Introduction aux systèmes embarqués temps réel

Conception et mise en œuvre

Emmanuel Grolleau
Professeur à l'ISAE-ENSMA et directeur adjoint du laboratoire LIAS.

Jérôme Hugues
Enseignant-chercheur à l'ISAE-SUPAERO et responsable de l'équipe IpSC.

Yassine Ouhammou
Maître de conférences à l'ISAE-ENSMA et membre du laboratoire LIAS.

Henri Bauer
Maître de conférences à l'ISAE-ENSMA et membre de l'équipe Temps-Réel du laboratoire LIAS.

DUNOD

Illustration de couverture : Inok/iStockphoto.com

Le pictogramme qui figure ci-contre mérite une explication. Son objet est d'alerter le lecteur sur la menace que représente pour l'avenir de l'écrit, particulièrement dans le domaine de l'édition technique et universitaire, le développement massif du photocopillage.
Le Code de la propriété intellectuelle du 1er juillet 1992 interdit en effet expressément la photocopie à usage collectif sans autorisation des ayants droit. Or, cette pratique s'est généralisée dans les établissements d'enseignement supérieur, provoquant une baisse brutale des achats de livres et de revues, au point que la possibilité même pour les auteurs de créer des œuvres nouvelles et de les faire éditer correctement est aujourd'hui menacée.
Nous rappelons donc que toute reproduction, partielle ou totale, de la présente publication est interdite sans autorisation de l'auteur, de son éditeur ou du Centre français d'exploitation du droit de copie (CFC, 20, rue des Grands-Augustins, 75006 Paris).

DANGER
LE PHOTOCOPILLAGE TUE LE LIVRE

© Dunod 2018
11, rue Paul Bert, 92240 Malakoff
www.dunod.com
ISBN 978-2-10-077842-3

Le Code de la propriété intellectuelle n'autorisant, aux termes de l'article L. 122-5, 2° et 3° a), d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective » et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause est illicite » (art. L. 122-4).
Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles L. 335-2 et suivants du Code de la propriété intellectuelle.

Avant-propos

1 Introduction générale à destination de l'étudiant

Les quatre auteurs de cet ouvrage cumulent quarante ans d'enseignement des cours « informatique embarquée » dans des écoles d'ingénieurs, ISAE-ENSMA, ISAE-SUPAERO, mais ils sont aussi intervenus dans d'autres écoles et universités : SupMéca, Université de Poitiers, Télécom ParisTech, CNAM, Institut Polytechnique de Hanoï (Vietnam), Institut polytechnique de Hô-Chi-Minh-Ville (Vietnam), Université de l'aviation civile de Tianjin (Chine). Ils ont aussi eu l'occasion de donner des tutoriaux sur le thème dans des séminaires et écoles pour jeunes chercheurs, notamment au Georgia Institute of Technology, à l'École Centrale de Nantes, à l'Université de Bretagne-Occidentale, l'INSA de Toulouse, l'École Nationale d'Informatique d'Alger, ou des conférences telles que MODELS ou Embedded Systems Week.

Le choix des formalismes utilisés dans cet ouvrage est issu de nombreuses évolutions de ce cours, débuté il y a trois décennies, et nourri depuis par de nombreuses collaborations de recherche avec des industriels, allant des grands groupes aux PME (petites et moyennes entreprises).

Cet ouvrage s'adresse principalement à des ingénieurs, élèves ingénieurs, ou élèves destinés à effectuer de la conception de systèmes embarqués, et des systèmes cyber-physiques, s'intéressant en particulier à la métrique « temps », donc pouvant s'occuper de la conception de systèmes à contraintes temps réel. La plupart de nos élèves étant plutôt des élèves ingénieurs en mécanique, nous couvrons dans l'ouvrage certains aspects de base de l'informatique et de l'architecture bas niveau des processeurs, en passant par les bonnes pratiques de programmation. Nous avons fait en sorte que l'ouvrage soit accessible, à différents niveaux de lecture, aussi bien à l'expert informatique souhaitant développer un système cyber-physique, qu'à l'ingénieur en mécanique souhaitant découvrir l'informatique embarquée.

La pédagogie de cet ouvrage se veut ascendante : plutôt que de présenter les concepts en long et en large, avant d'aller à leur application, nous avons fait le choix de poser des problèmes, pour introduire au fur et à mesure les concepts nécessaires à leur résolution. Nous utilisons donc des exemples, comme fils conducteurs permettant de présenter les concepts et méthodes les traitant. Il est donc parfois nécessaire d'interrompre le fil du texte pour introduire un concept, ou des informations liées à une technologie : dans ce cas, les auteurs utilisent des encadrés. Voici la sémantique associée aux différents encadrés :

– Encadré **Ressources** faisant référence à des ressources spécifiques liées à l'ouvrage, telles que les codes sources téléchargeables en ligne (marqués par une ligne en pointillés).

© Dunod – Toute reproduction non autorisée est un délit.

– Encadré **Concepts** introduisant des principes ou concepts (sur fond gris).
– `Code informatique` sur fond gris.

Afin de donner un aperçu global de la méthodologie proposée de prime abord, ce qui nous semble souhaitable afin que le lecteur puisse comprendre au mieux les concepts introduits, cet ouvrage est rédigé en deux parties.

La première partie considère un exemple de système très simple, et présente un programme permettant de le contrôler. L'idée est de montrer au lecteur non informaticien, ce qu'est une fonction logicielle, et parallèlement à cela, de montrer au lecteur informaticien ce qu'est un capteur, un actionneur, ou un signal électrique. À partir de cet exemple, nous posons la question : « Comment aurait-on pu spécifier et concevoir le système de contrôle ? », et nous introduisons alors les formalismes de la méthodologie proposée : différents diagrammes Capella, les automates de Harel, et le langage AADL. L'utilisation de ces modèles permet au concepteur de détailler ses choix de conception, et de valider la cohérence d'ensemble de la solution en cours de construction.

Sur l'implémentation purement séquentielle du système, nous constatons qu'un temps considérable est « perdu » du fait de la gestion par scrutation des entrées/sorties notamment. Nous présentons donc les interruptions matérielles, qui historiquement ont été introduites sur les calculateurs pour réduire cette perte de temps. Cela nous permet de présenter une architecture logicielle typique de système de contrôle dit « bare-metal », c'est-à-dire un programme ne nécessitant pas de système d'exploitation. Il faut savoir que, bien que cette architecture logicielle ait commencé à voir le jour dans les années 1960, elle peut encore avoir de l'intérêt aujourd'hui pour certains systèmes de contrôle simples. Nous réappliquons la méthodologie pour aboutir à cette architecture logicielle : d'abord nous montrons que sur l'aspect spécification fonctionnelle, il y a peu de modifications, et que les changements s'opèrent au niveau de la co-conception logiciel/matériel. En observant les faiblesses de cette implémentation uniquement basée sur les interruptions pour le parallélisme, nous montrons comment, historiquement, la programmation cyclique a pu être utilisée pour introduire du parallélisme : toujours sur du « bare-metal », mais en utilisant des techniques très spécifiques, qui ne fonctionnent pas sur tout type de système. Enfin, nous voyons comment l'utilisation de la programmation multitâche permet d'introduire le parallélisme nécessaire au contrôle efficace de systèmes complexes, et nous présentons succinctement comment le langage AADL permet de le représenter. La première partie se conclue par une section d'exercices corrigés. L'objectif de la première partie est d'une part d'introduire des bases de bonne pratique de développement, et surtout de faire comprendre au lecteur ce que le multitâche apporte, les cas dans lesquels l'utiliser, tout en introduisant les formalismes utilisés dans la méthodologie de conception proposée.

La seconde partie suppose donc que le lecteur ait été familiarisé avec les concepts de fonction, tâches, flot de données, et présente, en se basant sur un second exemple fil conducteur, la démarche classique partant d'un ensemble d'exigences et aboutissant à l'implémentation du système, ainsi qu'à sa validation temporelle. Nous y présentons la méthode des rythmes, un guide permettant de choisir la meilleure allocation des fonctions aux tâches en considérant la réactivité du système. Nous abordons ensuite

les aspects de la tolérance aux fautes et montrons comment un système peut être conçu pour continuer à fonctionner, y compris en mode « dégradé », lorsqu'une partie du système est en défaut. Pour l'aspect implémentation, nous avons choisi l'interface POSIX pthread, qui s'est très largement répandue ces dernières années, et s'affiche comme un standard incontournable de la programmation temps réel. Cette interface de programmation est en outre la plus flexible et la plus puissante, offrant la puissance du moniteur de Hoare par exemple, ce qui permet de créer tout outil de communication/synchronisation. Nous montrons ensuite comment l'utilisation d'AADL comme langage de conception de haut niveau permet de faciliter la génération de code, que la cible soit conforme à la norme POSIX, Ada, ou toute autre norme. Puis nous montrons comment les hypothèses effectuées lors de la phase de conception, notamment au sujet du respect des contraintes temporelles, sont vérifiées, en nous appuyant sur la théorie de l'ordonnancement temps réel. Différents exercices corrigés sont alors proposés pour conforter le lecteur dans sa compréhension de cette seconde partie. Enfin, en guise d'ouverture, nous introduisons les concepts de l'ingénierie dirigée par les modèles, et, toujours en nous basant sur l'exemple, nous montrons comment celle-ci facilite la création des langages de modélisation, la transformation de modèles et la génération de code.

2 Introduction générale à destination de l'enseignant

Au fil des années, il nous est apparu comme une évidence que le moyen le plus naturel de concevoir un système pour un ingénieur est une approche descendante, partant de fonctions de haut niveau, qui seront décomposées de façon arborescente jusqu'à des fonctions élémentaires. Différents formalismes ont été utilisés par le passé, comme SART (*Structured Analysis for Real-time*), ou SysML (*Systems Modeling Language*) : ceux-ci permettaient déjà d'opérer une analyse fonctionnelle descendante. Le formalisme retenu dans cet ouvrage est basé sur les diagrammes flots de données offerts par l'environnement Capella, lui-même développé pour instrumenter la méthode Arcadia. Notons que l'ouvrage propose une méthodologie outillée par Capella et AADL, mais n'est pas, même si elle présente des similarités importantes, totalement conforme à Arcadia. Le point commun de ces trois formalismes (SART, SysML, Capella) est que l'on utilise des diagrammes flots de données pour la décomposition fonctionnelle. En effet, si l'on pouvait faire l'hypothèse que tout traitement s'exécute en continu et de façon instantanée, hypothèse qui est généralement faite pendant la phase de décomposition fonctionnelle, alors tout système embarqué pourrait être totalement conçu à base de flots de données. L'exécution de celui-ci contrôlerait le système tel que le concepteur le souhaitait.

La manière dont les systèmes sont conçus a aussi évolué au cours de ces dernières décennies. De façon à diminuer les coûts, on a pu voir des sociétés se spécialiser dans différentes phases, et il est de plus en plus fréquent qu'un système soit spécifié par un donneur d'ordres, décomposé en sous-systèmes, eux-mêmes spécifiés de façon détaillée par un équipementier,

© Dunod – Toute reproduction non autorisée est un délit.

qui sera à son tour donneur d'ordre pour externaliser le développement de certaines fonctions élémentaires, développement ayant lieu souvent en off-shore de façon à en diminuer les coûts. Ce sont donc souvent des équipes pluriculturelles, ne communiquant pas directement, qui sont amenées à joindre leurs efforts pour développer un système de grande envergure. Il est donc primordial qu'une spécification soit bien comprise, et que toute fonction non triviale puisse être exprimée de façon non ambigüe. L'expression formelle du comportement attendu dans la conception de systèmes est un très bon moyen de lever toute ambiguïté. Elle a bien entendu un coût en temps non négligeable, puisqu'il est plus rapide de spécifier une fonction, en exprimant le comportement attendu en langage naturel, plutôt que d'exprimer son comportement par un modèle formel. Cependant, le modèle formel pourra être utilisé lors de la phase de validation formelle, si elle existe pour le système, mais aussi simplement comme outil de test des fonctions développées, voire même pour générer automatiquement le code de celles-ci. L'expression formelle des comportements a pris plusieurs formes dans les méthodes de spécification et conception durant les dernières décennies : de l'utilisation de réseaux de Pétri, aux Grafcets, en passant par l'utilisation de différents formalismes de machines à états. L'un de ces formalismes nous semble particulièrement intéressant pour sa relative simplicité d'utilisation : le statechart, issu des travaux de David Harel, et normalisé par l'OMG comme machine à états d'UML. Le choix de ce formalisme n'est pas un hasard : en effet, il intègre en particulier le concept de régions orthogonales, permettant d'exprimer de façon parallèle le comportement d'un système. Il est ainsi possible de spécifier formellement le comportement d'un système de façon ascendante : le concepteur peut ajouter des détails au fur et à mesure qu'il avance dans l'expression du comportement. Marier ce formalisme ascendant à une méthode de spécification fonctionnelle descendante ouvre de nombreuses possibilités, puisque la méthode de spécification mêlant flots de données et automates de Harel peut être, au besoin, ascendante ou descendante.

L'approche « flots de données » pourrait donner directement l'implémentation d'un système, si l'on pouvait considérer que toutes les fonctions s'exécutent instantanément et en continu. Cependant, cette hypothèse est généralement éloignée de la réalité sur un système embarqué. En effet, plus les ressources de calcul sont importantes, plus l'énergie consommée par celles-ci est importante. Dans la conception de systèmes embarqués, la dernière phase précédant l'implémentation est donc la phase de co-conception logiciel/matériel, pendant laquelle le concepteur est amené à choisir un rythme discret d'exécution de ses fonctions, et qui l'amène à casser la confortable sémantique flots de données utilisée jusqu'alors, dans le but de minimiser les ressources de calcul, et donc d'énergie, à embarquer sur le système. Lorsqu'on parle de co-conception logiciel/matériel, on pense aussi à la conception de matériel. Or, il existe aujourd'hui de très nombreuses cartes électroniques sur étagère. Il nous semble donc que la diversité et le coût réduit de ces plateformes sur lesquelles les systèmes embarqués sont déployés rend relativement rare le besoin de développer sa propre architecture matérielle. Par conséquent, le parti pris de l'ouvrage est de montrer comment déployer le système logiciel sur des ressources matérielles existantes. Les métriques non fonctionnelles, en particulier la réactivité du système de contrôle par rapport au procédé contrôlé, sont très fortement impactées par le rythme choisi pour l'exécution des fonctions. Cependant, le problème général du choix du rythme d'exécution des fonctions pour respecter des contraintes

temporelles de bout en bout est en général un problème complexe. D'autant qu'à ce niveau de la phase de conception, de nombreuses informations manquent. Par exemple, on n'a généralement qu'une idée approximative de la durée d'exécution des fonctions, même prises individuellement, puisque celles-ci n'ont souvent pas encore été déployées sur la plateforme cible. Nous proposons donc dans cet ouvrage une méthode inédite que nous avons mise au point, appelée méthode des rythmes, fournissant un guide au concepteur pour l'affectation des fonctions aux tâches, en se basant sur les contraintes de temps inhérentes aux chaînes fonctionnelles (enchaînement de fonctions partant d'un capteur et allant à un actionneur). Cette méthode, bien que non optimale sur des architectures multicœur, donne un bon guide pour les processeurs monocœur, et reste utilisable dans le cas multicœur. Nous avons retenu pour cet ouvrage le formalisme AADL pour exprimer cette co-conception logiciel/matériel, car il permet de représenter tous les concepts qui nous semblent importants dans cette étape. De plus, il est bien instrumenté, et de nombreux outils ont été développés pour effectuer de l'analyse ou de la génération de code automatique à partir de ce langage.

Le développement d'un système cyber-physique concerne de nombreux domaines : la mécanique, la thermique, l'énergétique, l'informatique, l'automatique, la physique, voire la chimie, etc. Or aujourd'hui, ces domaines sont tellement développés qu'il est rare de trouver un expert de plus d'un domaine à la fois. Chacun utilise ses propres modèles, qui, par définition même du modèle, simplifient une grande partie de la réalité pour se focaliser sur les concepts importants du domaine, en négligeant le reste. Chaque domaine a pu se développer, et sait apporter de bonnes solutions sur son cas particulier, sur ses propres modèles de la réalité. Le développement d'un système cyber-physique aujourd'hui correspond donc souvent à la combinaison de plusieurs briques, chacune traitant du système dans sa globalité, mais uniquement du point de vue de son domaine. À partir de son propre modèle, chaque brique tente donc d'optimiser un certain nombre de métriques, mais en prenant en considération les autres briques de façon extrêmement simplifiée, *via* des contraintes de haut niveau, (typiquement, un automaticien pourra dire à l'architecte système que le correcteur fourni devra s'exécuter à 50 Hz). Les experts de chaque domaine ne sont souvent pas à même de communiquer finement sur les leviers qui pourraient être employés pour leur permettre d'améliorer leur solution (si la contrainte de 50 Hz a un impact important sur l'architecture logicielle, est-ce que 51 Hz est acceptable ? De plus, comment exprimer le fait que les données provenant des capteurs aient un certain délai ? Est-ce que cela pourrait rendre le correcteur instable ?). L'ingénierie collaborative tente de faire tomber les frontières entre les domaines, en permettant aux modèles d'être facilement traduits d'un domaine à un autre. L'un des principaux candidats pour faciliter la « traduction » inter-domaines est l'ingénierie dirigée par les modèles. Dans le domaine des systèmes cyber-physiques, il nous semble que cette technologie est prometteuse pour permettre, à terme, de concevoir un système non pas comme un assemblage d'optimum locaux mais comme un optimum global. C'est pour cela que l'ouvrage présente, en plus des méthodes de spécification, conception, et implémentation, les concepts de base de l'ingénierie dirigée par les modèles. Les formalismes retenus dans l'ouvrage (Capella, AADL, s*tatecharts*) sont outillés par des environnements compatibles avec cette technologie et disponibles gratuitement dans un cadre personnel ou académique.

© Dunod – Toute reproduction non autorisée est un délit.

Table des matières

Préface 1

Partie 1

Introduction à la conception par l'exemple 2

Chapitre 1 Présentation du système exemple 3

1. Montage expérimental 3
2. Programme de contrôle 7

Chapitre 2 Programmation séquentielle 33

1. État des lieux 33
2. Spécification fonctionnelle du système 41
3. Conception de l'architecture logicielle et matérielle 71
4. Étude de l'exécution du système 82

Chapitre 3 Utilisation des interruptions matérielles 84

1. Spécification formelle 84
2. Implémentation par routine de traitement d'interruption 89
3. Conception de l'architecture logicielle et matérielle 93
4. Étude de l'exécution du système 94

Chapitre 4 Exécution cyclique 95

1. Programmation 95
2. Spécification du système 96
3. Implémentation 99

Chapitre 5 Programmation multitâche 101

1. Spécification et conception de l'architecture logicielle et matérielle 102

2. Programmation 102
3. Ordonnancement du système 106
Conclusion de la partie 1 108
Entraînez-vous 111
Solutions 114

Partie 2
Les concepts présentés à travers une étude de cas détaillée 121

Chapitre 6 Spécification fonctionnelle 123
1. Présentation de l'étude de cas 123
2. Analyse opérationnelle 125
3. Analyse système 125
4. Analyse logique 128
5. Spécification formelle par statecharts 132

Chapitre 7 Conception multitâche par la méthode des rythmes 142
1. Durée d'exécution, période et charge processeur 143
2. Du flot de données au multitâche 146
3. Application de la méthode des rythmes au système de contrôle de la mine 155

Chapitre 8 Impacts d'un changement de spécification et tolérance aux fautes 159
1. Erreur *versus* panne 160
2. Activation par événement externe et chien de garde 162

© Dunod – Toute reproduction non autorisée est un délit.

Chapitre 9 Implémentation POSIX avec simulateur 169

1. La norme POSIX 1003.1 169
2. AADL vers POSIX pthread 173
3. Implémentation du système avec simulateur 214

Chapitre 10 Génération automatique de code depuis AADL 228

1. Modélisation des données en AADL 229
2. Modélisation des sous-programmes en AADL 232
3. Modélisation des tâches et exécutif AADL 233
4. Processus, système 236
5. Cas d'étude d'un exécutif AADL : PolyORB-HI/C 236

Chapitre 11 Validation temporelle du système 241

1. Calcul de pire durée d'exécution 241
2. Ordonnancement à priorités fixes aux tâches 246

Chapitre 12 De la conception des systèmes à la conception des modèles : une utilisation avancée des modèles 280

1. Ingénierie dirigée par les modèles 281
2. Modélisation et méta-modélisation 283
3. Langages de modélisation dédiés 291
4. Transformation de modèles et génération de code 294
5. Transformation modèle vers modèle 296
6. Transformation modèle vers texte 302

Entraînez-vous 310

Solutions 315

Index 321

Préface

Nous utilisons (parfois sans le savoir) de nombreux systèmes informatiques. Parmi ces systèmes, certains sont dits « critiques », car leur défaillance peut avoir un impact majeur, par exemple dans les domaines aéronautique, ferroviaire ou encore nucléaire. Ces systèmes sont soumis à des contraintes réglementaires fortes. Il ne s'agit pas uniquement de « coder » les fonctions logicielles nécessaires, il est également nécessaire de démontrer le bon fonctionnement du système aux autorités compétentes. Les dimensions temps réel, embarqués ou encore critiques étant prédominantes. L'architecture du système joue un rôle structurant pour garantir les propriétés attendues.

Au-delà des techniques de conception « classiques » pour systèmes temps réel, industriels et chercheurs ont défini de nouvelles techniques de conception basées modèles, permettant d'appréhender toutes les facettes du système, du besoin fonctionnel jusqu'à son implémentation matérielle et logicielle.

L'Ingénierie basée modèles (*Model-Based Systems Engineering* ou MBSE) sert de base à la méthodologie décrite dans cet ouvrage. Elle est au cœur de l'ingénierie de ces systèmes complexes, car elle permet d'échanger une vision consolidée d'un système, construite collectivement, de procéder à son analyse, voire à sa vérification, et ultimement de construire tout ou partie du logiciel et du matériel.

Le MBSE révolutionne le travail de l'architecte système ainsi que celui des ingénieurs concepteurs de systèmes embarqués. Grâce à cette approche, ils peuvent s'abstraire de documents souvent longs et complexes au profit de modèles avec lesquels il est possible de façonner petit à petit une solution en l'enrichissant, la simulant et la vérifiant. Les approches MBSE sont maintenant couramment utilisées dans tous les secteurs de l'informatique critique, où le logiciel doit répondre de façon fiable aux demandes des utilisateurs, et plus largement de la société en systèmes fiables, sécurisés et innovants. Elles sont déployées au sein du groupe Thales, mais aussi de toute l'industrie des systèmes embarqués.

L'enseignement des technologies MBSE est une nécessité dans la construction de systèmes devenant de plus en plus complexes. Mettre à disposition des ingénieurs et utilisateurs un livre traitant de ce sujet contribue grandement à sa diffusion. Cet ouvrage, rédigé par plusieurs enseignants-chercheurs de l'ISAE-ENSMA et l'ISAE-SUPAERO répond à ce besoin. En suivant une approche didactique, il fournit toutes les briques nécessaires pour la pleine compréhension des outils et méthodes permettant de concevoir et implanter des systèmes temps réel, en confrontant théorie et pratique, et en se fondant sur deux approches complémentaires : Capella et AADL. Nul doute que ce livre saura être utile aux ingénieurs et futurs ingénieurs systèmes embarqués.

Christophe Gatti, Thales — MDE & OSS Expert

© Dunod – Toute reproduction non autorisée est un délit.