Journées "Formalisation des activités Concurrentes" 2004

Définition d'un modèle formel pour la spécification et la vérification de la Qualité de Service temporelle

François Carcenac* — Frédéric Boniol* — Zoubir Mammeri**

** ONERA – CERT*
2, Avenue Edouard Belin 31055 Toulouse
{carcenac, boniol} @cert.fr

*** IRIT-UPS*
118, route de Narbonne 31062 Toulouse
mammeri@irit.fr

RESUME : le modèle que nous proposons offre un cadre formel à la spécification et à la vérification de la qualité de service temporelle.Le modèle, fondé sur les notions de composant, service et fournisseur de service, autorise la spécification du système indépendamment de son architecture d'implantation. Une traduction du modèle vers les automates temporisés est proposée, définissant ainsi,un modèle sémantique central. A terme, les automates temporisés permettront la vérification formelle de propriétés de qualité de service temporelle.

MOTS-CLES : qualité de service temporelle, spécification, méthodes formelles, automates temporisés, vérification

1. Introduction

La complexité croissante des systèmes embarqués et le besoin de réduire l'espace qu'ils occupent, imposent un partage intensif des ressources de calcul et de communication. Ces besoins existent, en particulier, dans le domaine de l'avionique mais également dans le domaine automobile et robotique.
Un autre besoin exprimé par les principes de l'Avionique Modulaire Intégrée concerne la forte modularité tant sur le plan de la conception d'applications, que sur le plan de l'architecture du système. En effet, un des objectifs de l'avionique modulaire intégrée [10] [11] [13] est de définir une architecture où "l'application" peut être considérée indépendamment de la plate-forme d'exécution. En d'autres termes, nous pouvons caractériser l'avionique modulaire intégrée comme un ensemble de services réalisant l'interface entre le niveau applicatif du système et le niveau matériel. Cette vision est proche de la notion de middleware, comme CORBA.
Le modèle que nous définissons vise à établir un cadre formel autorisant, d'une part, la spécification de systèmes temps réel embarqués en abstrayant l'équipement (ressource) par le service qu'il fournit et, d'autre part, permettant de vérifier que cette ressource, caractérisée par la qualité de service temporelle [3] [4] qu'elle offre, satisfait la spécification. De cette manière, différents composants offrant le même service mais avec une qualité de service différente, peuvent être confrontés à la spécification sans la modifier. Les concepts du modèle sont inspirés du domaine des télécommunications [1] [7] [8].
Dans la seconde partie, nous définissons le modèle. Les différentes "briques" du modèle sont introduites ainsi que les relations liant ces briques les unes entre elles. Dans la troisième partie, nous définissons le langage défini, une logique de causalité temporisée, pour spécifier les services ainsi que sa relation de satisfaction au moyen des traces d'événements temporisées. En se basant sur cette définition formelle, nous présentons, dans une quatrième partie, une traduction du modèle vers les automates temporisés [2]. Cette traduction nous permettra de définir les règles de cohérence et de vérification qui seront présentées dans la cinquième partie de l'article.

2. Définition du modèle

Le concept de qualité de service est une notion large recouvrant la totalité des aspects non fonctionnels d'un système. Elle est, notamment, très utilisée dans le domaine des télécommunications où la qualité de service est gérée par des mécanismes bien spécifiques.
Afin de cerner au mieux ce concept, nous nous focalisons ici sur les caractéristiques temporelles. La qualité de service temporelle traite des aspects tels que les attributs de performance et les contraintes temporelles.

Le modèle définit un ensemble d'éléments fondamentaux et deux relations d'assemblages.

2.1. Définition des éléments du modèle

Notre modèle s'articule autour de quatre concepts fondamentaux : le composant, le service, le fournisseur de service, la qualité de service.

Dans la suite de l'article, nous introduisons la notation $I(R^+)$. Cette notation désigne l'ensemble des intervalles bornés de R^+ (modèle de temps continu).

2.1.1. Composants

Un composant est une entité structurée. Il décrit une ressource matérielle ou logicielle du système spécifié.
Nous avons une vision "boite grise" du composant. Il est décrit par un ensemble d'interfaces et son comportement fonctionnel.
La définition d'un composant est la suivante :

$$Composant =< Setof(Interface), Comportement >$$
$$Interface =< In, Out, ComportementTemporisé >$$
$$In = Setof(Event)$$
$$Out = Setof(Event)$$
$$In \cap Out = \varnothing$$

Une interface décrit un langage temporisé dont le vocabulaire est défini par l'union des entrées et des sorties. De plus, le comportement temporisé d'une interface est spécifié par un automate temporisé. Il modélise le protocole d'utilisation du service.
Le comportement du composant, décrit par un automate temporisé, spécifie l'aspect fonctionnel du composant.

2.1.2. Services

Un service décrit une fonctionnalité abstraite. Il se décrit par un ensemble de points d'accès. Ces derniers définissent le vocabulaire (ensemble d'événements) du ou des différents types de client. Considérons un service de communication asynchrone monodirectionnel ; la spécification du service comportera deux points d'accès (deux types de client), le premier désignant l'émetteur et le deuxième désignant le récepteur.
Afin de définir l'aspect fonctionnel d'un service, nous définissons une logique de causalité temporisée. Cette logique permet d'exprimer de manière naturelle et intuitive le comportement attendu d'un service. La syntaxe et la relation de satisfaction sont définies dans la section 3.
La définition d'un service est donnée ci-dessous :

$$Service =< PtsAccès, Evts\_Internes, CausalitéTemporisée, ParamètresQdS >$$
$$PtsAccès = Setof(PA)$$
$$Evts\_Internes = Setof(ServiceEvent)$$
$$PA = Setof(ServiceEvent)$$
$$\forall PA1, PA2 \in PtsAccès. PA1 \cap PA2 = \varnothing$$
$$primitives = \left(\bigcup_{i \in PtsAccès} i \right) \cup Evts\_Internes$$
$$CausalitéTemporisée \in L_C$$

2.1.3 Fournisseusr de service

Le dernière brique de notre modèle concerne l'entité "fournisseur de service". Il s'agit d'un élément à distinguer du composant. En effet, le fournisseur de service est considéré comme une entité de type "boite noire". On ne connaît rien de son comportement. Néanmoins, à un niveau de description donné, on connaît d'un fournisseur de service les aspects métriques (en terme de qualité de service).
Le fournisseur de service est caractérisé par un ensemble d'attributs de performance et par une fonction de valuation dont le domaine d'arrivée est l'ensemble des intervalles temporels bornés.
La fonction de valuation est une fonction totale, tous les éléments appartenant au domaine de cette fonction ont une image.

La définition du fournisseur de service est donnée ci-dessous :

$$FournisseurdeService =< AtributsPerfo, QdSofferte >$$
$$QdSofferte : AttributsPerfo \rightarrow I(R^+)$$

2.2. Définition de l'assemblage des éléments

Afin de construire le système global, nous définissons deux relations d'assemblages : la relation Client qui associe un composant à un service et la relation Fournisseur associant un fournisseur de service à un service.

2.2.1. Relation client

La relation associe un composant à un service et définit un ensemble d'exigences sur ce service. L'ensemble définit en fait une conjonction de formules logiques. Le choix de la logique s'est porté sur la logique XSL (eXtended Safety modal Logic) définie par Augusto Burgueño [5]. Cette logique d'actions permet d'exprimer des propriétés

temporisées sur des successions d'événements émis ou reçus par le client du service. De plus, cette logique admet une traduction vers les automates temporisés.

$$Client = \left\{ (comp, i, serv, pa, mapping, QdSrequise) \left| \begin{array}{l} comp \in Composant \wedge \\ i \in comp.\text{int} \wedge \\ serv \in Service \wedge \\ pa \in serv.ptsAccès \wedge \\ mapping = (i.Out \times pa_1) \cup (pa_2 \times i.In) \wedge \\ pa_1 \cap pa_2 = \varnothing \wedge \\ pa_1 \cup pa_2 = pa \wedge \\ QdSrequise \in XSL \end{array} \right. \right\}$$

La relation associe, plus précisément, l'interface d'un composant à un point d'accès du service. Cette association nécessite la définition d'une relation, appelée *mapping*, chargée de définir les événements à synchroniser.
La relation *mapping* définit deux partitions du point d'accès *pa*. L'une des deux partitions, notée pa_1 exprime l'ensemble des événements de *pa* synchronisés sur les événements en sortie de l'interface *i* du composant (*i.Out*). L'autre partition, pa_2, représente les événements sur lesquels sont synchronisées les entrées de *i* (*i.In*).

2.2.2. Relation Fournisseur

La relation associe un fournisseur de service à un service. Cette association permet d'obtenir une valuation des caractéristiques de qualité de service.
Pour ce faire, nous avons fait l'hypothèse que l'ensemble des paramètres de qualité de service est inclus dans l'ensemble des attributs de performance.
Cette hypothèse permet de garantir que tous les paramètres de qualité de service sont valués. Nous rappelons, en effet, que la fonction de performance (cf. 2.1.3) est totale. Par conséquent, tous les paramètres du service admettent une valeur dans $I(R^+)$.

$$Fournisseur = \left\{ (prov, serv) \left| \begin{array}{l} prov \in Fournisseur \wedge \\ serv \in Service \wedge \\ serv.ParamètresQdS \subseteq prov.attributsPerfo \end{array} \right. \right\}$$

3. Langage pour la spécification de services

Les services définissent l'interface entre la ressource (fournisseur de service) et le composant utilisant cette ressource. Afin de spécifier ces services, nous avons choisi d'adopter un formalisme permettant de représenter les liens de cause à effet entre les événements intervenant dans leur réalisation. L'objectif de notre modèle étant de vérifier des propriétés temporelles, ce formalisme est temporisé.

Le langage défini pour la spécification est une logique de causalité temporisée. L'utilisation d'une telle logique se justifie par sa simplicité d'utilisation et de compréhension.

3.1. Syntaxe

La logique de causalité est définie au moyen de trois constructeurs. L'objectif initial concerne l'expression de proposition telles que "l'occurrence de e1 entraîne l'occurrence de e2 dans l'intervalle de temps I".
Soit E un ensemble d'événements. Une formule φ appartient à la logique si et seulement si φ est de la forme :

$$\varphi ::= (evt_1 \rightarrow_I evt_2) \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \ , \ evt_1, evt_2 \in E$$

3.2. Satisfaction

La satisfaction de la logique est définie au moyen des traces d'événements temporisées.
Une trace d'événements temporisée est une séquence d'événements totalement ordonnés définie comme étant de la forme σ = (e1,t1) (e2,t2) … (en,tn), où le couple (e,t) désigne l'occurrence de l'événement e à l'instant t.
La relation de satisfaction de la logique est la suivante :

$$\sigma \models e_1 \rightarrow_I e_2 \ \texttt{ssi} \ \forall (e,t) \in \sigma . e = e_1 \Rightarrow \exists (e',t') \in \sigma \exists d \in I . e' = e_2 \wedge t' = t + d$$

$$\sigma \models f_1 \wedge f_2 \ \texttt{ssi} \ \sigma \models f_1 \ \wedge \ \sigma \models f_2$$

$$\sigma \models f_1 \vee f_2 \ \texttt{ssi} \ \sigma \models f_1 \ \vee \ \sigma \models f_2$$

4. Traduction vers les automates temporisés

Le modèle que l'on a choisi pour exprimer la sémantique du modèle est le formalisme des automates temporisés. Ce modèle est étendu par l'ajout d'invariants d'état. L'invariant d'état est un prédicat exprimé sur les variables de l'automate permettant de forcer le tir d'une transition. En effet, la garde permet simplement de déterminer si la transition est franchissable ou non. Une transition **peut** être franchie lorsque sa garde est *vraie*. En revanche, elle **doit** être franchie quand la garde est *vraie* et l'invariant de l'état courant est *faux*.

La traduction du système en automates temporisés permet de préciser la sémantique dynamique du modèle.

Les éléments du modèle faisant l'objet d'une traduction sont : le service et les synchronisations résultant de la relation *mapping*.

L'automate du service est la composition parallèle de tous les automates résultant de la traduction d'un triplet (e1, I, e2). La traduction d'un triplet de la relation de causalité est définie dans la figure 1a. Les relations R1 et R2 désigne respectivement les inégalités < ou ≤, et > ou ≥. Les inégalités sont larges si les bornes sont exclues et strictes dans le cas contraire.
La relation de *mapping* possède également son équivalent en terme d'automates temporisés. L'automate temporisé d'un couple de la relation *mapping* est défini dans la figure 1b. Tout appartenant à cette relation décrit une synchronisation instantanée.

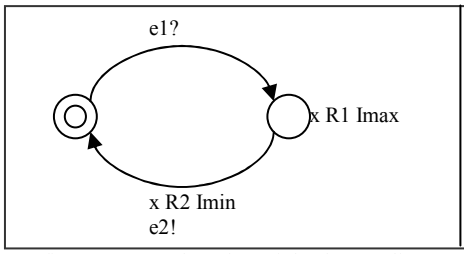

figure 1a : encodage d'un triplet de causalité

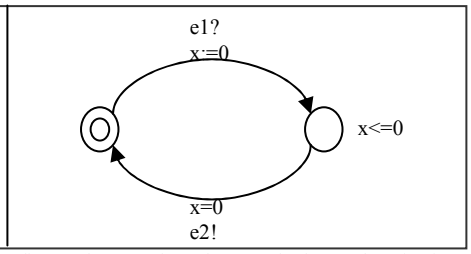

figure 1b : encodage d'une règle de synchronisation

5. Cohérence et vérification

5.1. Cohérence du modèle

La cohérence structurelle est établie par définition du modèle. Cependant, il reste à vérifier la cohérence comportementale du modèle ainsi que la compatibilité de qualité de service.
On dit qu'un modèle est cohérent si la composition parallèle des automates temporisés résulte en un système non bloquant. Le blocage d'un système d'automates temporisés se présente de plusieurs manières. La première cause de blocage réside dans l'exécution d'une infinité de transitions dans un intervalle de temps borné. Le temps global du système ne peut donc plus évoluer et le système se retrouve bloqué. Ce problème de blocage est appelé problème de Zénon.
La seconde cause de blocage est appelée interblocage. Ce problème intervient lorsque deux automates A1 et A2 se retrouvent simultanément dans un état où A1 attend e1 (émis par A2) et A2 attend e2 (émis par A1).
La dernière cause de blocage dans un automate temporisé est due à la propriété d'invariant d'état. Le blocage du temps intervient quand l'invariant d'un état est falsifié.
Ces blocages peuvent être détectés au moyen d'outils de vérification tels qu'Uppaal [12].
La cohérence comportementale du système est établie en faisant la composition parallèle des automates résultant de la traduction du modèle.

La compatibilité de qualité de service est établie en confrontant la qualité de service offerte à la qualité de service requise. Sur le modèle d'automates résultant de la traduction, la compatibilité peut être vérifiée en composant les automates de chaque élément (composant, service, synchronisations) avec l'automate de test représentant les exigences de qualité de service. Si ce système n'atteint jamais l'état falsifiant les exigences de QdS, la qualité de service offerte est dite "compatible" avec la qualité de service requise. Cette vérification peut être effectuée à l'aide de l'outil Uppaal.

5.2. Vérification de propriétés

La vérification de propriétés globales du système est effectuée en procédant de la même façon que précédemment. Le système d'automates temporisés résultant de la traduction est composé en parallèle avec l'automate de test représentant la propriété globale.

6. Perspectives

Notre modèle définit un cadre formel autorisant la spécification de la qualité de service temporelle. L'utilisation des automates temporisés nous permet de vérifier la cohérence du modèle et la compatibilité de qualité de service au moyen de techniques de model checking.
Toutefois, la traduction vers les automates temporisés n'est pas définie formellement. La prochaine étape de nos travaux consistera à prouver que la logique est bien satisfaite par notre traduction.
La seconde perspective consisterait à définir le raffinement d'un service afin d'en avoir une vision plus "fine".
Pour finir, une dernière perspective consisterait à étudier le rapprochement de notre modèle avec le modèle de ressources défini dans le profil UML temps réel [1].

7. Bibliographie

[1] OMG. *UML Profile for Time, Performance and Scheduling Specification*, Mars 2002. Document OMG numéro ptc/02-03-02.

[2] Alur et Dill. *A theory of timed automata*, 1994.

[3] Z. MAMMERI. *Gestion de la qualité de service : concepts et mécanismes*, Septembre 2003. Ecole d'été Temps Réel 2003. ISBN 2-9520326-0-2.

[4] J.P. THOMESSE. *La qualité de service dans les réseaux industriels : les normes IEC*, Septembre 2003. Ecole d'été Temps Réel 2003. ISBN 2-9520326-0-2.

[5] Augusto BURGUEÑO ARJONA. *Vérification et synthèse de systèmes temporisés par des méthodes d'observation et d'analyse paramétrique,* 1998. Thèse de doctorat. Ecole Nationale Supérieure de l'Aéronautique et de l'Espace.

[6] J. FREDRIKSSON, J. HAMMARBERG, et autres. *Component Based Software Engineering for Embedded Systems : a literature survey*, Juin 2003. Rapport Technique. Projet SAVE. ISSN 1404-3041.

[7] ISO. *The reference model of Open Systems Interconnection*, 1982. Standard international ISO IS-7498.

[8] ISO. *Open Distributed Processing-reference model,* Juillet 1995. ISO/IEC 13236.

[9] Bran SELIC, Garth GULLEKSON, Paul T. Ward. *Real Time Object Oriented Modelling*, 1994. Wiley Professional Computing. ISBN 0-471-59917-4

[10] Equipe PRISME. *PRISME Avionique nouvelle*, 1997.Projet PRISME, Rapport d'activité.

[11] J. FOISSEAU, V. WIELS, F. BONIOL. *Un exemple de modèle conceptuel de référence pour l'ingénierie des systèmes avioniques*, Juin 2001. Conférence annuelle d'ingénierie système, AFIS'2001 Toulouse.

[12] K. LARSEN, P. PETTERSSON, W. YI. *Compositional and symbolic model checking of real time systems,* 1995. Proceedings of the 16th IEEE Real Time Systems Symposium, RTSS'95.

[13] C.R. SPITZER, *Digital Avionics Systems*, Mc Graw-Hill, 1993.