Marte, le nouveau standard UML pour les systèmes temps réel embarqués

Sébastien Demathieu, Sébastien Gérard, Frédéric Mallet

Texte intégral – voir 2 Mars 2008 n°189 - Electronique

Marte, le nouveau standard UML pour les systèmes temps réel embarqués.

Sébastien Demathieu (Thales Research and Technology)
Sébastien Gérard (CEA)
Frédéric Mallet (I3S/INRIA)

L'OMG vient d'adopter un nouveau standard qui étend le langage de modélisation UML 2 pour la conception et l'analyse de modèles de systèmes temps réel embarqués. Appelé Marte (Modeling and analysis of real-time and embedded systems), il devrait contribuer à l'élargissement du marché des logiciels dédiés à la conception de systèmes temps réel embarqués, vers des outils génériques basés sur UML 2.

Le langage de modélisation UML (Unified modeling language) apparaît aujourd'hui comme une solution viable pour la conception de systèmes temps réel embarqués. Ce langage, standardisé depuis plus de dix ans par le consortium industriel OMG (Object management group), bénéficie en effet d'une large audience dans l'industrie du logiciel. Il existe notamment un marché conséquent d'outils de modélisation UML (plus d'une trentaine de vendeurs UML sont recensés dans le monde) et de nombreux industriels ont déployé cette technologie dans leurs programmes opérationnels. De plus, un nombre croissant de formations d'études supérieures en informatique en France permet d'apprendre les bases de ce langage, connu désormais de beaucoup d'ingénieurs logiciel et système.

Rappelons qu'UML est un langage généraliste qui couvre toutes les étapes du processus de développement logiciel (la spécification, la conception, l'implantation et le test). La dernière évolution majeure de ce langage, UML 2, permet une modélisation encore plus précise de la structure et du comportement d'un système Elle autorise en fait le passage d'une approche centrée sur le code à une véritable « ingénierie dirigée par les modèles » (ou MDE, Model Driven Engineering en anglais), dans laquelle les modèles deviennent les artefacts essentiels du processus d'ingénierie. Une des particularités d'UML est qu'il peut se « spécialiser » pour des domaines particuliers via le mécanisme dit de « profil ». Un profil UML introduit ainsi de nouveaux concepts pour un domaine spécifique, comme par exemple le temps réel embarqué. Ce mécanisme autorise l'emploi des outils UML classiques et, par voie de conséquence, la réduction des coûts en évitant le développement d'outils de modélisation spécifiques à un domaine d'application. On peut noter que ces extensions, via la notion de profil, peuvent aller bien au-delà des préoccupations classiques liées à la conception de logiciels, comme c'est

le cas avec le profil SysML pour l'ingénierie système [1]. Au-delà, dans le domaine particulier des systèmes temps réel embarqués, la prise en compte des propriétés non fonctionnelles (durée d'exécution, consommation électrique, empreinte mémoire…) est généralement aussi importante que le respect des exigences purement fonctionnelles, comme par exemple effectuer un calcul de trajectoire à partir de données fournies par des capteurs. Ainsi un dépassement d'échéance lors d'un calcul de trajectoire est aussi grave qu'une erreur qui pourrait survenir dans l'algorithme de calcul. Or, si UML offre de nombreux diagrammes pour représenter le comportement et la structure d'un système, il manque de solutions pour modéliser ces aspects non fonctionnels. Plusieurs tentatives ont été proposées pour combler ces limitations et adapter UML au domaine du temps réel embarqué. Ceci à travers des solutions commerciales propriétaires, Artisan d'Artisan Software, Rose RT d'IBM, Rhapsody de Telelogic, ou bien à travers des projets de recherche tels que Accord, Omega ou Gaspard.

L'OMG, de son côté, a travaillé à la standardisation d'un premier profil UML dans ce domaine avec le profil UML pour Scheduling performance time (SPT). La première version de SPT, adoptée en 2001, était destinée à UML 1.4. La spécification définit un ensemble d'annotations en vue de réaliser des analyses quantitatives sur les modèles UML. Mais de nombreuses modifications majeures se sont avérées nécessaires pour aligner SPT avec la nouvelle norme UML 2. C'est pourquoi en 2005, l'OMG a lancé un appel à proposition pour un nouveau standard qui viendrait remplacer SPT. Cette nouvelle norme devait assurer la couverture de tous les aspects, de la modélisation, de la spécification à l'implantation, et de l'analyse, en prenant en compte les aspects logiciels et matériels des systèmes embarqués (alors que SPT ne supportait principalement que l'étape d'analyse des propriétés temps réel de la partie logicielle d'un système). Cette norme, appelée Marte [2] a ainsi vu le jour en août 2007. Le profil UML pour Marte (ou Marte tout simplement) est donc le résultat du travail d'un consortium international regroupant à la fois des éditeurs d'outils UML, des vendeurs d'outils présents sur le marché du temps réel et de l'embarqué, des grands industriels utilisateurs, et des experts du domaine issus du monde universitaire. Profil officiel, Marte peut ainsi compléter n'importe quel outil de modélisation respectant la norme UML 2, via l'ajout de fonctionnalités d'analyse et de conception de systèmes temps réel. Il permet la mise en œuvre, depuis un environnement UML standard, des techniques existantes d'analyse des systèmes temps réel comme des analyses d'ordonnancement (le Rate monotonic analysis par exemple) ou de performance (la théorie des files d'attentes et les réseaux de Petri par exemple), tout en restant ouvert à de nouvelles approches telles que des techniques d'ordonnancement statique ou dynamique basées directement sur l'ingénierie des modèles. Marte est aussi compatible avec d'autres standards du domaine, comme SysML, SAE AADL v1, EAST-ADL v2, Arinc 653, et Posix.

I.- Marte en bref…

➜ **Le nouveau standard OMG Marte, associé à UML 2, rend possible le passage à une véritable « ingénierie dirigée par les modèles » pour les systèmes temps réel embarqués.** Ce profil UML fournit des constructions pour modéliser précisément les propriétés non-fonctionnelles, le temps ainsi que les ressources. Il permet de modéliser des plates-formes d'exécutions matérielles et logicielles, et d'allouer des éléments d'une application temps réel embarquée sur ces plates-formes. Il fournit aussi toutes les constructions pour réaliser des analyses quantitatives (analyse d'ordonnancement ou de

performance) à partir de ces modèles, et ainsi valider en amont la conception du système. Ces extensions sont génériques et n'imposent pas de modèle d'exécution, de technique d'analyse ou de technologie d'implantation particuliers. Marte est donc applicable à une grande variété de méthodologies et processus d'ingénierie. On peut finalement considérer Marte comme étant au domaine du temps réel embarqué ce que UML est au domaine du logiciel : une famille ouverte et extensible de langages de modélisation.

Un langage pour modéliser les propriétés non-fonctionnelles.

Les propriétés non-fonctionnelles sont l'une des principales caractéristiques qui rendent le développement des systèmes embarqués temps réel différent, et bien souvent plus difficile, que celui des systèmes dits à but général. La norme Marte prend en compte cette considération en proposant deux cadres techniques spécifiques : un paquetage générique dédié à la définition des propriétés non-fonctionnelles ; et un paquetage spécifiquement orienté vers le support de la notion du temps. Le paquetage dédié à la gestion des propriétés non-fonctionnelles, appelé NFP pour « Non-functional properties », contient des extensions à UML pour définir ses propres caractéristiques non-fonctionnelles relativement à son domaine d'application. Ce paquetage est complété par un langage de spécification de valeurs, associées à des propriétés non-fonctionnelles, et par une bibliothèque qui définit un ensemble de propriétés non-fonctionnelles couramment utilisées.

Le langage de spécification s'appelle VSL, pour « Value specification language ». La bibliothèque se nomme Marte_Library. Elle se décompose en sous-paquetages qui ++++définissent respectivement un ensemble d'unités de mesure basées sur le système international des unités (puissance, fréquence…), des types complexes comme les matrices et les vecteurs, et un ensemble de définitions de propriétés non-fonctionnelles fondamentales pour caractériser, par exemple, la nature (périodique, sporadique…) des événements reçus par le système. Cet ensemble de concepts, selon les besoins, est utilisable de plusieurs manières. Exemple parmi d'autres, on peut mettre ainsi en œuvre ces extensions pour caractériser le débit et la taille mémoire de plusieurs unités de contrôle d'un système (figure 1).

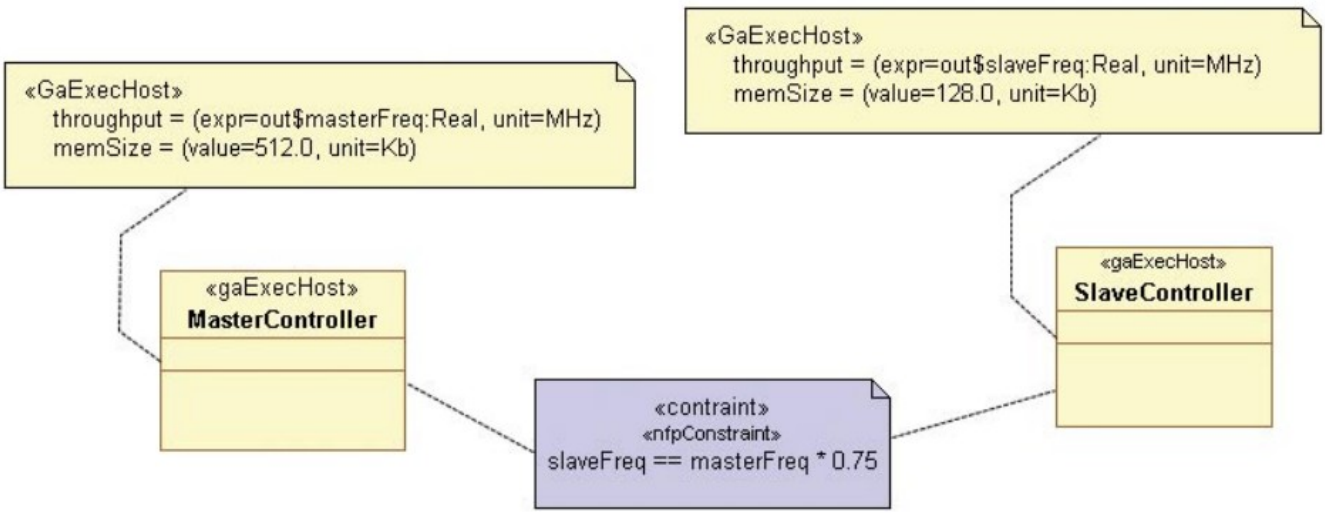

Figure 1: Modélisation de propriétés et contraintes non-fonctionnelles

Grâce aux extensions prévues dans Marte, on peut par exemple caractériser le débit et la taille mémoire de deux contrôleurs d'un système. Ici, les valeurs associées au débit sont représentées par des variables à évaluer. Celles-ci sont utilisées dans une contrainte pour définir une relation entre le débit du contrôleur maître et le débit du contrôleur esclave (cf. Figure 1).

II.- Trois voies d'utilisation pour la partie NFP de Marte

➜ L'ensemble des concepts proposés pour le paquetage NFP (Non functional properties) Marte est conçu pour être mis en œuvre de trois façons différentes :

- En définissant des profils spécifiques dédiés à une problématique particulière. Par exemple, les chapitres de Marte dédiés à l'analyse d'ordonnancement et à l'analyse de performance s'appuient sur le paquetage NFP et sur le langage VSL (Value specification language).
- Dans les modèles utilisateurs pour identifier et caractériser directement certaines propriétés d'un modèle comme étant des propriétés non-fonctionnelles de l'application en cours de modélisation, grâce à l'extension « nfp ».
- En définissant des contraintes non-fonctionnelles qui étendent les contraintes UML classiques. Ces dernières sont caractérisées par l'extension « NFP_Constraint » et portent sur des propriétés non-fonctionnelles du modèle.

Une modélisation précise des aspects temporels

Au-delà, Marte complète sa prise en compte des aspects non-fonctionnels d'une application temps réel embarquée, en définissant un cadre conceptuel très performant et précis qui supporte tous les aspects liés à la préoccupation du temps, une notion essentielle pour ce type d'application. En effet, dans les systèmes à contraintes temps réel strictes, le temps a souvent autant d'importance que la fonction elle-même. Or, modéliser le temps est assez délicat car son interprétation peut varier selon les domaines [3]. UML qui se veut généraliste ne fournit pas de modèle de temps complet ; il est donc considéré comme essentiellement atemporel. La version 2 introduit cependant un chapitre, SimpleTime, qui traite d'aspects temporels simples et conseille de recourir à des profils spécifiques pour des aspects plus complexes [4]. Dans la continuité, le chapitre Time de Marte est une extension qui couvre tous les aspects temporels liés à la conception ou l'analyse de systèmes temps réel embarqués. Il permet de définir la structure temporelle, de choisir des mécanismes de mesure du temps (horloges) et d'associer des informations temporelles (durée, date) à des éléments de modèle (activité, comportement, événement). Il offre également la possibilité de modéliser des contraintes de temps, en précisant s'il s'agit de contraintes de spécification, de contraintes impliquées ou simplement d'observations (figure 2).

La bibliothèque TimeLibrary de Marte contient ainsi une horloge idéale (idealClk) qui représente le temps des lois de la physique ou de la mécanique, dit temps physique. Marte permet aussi de construire des horloges logiques qui mettent en avant des aspects temporels qualitatifs, comme la précédence, ou la simultanéité entre deux événements. Marte supporte en outre la définition de modèles multi-horloges adaptés à une description de haut niveau de systèmes de type Gals (globalement asynchrones et localement synchrones), mais aussi aux systèmes temps réel répartis. Les systèmes répartis sont soumis au problème de la relativité des observations, c'est-à-dire à la difficulté de s'accorder sur la date

d’occurrence d’un événement. Les horloges logiques permettent ici de modéliser ces problèmes. On peut noter que cette répartition n’implique pas nécessairement des réseaux à grande échelle. On la retrouve dans les voitures ou les avions qui contiennent des réseaux embarqués avec plusieurs dizaines de calculateurs répartis, mais aussi dans les architectures de processeurs à plusieurs cœurs où chacun peut avoir son propre domaine d’horloge. L’intégration des systèmes sur puce est telle que, même avec un seul processeur, les temps de parcours de bout en bout ne sont plus négligeables. Un autre aspect traité par Marte est la définition d’horloges multiformes qui permettent d’exprimer des échéances en fonction d’une durée (25 fois par seconde) ou d’une date (avant demain 12 h), mais aussi, et ce de façon mixte, avec d’autres quantités (avant un quart de tour de l’arbre à cames).

Ces horloges logiques peuvent être manipulées grâce au langage CCSL [5] « the Clock Constraint Specification Language » définit en annexe de la spécification OMG.

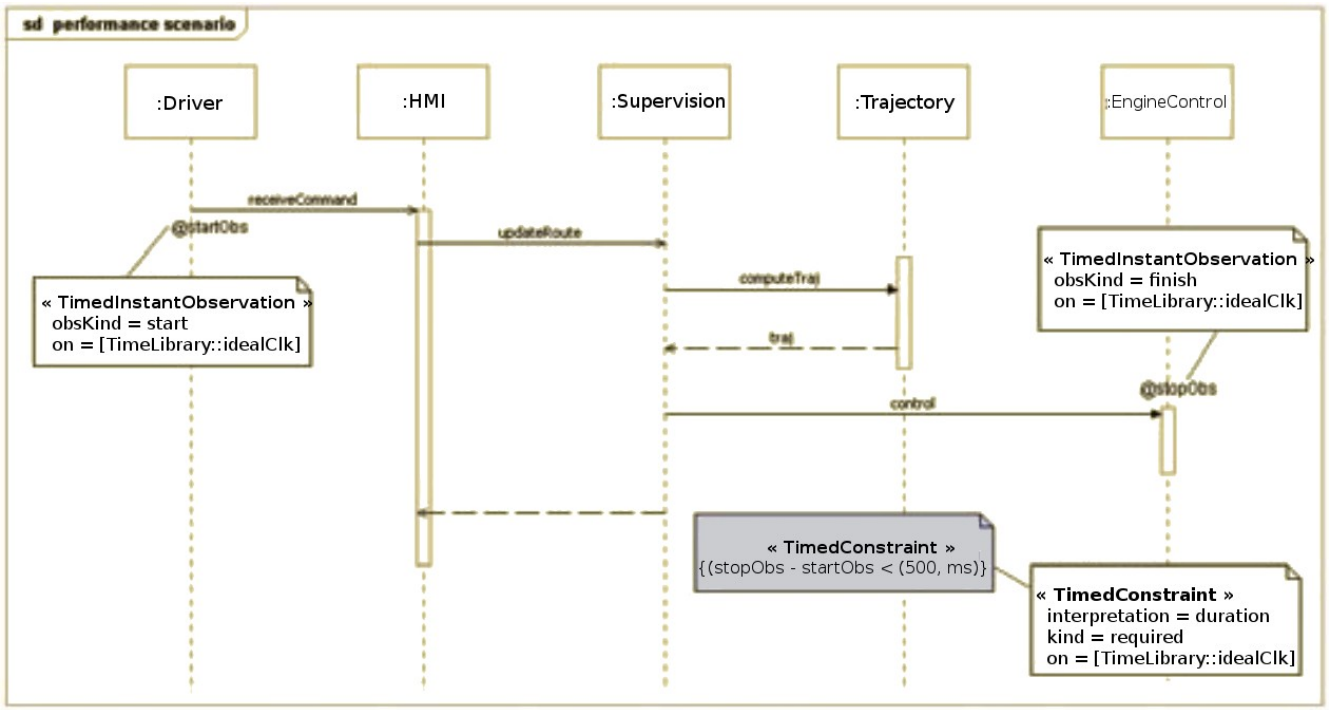

Figure 2: Spécification d'une contrainte de temps dans un scénario de performance

Dans le diagramme d’interactions ci-dessus, on voit un exemple d’annotation temporelle. Les observations définies dans UML portent l’extension « TimedInstantObservation » pour spécifier à quelles horloges elles font référence. Ces observations sont ensuite utilisées dans une contrainte portant l’extension « TimedConstraint » pour spécifier une contrainte de durée.

Des modèles de calcul pour le temps réel

Le profil Marte est donc à même de servir de base à la définition de modèles de calculs spécifiques qui prennent en compte les caractéristiques quantitatives (échéances, période), ainsi que les caractéristiques

qualitatives associées à la gestion de la concurrence et des communications. Marte fournit ainsi un modèle de calcul élémentaire basé sur des unités temps réel, RtUnit (figure 3).

Les RtUnits de Marte sont similaires aux objets actifs d'UML avec des propriétés spécifiques adaptées au domaine du temps réel embarqué. On retrouve des concepts parents dans Room et Rhapsody, par exemple. Une RtUnit est une unité d'exécution autonome capable de traiter des messages de manière concur-rente en respectant les contraintes associées. Elle encapsule dans une même entité les données, le comportement et le contrôle de concurrence.

Les PpUnits, de leur côté, sont des unités passives protégées qui modélisent les données partagées entre unités temps réel selon une politique d'accès précise (exclusion mutuelle, accès gardé, par exemple).

Les unités fournissent un ensemble de services, RtService, caractérisés entre autres par leur mode de communication (appel synchrone, synchrone retardé, asynchrone). Les services sont associés à des comportements repérés par l'extension RtBehavior appliquée sur une machine à états, une activité ou une interaction UML.

Il est à noter que chacun peut adapter Marte à son propre flot de conception et définir son modèle de calcul et de communication. Cette personnalisation doit être effectuée une fois pour toutes et n'est pas nécessairement connue de l'utilisateur final, qui se contentera d'utiliser une bibliothèque.

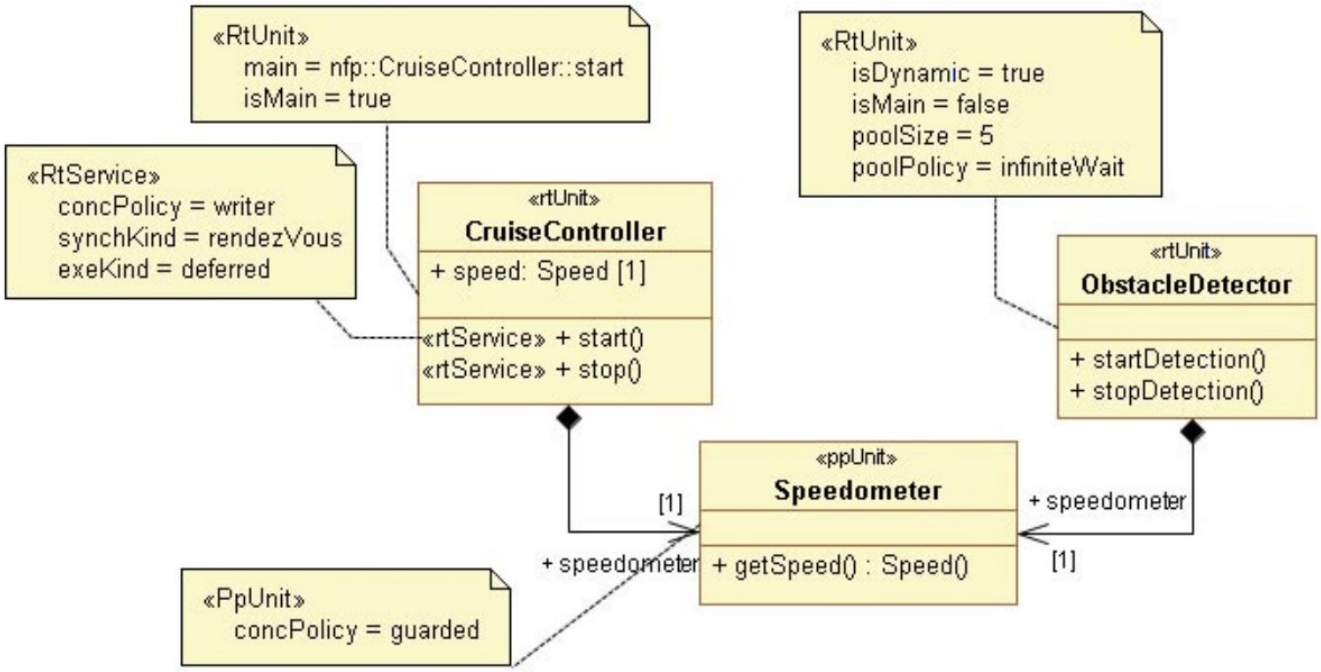

Figure 3: Modélisation d'un système de régulation de vitesse à l'aide d'unités temps réel

La figure ci-dessus montre une utilisation des unités temps réel de Marte sur un système de régulation de vitesse. Celui-ci contient deux RtUnits, le régulateur (CruiseController) étant la principale (isMain=true) et offre deux services. Le compteur de vitesse (Speedometer) est une ressource passive protégée partagée entre le régulateur et le détecteur d'obstacle (ObstacleDetector). Les accès à cette ressource sont gardés.

Modélisation de plates-formes d’exécution

Marte s’inscrit pleinement dans une démarche de type « ingénierie dirigée par les modèles » pour le développement des systèmes temps réel embarqués. Dans ce contexte, un point important de la norme est un support riche et particulièrement bien adapté au développement des applications visées, à la problématique de la modélisation des plates-formes dites d’exécution, ainsi qu’à la description des allocations possibles d’une application sur sa plate-forme d’exécution. Marte propose trois paquetages spécifiques contenant des extensions à UML pour modéliser : des plates-formes d’exécution génériques, (c’est-à-dire de niveau système), des plates-formes d’exécution de type logiciel comme des systèmes d’exploitation ou des machines virtuelles, et des plates-formes d’exécution matérielles.

Le paquetage GRM, pour « Generic resource model » contient un ensemble de ressources génériques pour caractériser les plates-formes d’exécution. On y trouve par exemple des extensions pour modéliser un hôte de calcul (« ComputingResource ») ou encore un hôte de communication (« Communication-Media »). GRM s’appuie sur un « patron » générique de modélisation qui considère qu’une plate-forme est constituée d’un ensemble de ressources (possiblement hiérarchiques), chacune proposant un ensemble de services (au minimum un service). On fait également la distinction entre le concept de type de plate-forme, recouvrant une famille de plates-formes ayant des traits communs, et le concept d’instance, décrivant une plate-forme d’exécution en particulier. Ce paquetage générique est spécialisé en deux paquetages dédiés respectivement à la description des plates-formes d’exécution logicielles et aux plates-formes d’exécution matérielles.

Le paquetage de description des plates-formes d’exécution logicielles supporte principalement la description des plates-formes de type système d’exploitation temps réel telles que Arinc653, Posix ou encore Osek. Par exemple, on y trouve notamment des ressources logicielles de calcul (des tâches ou « threads » au sens Posix) ou encore des ressources d’exclusion mutuelle (le concept équivalent aux sémaphores ou aux mutex présents dans Posix).

Le paquetage de description des plates-formes d’exécution matérielles propose quant à lui des extensions pour décrire, à différents niveaux de détails, les processeurs, les mémoires (cache, Rom, Ram…), ou encore les Asic et les PLD (cf. Figure 4 dans le texte original, ce diagramme décrit une architecture multiprocesseur symétrique (SMP) avec le profil Marte. Le modèle a été réalisé avec l’outil UML 2 Papyrus, disponible, ainsi qu’une implantation de Marte, en Open Source www.papyrusuml.org).

Une fois les plates-formes d’exécution décrites, il faut allouer les éléments de l’application sur ces plates-formes. Pour cela, Marte introduit une relation de dépendance spécifique inspirée de SysML, appelée allocation et dénotée par l’extension « allocate ». Cependant, alors que dans SysML l’allocation permet d’allouer n’importe quel élément de modèle sur n’importe quel autre, dans Marte, seuls les éléments de l’application (identifiés par l’extension « app_allocated ») peuvent être alloués, et seul des éléments de la plate-forme d’exécution (identifiés par l’extension « ep_allocated ») peuvent être la cible de l’allocation. De plus, Marte permet d’associer explicitement à une allocation des contraintes non fonctionnelles qui caractérisent le coût de l’allocation (en termes de consommation

électrique, ou de durée d'exécution). Le coût des différentes allocations possibles est pris en compte pour réaliser une exploration d'architecture et déterminer l'allocation effective (cf. figure 5 dans le texte original).

Une meilleure intégration des techniques d'analyse

La validation des caractéristiques d'un système temps réel embarqué avant les phases d'implantation et d'intégration reste un défi pour les concepteurs. Les techniques mises au point au cours des vingt dernières années dans le domaine de l'analyse d'ordonnancement ou l'analyse de performance proposent des solutions désormais reconnues (que ce soit par analyse statique ou par simulation). Cependant les outils d'analyse ne sont pas totalement exploités aujourd'hui, principalement parce qu'ils ne sont pas intégrés dans le processus d'ingénierie. Les ingénieurs et architectes en charge de la validation du comportement temps réel d'un système ont pourtant tout intérêt à valider la conception d'un système au niveau des modèles. Pour ce faire, ils doivent extraire les informations nécessaires à l'analyse depuis ces modèles, travailler avec des formalismes spécifiques, puis interpréter les résultats d'analyse et leur impact sur les modèles de conception.

Une approche unifiée autour d'UML, dans laquelle les modèles de conception viennent directement alimenter des outils d'analyse, devrait ainsi aider à la réduction des ruptures dans le processus d'ingénierie (cf. Figure 4).

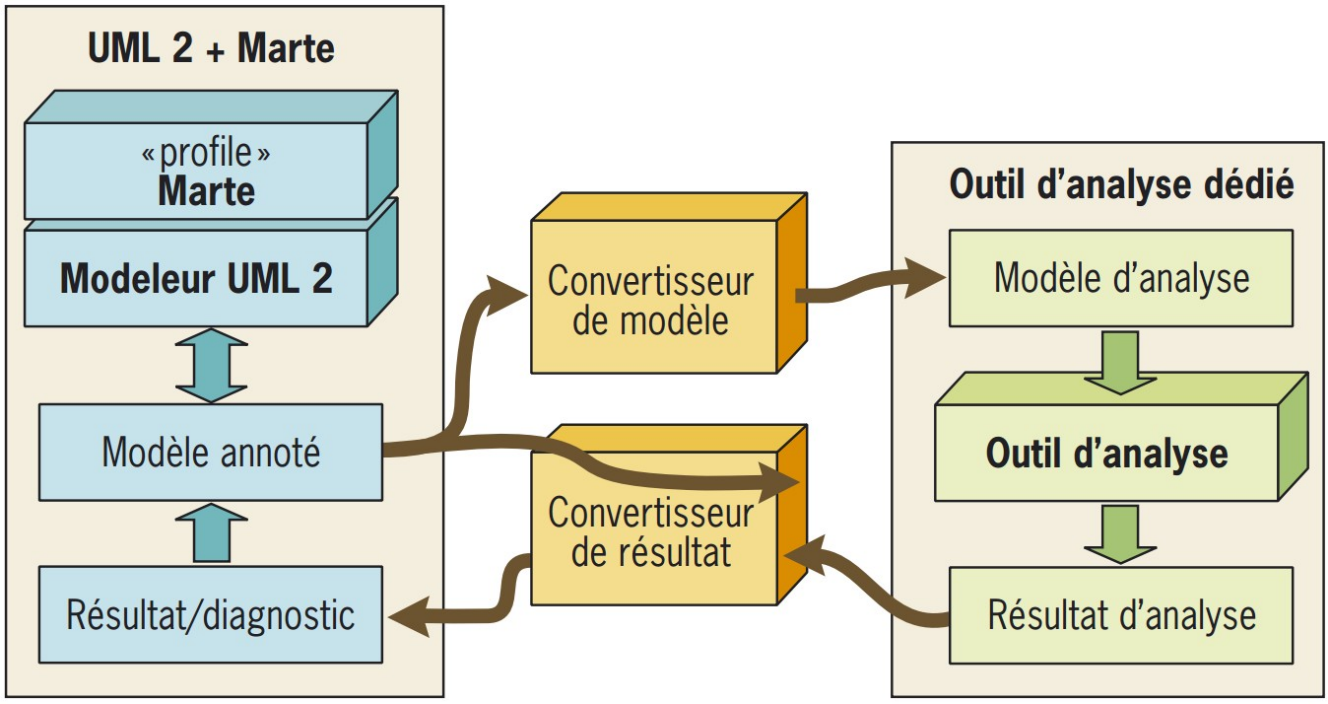

Figure 4: Atelier pour l'analyse quantitative d'une modèle Marte

La perspective d'avoir à disposition des outils standard intégrés devrait donc favoriser in fine l'utilisation de techniques d'analyse quantitative lors de la conception de systèmes temps réel embarqué.

Le profil Marte fournit aussi un support pour l'analyse quantitative qui vient enrichir les caractéristiques de son prédécesseur, le profil SPT. Les concepts repris de SPT ont en effet été alignés sur UML 2. Par ailleurs, les parties dédiées à l'analyse d'ordonnancement et à l'analyse de performances ont été harmonisées.

Enfin, de nouveaux concepts ont été introduits pour permettre une meilleure expressivité. A titre d'exemple, il est désormais possible de modéliser les contraintes de gigue, les coûts de transmission de messages, les taux d'échéances dépassées, mais aussi de caractériser différents types de temps de réponses (WCET, Worst case execution time ; BCET, Best case execution time ; ACET, Average case execution time).

Marte définit en outre un cadre commun à l'analyse d'ordonnancement (RMA, Rate monotonic priority assignment ; DMA, Deadline monotonic priority assignment, approches holistiques et modulaires) et à l'analyse de performances (réseaux de files d'attentes, réseau de Petri), tout en restant extensible à d'autres techniques d'analyse quantitative. Dans le cadre d'une analyse d'ordonnancement, on cherche à prendre en compte l'ordonnancement de tâches dans le système, pour évaluer au pire cas (WCET) si ce dernier va respecter ses échéances. Ce type de technique est généralement utilisé pour évaluer des échéances « strictes ».

On modélise donc les ressources d'exécution (tâches, processeurs), les ressources partagées (sémaphores, réseaux, bus, mémoires) et les charges de travail. Ainsi, une charge de travail décrit, grâce aux diagrammes UML comportementaux, la stimulation du système dans un contexte d'analyse particulier et l'utilisation des ressources résultantes. Pour modéliser les charges de travail, on peut se servir d'un diagramme d'activité identifiant les événements qui viennent stimuler le système et déclencher des scénarios de manière concurrente. Les extensions définies dans Marte permettent, par exemple, de fournir la période d'un événement, d'indiquer le temps d'exécution d'un scénario et de préciser l'échéance sur une exécution de bout en bout du scénario (cf. figure 8 dans le texte intégral, qui représente un scénario associé à une charge de travail et non la charge elle-même). L'ensemble de ces annotations peut être ensuite collecté automatiquement, par transformation de modèles pour alimenter un outil d'analyse (de type RMA, par exemple), et ainsi valider le respect de l'ensemble des échéances.

Références

[1] Site www.omgsysml.org pour le langage SysML
[2] Site www.omgmarte.org pour le profil Marte
[3] F.A. Schreiber. Is time a real time ? an overview of time ontology in informatics. Real-Time Computing. F127:283-307, 1994.
[4] OMG. UML 2.1.2 Superstructure specification. OMG Document Number : formal/2007-11-02. Novembre 2002. p. 423 (http://www.omg.org/spec/UML/2.1.2/)
[5] F. Mallet, C. André and R. de Simone. CCSL: Specifying clock constraints with UML/Marte. Innovations in Systems and Software Engineering 4(3):309-314, Springer, 2008.