet stimulant auprès des étudiants. Cet effet se traduit par la réalisation d'applications de taille bien plus importante par rapport à des énoncés classiques [14]. Tout cela, sans pour autant nuire au sérieux des thématiques enseignées et aux objectifs de la formation.

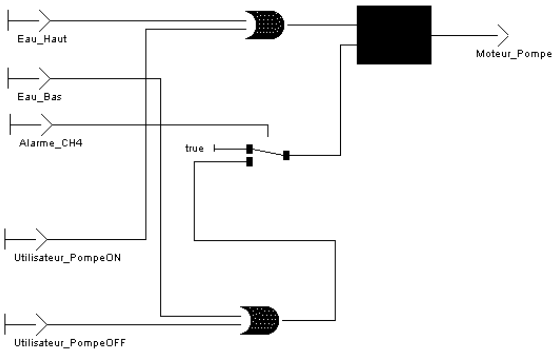

Fig. 9 : Contrôle partiel d'une pompe de puisard de mine simulée (extrait SCADE [21]*).*

Bibliographie

1. M. A. Ajo. *ORK for PC architecture*. http://www.openravenscar.org.
2. ObjectAda. *ObjectAda for Windows 95 and Windows NT*. Aonix, 1997.
3. C. André, H. Boufaïed, S. Dissoubray. *SyncCharts: un modèle graphique synchrone pour systèmes réactifs complexes*. RTS 98, *Conference proceedings*, janvier 1998.
4. E. Basilico. *Exécutif LegOS*. Document interne LII, juin 2001.
5. E. Basilico, F. Carollo. *Commandes pour le chargement de LegOS 2.2.0 et d'une application en C*. Document interne LII, avril 2000.
6. G. Berry. *The Esterel v5 Language Primer. Version 5.91*. Centre de Mathématiques Appliquées, Ecole des Mines and INRIA, 1999.
7. P. Breguet, M. Girardet. *Simulateur de maquette*. Document utilisateur, Polycopié EIVD, version 3.1, janvier 1999.
8. P. Breguet, L. Zaffalon. Railway Scale Model Simulator, in *Proceedings of the Ada-Europe International Conference on Reliable Software Technologies*. Santander, Spain, June 1999, Lectures Notes in Computer Science, No 1622, p. 170-180.
9. http://www.esterel-technologies.com
10. B. Fagin. *Ada/Mindstorms 3.0: A Computational Environment for Introductory Robotics and Programming*. Department of Computer Science, US Air Force Academy, Colorado Springs, CO. www.faginfamily.net/barry.
11. P. Girardet. *Interface graphique pour un simulateur de maquettes de trains*. EIVD, 1997.
12. N. Halbwachs. *Synchronous Programming of Reactive System*. Kluwer Academic Publishers, 1993.
13. O. Loosli. *Commandes pour l'installation et le chargement d'une application Esterel Studio sur Game Boy Advance*. Document interne LII, octobre 2002.
14. J. W. McCormick. *Software Engineering Education: On the Right Track*. http://www.stsc.hill.af.mil/crosstalk/2000/08/mccormick.html.
15. C. Mauras, M. Richard. *How to use synchronous languages to play with the Lego Mindstorms*. http://www.emn.fr/x-info/lego//
16. B. Muller. *Paquetages pour le contrôle d'une maquette de trains depuis une plate-forme cible ETS Pharlap*. Document interne LII, octobre 2002.
17. S. Nielsson. *Introduction to the legOS kernel*. Sept. 2000. http://legos.sourceforge.net/docs/
18. B. Regrain. *Contrôle d'un robot Lego par un assistant personnel*. Mémoire de diplôme HES, Docuent interne LII, décembre 2001.
19. C. Rodriguez. Editeur de circuits destinés au simulateur de maquettes de trains miniatures. Document interne LII, 1999.
20. M. A. Rivas, G. Harbour. *Minimal Real-Time Operating System for Embedded Applications*. http://marte.unican.es/
21. M. von Wyl. *Notice d'introduction à l'environnement SCADE Suite*. Document interne LII, juin 2003.
22. L. Zaffalon, P. Breguet. *Programmation concurrente et temps réel avec Ada 95*. PPUR, 1999.
23. L. Zaffalon. *Programmation synchrone de systèmes réactifs avec Esterel Studio*. Document interne LII, 2002.
24. http://eig.unige.ch/lii/
25. http://ina.eivd.ch/Laboratoires/Ltr/default.htm