La place de l'informatique dans l'enseignement des logiciels et systèmes embarqués*

Florence Maraninchi et Paul Caspi
Laboratoire Verimag (UJF, CNRS, INPG)
http://www-verimag.imag.fr

Avril 2003

Les systèmes embarqués, s'ils ne constituent pas un problème nouveau, prennent néanmoins une importance économique sans cesse accrue : aux systèmes traditionnels de contrôle-commande avioniques, spatiaux, industriels, ferroviaires, etc. et aux systèmes classiques de télécommunications s'ajoute maintenant l'explosion de la téléphonie mobile, des systèmes de paiement, et de l'électronique de consommation courante.

Devant cette importance croissante, le réseau européen ARTIST[1] s'est posé la question de l'enseignement de cette discipline, et nous avons coordonné les travaux du groupe de réflexion constitué à cet effet. Ce sont ces travaux que nous nous proposons de retracer ici. Nous commencerons par esquisser un état des lieux des enseignements et pratiques du domaine, puis nous indiquerons quelles sont nos propositions.

Un état des lieux très dispersé

Sans que l'on puisse l'attester par des comparaisons avec d'autres domaines, comparaisons qui seraient de toute façon difficiles à établir, il nous a semblé néanmoins que le domaine des systèmes embarqués montrait une grande dispersion, tant dans son enseignement que dans ses pratiques.

Des pratiques très diverses

L'examen des pratiques industrielles montre une grande dispersion, au point que l'on puisse dire, sans craindre de se tromper, qu'il y a une informatique aéronautique, différente des informatiques automobile, spatiale, ferroviaire ou télé-communicante.

Pour dire les choses de façon un peu caricaturale, un contrôleur d'avion sera décrit en SCADE-LUSTRE, et un code mono-boucle sera généré automatiquement, puis implanté sur

*Les auteurs remercient Pierre-Claude Scholl et Yassine Lakhnech pour leurs remarques constructives

[1] http ://www.artist-embedded.org/

calculateur nu. Dans le cas réparti et pour la tolérance aux fautes, l'architecture sera de type GALS (globalement asynchrone, localement synchrone). Un contrôleur semblable, pour satellite sera modélisé et simulé en SDL, et codé à la main en tâches ADA qui seront implantées sur un exécutif temps-réel. Dans l'automobile, un contrôleur semblable sera décrit et simulé en MATLAB/SIMULINK, puis codé, soit à la main, soit par un générateur de code automatique associé à cet outil. L'implantation se fera avec un ordonnanceur tabulé. Pour la répartition et la tolérance aux fautes, on utilisera un bus synchrone de type TTA. Dans le ferroviaire, une méthode de conception prouvée à la B sera utilisée, qui aboutira, par raffinements successifs à un code ADA mono-boucle sur machine nue. En revanche, la répartition sera aussi de type GALS. En téléphonie mobile, une modélisation UML sera codée à la main et implantée sur un exécutif temps-réel avec répartition de type GALS.

Comment cela se fait-il ? Comment est-il possible par exemple que des industries ayant des besoins apparemment aussi proches que l'aéronautique et l'espace aient des pratiques si différentes ?

Les réponses tiennent sans doute à l'histoire de chaque domaine d'application, à la fragmentation des enseignements déjà notée et, plus généralement, à la jeunesse et au manque de maturité du thème des systèmes embarqués. Mais, quoi qu'il en soit, cette situation n'est pas bonne. Deux points, parmi d'autres, attestent des problèmes que cela soulève :

- Chaque domaine particulier a son propre espace de choix de solutions et il n'y a pas ou peu de comparaisons inter-domaines et peu de possibilités d'optimisation. De même, les spécialistes circulent peu d'un domaine à l'autre. Même la recherche reste fragmentée.
- Les systèmes sont de plus en plus complexes et mixtes et les problématiques particulières s'interpénètrent ; comment va-t-on alors concevoir des applications où commande et télécommunications coopèrent ? comment les simuler, les tester ? comment va-t-on les vérifier ou les prouver lorsqu'il s'agit d'applications critiques ?

Les propositions que nous faisons dans ce document visent à remédier à cet état de fait.

Des enseignements multiples

Les systèmes embarqués sont rarement un but en eux-mêmes mais, au contraire, peuvent être vus comme des sous-systèmes de systèmes plus importants qu'ils sont chargés de commander et/ou de surveiller. Ils sont donc très étroitement liés à ces systèmes dont ils peuvent être vus comme des appendices, importants certes, mais appendices tout de même. Or les systèmes en question sont d'origines très diverses, mécanique, automatique, aéronautique, ferroviaires, nucléaire, télécommunications, etc. et tous ces domaines font l'objet d'enseignements, plus ou moins organisés en écoles ou facultés. D'autre part, dans les domaines les plus anciens, l'informatique hérite des techniques précédemment utilisées pour réaliser, de façon plus fruste, les mêmes fonctions.

Il est donc naturel que ce soient constitués, dans ces diverses écoles ou facultés, des enseignements d'informatique, adaptés aux besoins de chaque domaine d'application. Au fond, concernant les systèmes embarqués, tout le monde « fait de l'informatique », sous des appellations diverses, par exemple « commande par ordinateur » ou « systèmes temps-réel ». Il y a donc de l'informatique en mécanique, automatique, traitement de signal, aéronautique etc. Le

plus souvent, les visées en sont très pratiques, l'informatique y est vue comme un outil, une technique au mieux, mais rarement comme une science en soi.

Cette situation est-elle satisfaisante ? Sinon, quelle peut être la place de l'informatique en tant que science dans ce panorama et, donc, quels rôles peuvent y jouer les écoles et facultés d'informatique ? Pour répondre à ces questions, il nous faut confronter les formations existantes à l'état des pratiques industrielles dans le domaine.

Comment l'informatique peut-elle remédier à cela ?

Notons tout d'abord, avant d'essayer de répondre à cette question, qu'à notre avis, seule l'informatique est susceptible d'apporter des remèdes à cette situation si dispersée, premièrement parce que c'est la seule discipline dont c'est la vocation et, surtout, parce qu'elle se veut une science et non simplement une technique comme les autres disciplines ont tendance à la considérer. Elle a donc à la fois les moyens intellectuels et le devoir de le faire.

Maintenant, comment peut-elle y parvenir ? Les réponses que nous proposons constituent une ébauche de programme de master (au sens européen du terme) en logiciels et systèmes embarqués. Ébauche car nous ne faisons ici qu'insister sur quelques points clés et ne visons pas à l'exhaustivité. Ce programme vise à former des ingénieurs ouverts au domaine et capables d'explorer l'espace des choix de conception. Mais, selon l'accent mis sur la théorie, il s'adapte aussi à la formation de chercheurs dans le domaine. Enfin, il est aussi pensé dans l'idée que la vie professionnelle de l'étudiant va durer plusieurs dizaines d'années, au cours desquelles le professionnel qu'il va devenir sera soumis à une formation permanente, école maison, formation de vendeurs d'outils, séminaires, etc. Il y apprendra bien des choses, mais sans doute pas les bases difficiles à apprendre et permettant de vraiment comprendre au fond ce qu'on lui enseigne. C'est donc là-dessus que nous voulons insister.

Nos propositions visent à identifier des thèmes fondamentaux, qu'il faut prendre le temps d'enseigner en détail, depuis les bases théoriques, jusqu'à l'étude des grandes classes de mises en oeuvre. Cela nous paraît bien plus important que d'offrir un panorama exhaustif (qui a, de toute façon, bien du mal à le rester) des solutions existantes à un problème particulier.

Plus précisément, nos propositions se structurent selon six axes :

Apprendre les bases de l'automatique et du traitement de signal…

Le point fondamental est ici que, pour que l'informatique parvienne à jouer le rôle qu'on lui assigne, il faut qu'elle s'ouvre aux techniques et fondements des domaines d'application. Apprendre comment ceux-ci formalisent, modélisent, surveillent et commandent les objets physiques auxquels ils s'adressent nous semble un bon moyen de réaliser cette ouverture. En même temps, de tels enseignements paraissent indispensables si nous voulons que nos étudiants interviennent dans les systèmes embarqués. En effet, un système est un tout et on ne peut souvent pas dissocier l'ordinateur du reste. Un informaticien doit savoir qu'une modification mineure d'un point de vue informatique peut avoir des conséquences très importantes pour des propriétés globales (stabilité par exemple) d'un système. D'autre part, il faut savoir que les outils de conception

d'origine automatique comme SIMULINK sont devenus des standards de fait dans de nombreux domaines comme l´électronique automobile et il est clair que les formations sur le tas dans ce domaine ne peuvent apporter les bases nécessaires à une compréhension approfondie de tels outils.

. . . sans oublier la théorie de la programmation

Mais, en même temps, il faut apporter un soin égal à la formalisation du fonctionnement des ordinateurs, ne serait-ce que pour montrer aux étudiants que l'on peut formaliser et raisonner sur les programmes avec autant de rigueur que les automaticiens formalisent et raisonnent sur leurs systèmes. Il faut donc aborder les différentes sémantiques des langages de programmation, les théories de la concurrence, les méthodes de preuve et vérification qui ouvrent la voie à la validation des systèmes critiques.

Par ailleurs, l'étude de la compilation permet de comprendre comment des outils effectifs d'implantation et de validation de programmes sont dérivés de ces bases de sémantique formelle. C'est particulièrement important pour les langages utilisés dans le développement des systèmes critiques, qu'ils soient de modélisation ou de programmation, généraux ou dédiés. La compilation est en quelque sorte *le* domaine de l'informatique où des bases théoriques formalisées ont donné naissance à des outils d'implantation à la fois efficaces et sûrs, largement disponibles, et définitivement adoptés par la pratique industrielle. En ce sens, c'est un excellent d'exemple de l'intérêt d'une formalisation et d'un développement rigoureux de solutions informatiques, de l'expression théorique à l'implantation effective.

Enfin, il n'est pas mauvais d'avoir des aperçus sur les limites de l'informatique, via la calculabilité et la complexité.

Un point intéressant est que la coexistence de ces deux types de cours (bases d'automatique et théorie de la programmation) dans un même curriculum peut suggérer des rapprochements intéressants, vers une modélisation conjointe des systèmes et des programmes.

Étudier plusieurs approches de la programmation temps-réel

Les cours de systèmes temps-réel sont très nombreux et, le plus souvent, constituent la base des enseignements en logiciels embarqués. Malheureusement, ils se placent souvent d'un point de vue limité en ne présentant qu'une seule approche, en général de tradition informatique et fondée sur les théories de l'ordonnancement. Or, nous l'avons vu précédemment, les approches utilisées en pratique sont plus variées et puisent dans plusieurs traditions, langages (ADA, JAVA), exécutifs temps-réel, automatique notamment avec les langages synchrones. Enseigner et faire pratiquer plusieurs approches de programmation temps-réel paraît donc être une façon louable d'élargir le domaine de choix des futurs concepteurs de systèmes. On peut aussi penser que la coexistence de cours de tradition différente peut donner lieu à des rapprochements utiles.

L'algorithmique répartie et ses relations avec la tolérance aux fautes et les architectures de réseaux

La répartition est un point clé des systèmes embarqués, pour des raisons diverses dont, notamment, la tolérance aux fautes. Or, la programmation répartie est beaucoup plus difficile que la programmation centralisée et, au cours des décennies précédentes, deux types de progrès importants ont été accomplis :

- des progrès dans la structuration et la maîtrise de la programmation répartie, comme en témoigne les développements d'Internet,
- et des progrès dans l'algorithmique des systèmes répartis tolérant aux fautes, débouchant sur des résultats fondamentaux qui sont à la base des architectures et bus répartis tels que TTA.

Si le premier point est assez largement connu, le second est moins enseigné car il est le produit d'une communauté de recherche un peu isolée à cheval sur l'algorithmique et la sûreté de fonctionnement.[2] Or il s'agit de résultats (notamment d'impossibilité) fondamentaux qui ont une influence majeure sur les architectures de réseaux embarqués. Il nous semble donc important d'attirer l'attention dessus en ce qui concerne les systèmes embarqués.

Mesurer, évaluer et optimiser les propriétés non fonctionnelles

Les propriétés non fonctionnelles (sûreté de fonctionnement, performances, mais aussi encombrement, ou consommation d'énergie) ont une grande importance pour les systèmes embarqués et, d'autre part, les activités de mesure, évaluation et optimisation ont été, de tous temps des activités de base du métier d'ingénieur. Il est donc important que les étudiants aient des aperçus de ces techniques en même temps qu'ils apprennent les techniques pour assurer et améliorer ces propriétés.

Pratiquer l'architecture de systèmes et les méthodes de développement

Enfin, relier entre eux ces divers corps de savoir pour aboutir à des systèmes cohérents en divers aspects (automatique, programmation, temps-réel, répartition, performances, etc.) est un art difficile, l'art de l'architecte, qui tout à la fois exige de la méthode, des outils, et qui doit se pratiquer. Étudier ces méthodes et les pratiquer est donc important.

Conclusion

Nous pensons qu'un tel programme vaudrait d'être implanté car il serait susceptible de produire des ingénieurs informaticiens bien au fait des problèmes et de leurs solutions, maîtrisant l'espace de choix et adaptables à plusieurs domaines d'application. Ils pourraient être en quelque

[2] Il est symptomatique de cet isolement que des termes aussi fondamentaux que « synchrone-asynchrone » ou « temporisé-non temporisé » n'ont pas le même sens pour cette communauté, qu'en théorie de la concurrence ou dans les langages.

sorte, des spécialistes de l'implantation, sur (réseaux d') ordinateurs, de systèmes de commande, de traitement de signal ou de communication. La recherche pourrait aussi bénéficier de telles formations.

Remplir ce programme, cependant, n'est pas simple, tant il faut d'enseignants qualifiés d'origines diverses. Pour pallier ces difficultés, une solution serait de s'inspirer du réseau suédois ARTES[3] qui sélectionne des cours dans des universités suédoises et organise la mobilité d'étudiants sélectionnés autour de ces cours. Plus près de nous, et sans prétendre au modèle en matière d'implantation, observons qu'à Grenoble, les deux établissements scientifiques (Université Joseph Fourier et Institut National Polytechnique) abordent le problème de manière assez différente. A l'UJF, la mise en place des masters est l'occasion de développer un programme de formation "systèmes embarqués" en 2 ans, ciblé professionnel et recherche, essentiellement défini par les informaticiens. Ceux-ci font appel à des compétences extérieures quand c'est nécessaire. A l'INPG, l'évolution est moins radicale, puisqu'elle s'inscrit dans le cadre d'une collaboration de longue date entre l'ENSIMAG (Informatique et Mathématiques Appliquées) et l'ENSERG (Electronique et radio électricité), qui a conduit à la création du département Telecom en 1999. Pour répondre aux sollicitations du monde professionnel, et tenir compte des implications locales de l'installation du pôle Minatec, l'une des options du département Telecom devient en 2003 l'option "Systèmes intégrés embarqués", avec un programme équilibré entre électronique, conception de circuits, signal d'un côté, informatique de l'autre. La collaboration bien établie entre les équipes des deux écoles a permis de construire un programme théorique et de disposer de plates-formes d'expérimentation.

Certains vont même plus loin et, à la manière des départements « Electrical Engineering and Computer Science » des universités américaines, voient dans ce type de tentatives l´ébauche d'un programme commun tendant à une unification des deux disciplines et une future science des systèmes. C'est, bien qu'encore éloignée, une grande ambition.

[3] http ://www.artes.uu.se/