Observateurs de propriétés temporelles pour les systèmes d'acquisition de données

Belgacem BEN HEDIA Jean Philippe BABAU Fabrice JUMEL
CITI-Insa Lyon, 20 avenue Albert Enstein, 69621 Villeurbanne Cedex
{belgacem.ben-hedia ;jean-philippe.babau ;fabrice.jumel}@insa-lyon.fr

Riadh ROBBANA
EPT, Rue El khawarizmi 2078 La Marsa Tunisie
riadh.robbona@fst.rnu.tn

Résumé

Dans le domaine des applications temps réel de contrôle des procédés, ce travail cherche à caractériser temporellement le retard des données acquises sur l'état du procédé, via un logiciel dédié appelé pilote. Ce travail propose un modèle formel des pilotes d'acquisition de données basé sur les automates temporisé communicants, l'utilisation d'observateur pour la mesure de retards et enfin montre l'influence des paramètres du pilote sur les retards.
MOTS-CLÉS : temps réel, automates temporisés, pilotes d'équipements, validation, qualité de service

1 Introduction

L'utilisation de l'informatique dans le cadre du contrôle des procédés est de plus en plus courante. Elle permet d'implémenter des lois de contrôle utilisant les données délivrées par les capteurs pour produire des commandes au niveau des actionneurs. Actuellement ces systèmes sont de plus en plus complexes et la réutilisation des composants logiciels doit permettre de réduire les coûts de développement. Dans le domaine du contrôle des procédés, les composants réutilisables sont classiquement des services fournis par un système d'exploitation temps réel respectant les normes OSEK [13] ou POSIX [1] et des services de communication avec le procédé contrôlé via des pilotes d'équipements (systèmes d'entrée/sortie). Le but de ce pilote d'équipement et de fournir un interface entre le capteur physique et l'application. Dans ce travail nous considérons les pilotes d'acquisition de données [7]. Du fait des contraintes de QoS inhérentes aux systèmes considérés, il est nécessaire de pouvoir caractériser la QoS fournie par un pilote. Dans ce travail nous considérons un critère classique de QoS pour un système de contrôle des procédés : le retard. Les techniques le plus connues pour calculer un retard maximum sont basées sur une analyse d'ordonnancement des tâches et des messages [10]. Ces techniques considèrent des architectures simples, généralement statiques et produisent des bornes généralement pessimistes. Dans le but de remédier à ce problème, plusieurs travaux se basent sur des techniques permettant d'avoir une modélisation exhaustive du comportement temporel des systèmes. Il s'agit en particulier d'utiliser les automates temporisés communicants ou les automates hybrides [2], [6] [12]. Nous avons choisi d'utiliser ce type d'approche pour l'étude de la QoS des pilotes (à l'aide de l'outil IF [4], [5]). Ceci permet d'avoir des caractéristiques temporelles plus réalistes de l'application pour des architectures plus complexes (prise en compte du comportement dynamique). Le reste de l'article est organisé de la manière suivante. La partie suivante présente le système d'acquisition des données et les critères de QoS a évalué. La troisième partie présente la stratégie de modélisation. Enfin la dernière partie illustre l'approche en analysant l'impact de paramètres de systèmes d'acquisition sur les critères de QoS. Nous terminons par une conclusion générale et quelques perspectives.

2 Contexte

2.1 Système d'acquisition des données

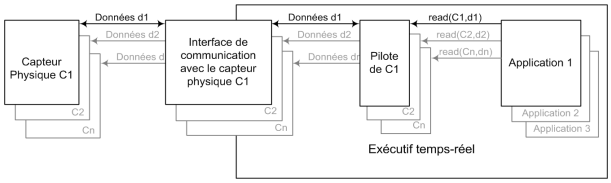

FIG. 1 – Système d'acquisition de données

Un système d'acquisition des données *figure 1* est composé de quatre éléments :

Capteur physique : Il permet de convertir les informations (température, vitesse) produites par l'environnement physique en données numériques.

Interface de communication : Il connecte le capteur à la partie logicielle du système. Il récupère les données produites par le capteur physique et les sauvegarde dans des registres accessibles par le pilote.

Pilote d'équipement : C'est un logiciel dédié intégré au système d'exploitation et indépendant de l'application. Il permet l'abstraction de la couche matérielle. Il permet à l'application d'accéder aux données.

L'application temps réel : une application temps réel récupère les données du pilote.

On distingue ici le pilote matériel (appelé ici interface de communication) et le pilote logiciel (appelé ici pilote d'équipement).

2.2 Définition des critères de QoS

Suivant les applications, les critères de QoS les plus appropriés peuvent être différents (retard, taux de perte, sûreté de fonctionnement) [11]. Pour la suite, les critères de QoS considérés sont :

Retard : les données utilisées par l'application reflètent bien l'état du procédé mais, à un instant plus ou moins passé.

3 Modèles formels de système d'acquisition de données

3.1 Principe de modélisation du système

Cette partie présente les principes de modélisation de système d'acquisition de données en IF. Chaque élément de système est modélisé par un ou plusieurs processus IF (figure 2). Le pilote d'équipement est modélisé par un ensemble de process correspondant aux fonctions qu'on trouve dans un pilote d'équipement classique (*Read(), Write(), Open(), ...*), et un process correspondant à la tâche périodique *Driver()*. Les process (*DrvInstall, DevAdd, Open, Read*) sont créés selon les besoins exprimés par l'application, de la même manière que les fonctions correspondants sont appelées par l'application.
Les capteurs physiques et l'interface de communication sont modélisés par un process qui simule le fonctionnement d'un ou plusieurs capteurs appelé *SimulSensor()*
Chaque tâche de l'application temps réel est modélisée par un process qui va lire les données suivant une loi donnée (périodique, sporadique, ...)
Seul le comportement temporel est ici modélisé. Les valeurs des données ne sont pas considérées. Ainsi une donnée *message* navigant dans le système contient deux champs *(data, numero)* où *data* est la donnée en elle même et *numero* represente le numèro de l'occurence de la donnée *data*.
Nous considérons trois lois de production de données par le capteurs : périodique (avec ou sans gigue), sporadique, ou en rafale [9]. Le registre du stockage des données est de type FIFO ou LIFO. Puisque la durée d'une transition en IF est nulle, l'évolution du temps est considéré uniquement à travers les gardes temporelles sur les horloges. Ces gardes peuvent représenter un intervalle. Dû à la sémantique temporelle de IF (durée d'une transition nulle), modéliser une durée d'exécution (relier à la durée d'exécution et d'attente de processeur d'une tâche) peut ère obtenue grâce à l'ajout d'un état d'attente dans les processus concernés.

3.2 Modélisation de la QoS

Pour évaluer les critères de QoS nous avons d'abord proposé d'introduire des variables d'observation dans le modèle [8]. Les variables sont des compteurs d'instances

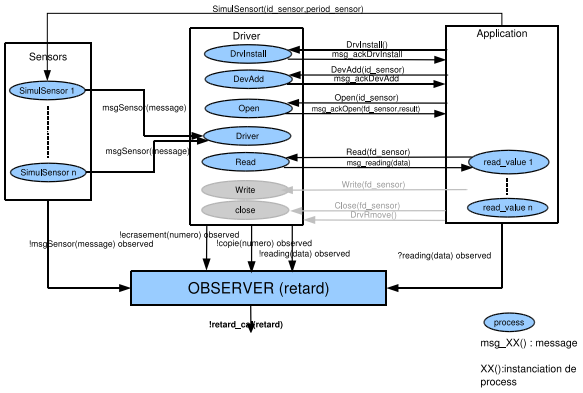

FIG. 2 – Modèle IF du Système

des signaux (reçus par l'application, produits par le capteur) et des compteurs temporels (date de production et de consommation des données). Ce travail n'étant pas généralisable, nous proposons, dans une deuxième approche, d'introduire des processus observateurs. Un processus observateur est un automate qui permet de suivre l'évolution du modèle et donc, en l'instumentant, d'évaluer une propriété de QoS. C'est ce que nous détaillons maintenant.

3.2.1 Généralité sur les observateurs des propriétés temporelles

La mise en place de la vérification d'une propriété nécessite l'expression de cette propriétés temporelles sous une forme adaptée qui permettra sa vérification *à la volée* (durant la génération du STE (système de transition étiqueté)) ou a *posteriori* (sur un STE déjà généré). une des techniques la plus utilisées consiste à exprimer cette propriété sous la forme d'un observateur [3]. Généralement, les observateurs permettent de fournir des réponses du type : une propriété est validée (*succès*) ou non-validée (*échec*). Généralement dans les techniques de vérification *à la volée* la génération de l'espace d'états est interrompue une fois la propreté n'est pas valide. Vue que les observateurs sont généralement utilises pour vérifier des propriétés de vivacité borné, et étant donnée que la propriété de retard qu'on veut évalué est une propriété de vivacité et elle est toujours bornée (une donnée ne peut pas rester infiniment dans le pilote d'équipement : elle est consommée par l'application ou écraser par une nouvelle donnée plus fraîche), alors l'observateur peut être utilisé pour évalué une telle propriété.

3.2.2 Observateurs IF

Un observateur IF est un automate exprimant la propriété à vérifié et réagit en fonction des événements, des changements des états et des modifications des valeurs des variables du système observé.
L'automate d'un observateur IF est semblable à l'automate d'un processus IF classique avec les particularités qu'il réagit d'une manière synchrone aux évènements et

conditions du système observés, il est plus prioritaire que de système observé et il est déterministe. Chaque état d'un observateur IF peut être : Un état dit *normale* (ou d'observation), Un état dit *échec* (proprité non valide) ou un état dit *succès*(propriété valide). De même que les autre observateurs les observateurs IF fournis des réponses du type : une propriété est validée (*succès*) ou non-validée (*échec*) Dans ce travail, on ne veut pas se contenter de vérifier une propriété de ce type, mais de faire de l'évaluation des valeurs possibles de retard. Dans la suite, nous présentons une démarche pour adapter les observateurs IF afin d'évaluer le retard sur les données.

3.2.3 Observateur de retard

Notre approche consiste à observer chaque occurrence de la donnée à partir de sa production par le capteur physique jusqu'à sa consommation par l'application. Et ainsi calculer les retards (délais d'acheminements) possibles de chaque occurrence de la donnée. L'observateur de retard est un observateur IF qui au lieu de vérifier une propriétés et de donner le résultat sous la forme (*succès/échec*) il fournit un résultat de type valeur de retard possible. L'observateur de retard observe alors les occurrences de la donnée et dès qu'une occurrence est reçue par l'application il passe dans l'état *succès* afin de signaler qu'un retard possible est détecté (calculable). L'observateur de retard ne contient pas alors d'état *échec* .

Fonctionnement de l'observateur de retard l'automate de la figure 3 représente le fonctionnement de l'observateur de retard :
Il contient deux états : un état *Init* dans lequel il est en état d'observation des occurrences de la donnée et un état *RetardCalc* qui correspond à l'état *succès* décrit précédament.
Pour une occurrence *(data,numero)* donnée l'observateur régit de la manière suivante :

- À la détection de l'émission *!msgSensor(message)* (production d'une occurrence *(data,numero)* par le capteur physique), l'observateur déclenche une horloge associée à cette occurrence *Xnumero :=0* (transition n° 2 de la figure 3).
- À la détection d'une lecture par l'application de l'occurrence *(data,numero) ?msg_Reading(data)* , l'observateur passe dans l'état de détection de retard *RetardCalc* (transition n° 5 de la figure 3).
- Un compteur des copies de l'occurrence *(data,numero)* qui sont dans le système permet d'arrêter l'observation de l'occurrence *(data, numero) reset Xnumero* , ainsi *numero* peut être réutilisé pour identifier une nouvelle occurrence (transitions n° : 1,3,4 de la figure 3).

La réutilisation des identifiant *numero* permet d'éviter l'explosion combinatoire dû à l'incrémentation à l'infini de l'identifiant d'occurrence *numero*

Une analyse du STE et de fichier des états du STE généré par IF permet d'identifier les états *RetardCalc* et de récupérer la valeur d'horloge *Xnumero* correspondant à l'occurrence reçue par l'application à cet instant. Les valeurs de *Xnumero* récupérés représentent les retards possibles.

4 Expérimentations

Dans le but d'évaluer l'intérêt de l'approche proposée, on considère un système mono capteur, mono application avec les hypothèses suivantes :

- La période de capteur physique Pc vaut 10 unités de temps.
- Le pilote est en mode scrutation et sa période Pp est inconnue *a priori*, la taille de son registre est de 1 avec écrasement de donnée si nécessaire, il a un temps de réponse égale à [1,2].
- La période de l'application Pa est égale à 20 unités de temps pour Fig. 5 et égale à 17 unités de temps pour Fig. 5 et son temps de réponse est égale à [6,7].
- Enfin, nous considérons que tous les processus du système (capteur, pilote et application) démarrent au même instant.

Les résultants des valeurs des retards possibles sont présentés dans les figures Fig. 4 et Fig. 5. Dans la figure Fig. 4, le retard minimal est de [6,7] (lecture toutes les $(20k + [6, 7])$ unités de temps d'une donnée a priori produites par le capteur toute les $(20k)$ unités de temps). Quelques points remarquables demeurent, correspondants à un séquencement particulier, par exemple pour 10 et 20 où la synchronisation entre l'application, le capteur et le pilote amène à l'existence d'un unique retard possible de [6,7] unités de temps. Enfin sur la figure Fig. 5, le fait de changer la période de l'application de 20 unités de temps à 17 unités de temps entraîne des changements considérables au niveau, bien sûr, des valeurs obtenues, mais aussi de la forme de la courbe. On remarque particulièrement que ce n'est pas en scrutant plus vite (17 unités de temps) c'est-à-dire en chargeant le système qu'on a des retards plus faibles ; au contraire le retard maximal est plus faible en scrutant à 20 unités de temps. Il est aussi important de remarquer qu'il y a des valeurs de retard impossibles (les intervalles de retard]7,16[et]17,25[sont des retard impossible dans le cas de système de la figure 4)

5 Analyse et Conclusion

Nous avons proposé des modèles de spécification comportementale et temporelle d'un système d'acquisition de données. Ces modèles permettent de caractériser un pilote d'équipement du point de vue des retards des données. Ces modèles ont été réalisés en IF et prennent en compte le capteur, l'interface de communication, le pilote d'équipement et l'application. Les résultats présentés ont montré que la forme des retards est complexe et dépend de plusieurs paramètres. L'évaluation des retards passe par la mise ne place d'un automate observateur du cycle de vie

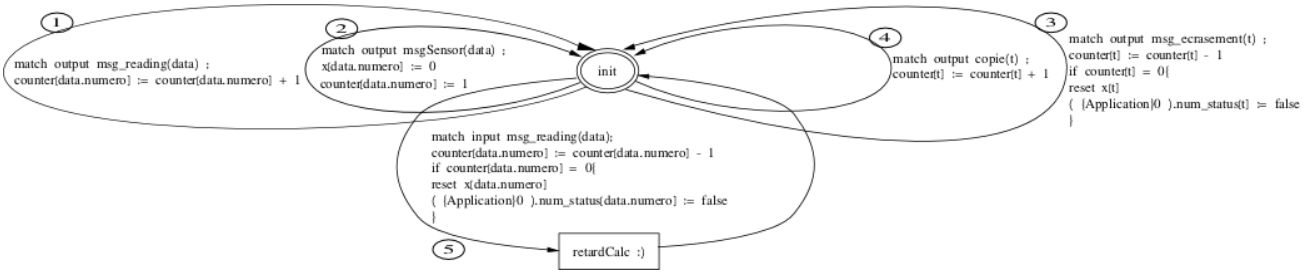

FIG. 3 – Automate de l'observateur de Retard

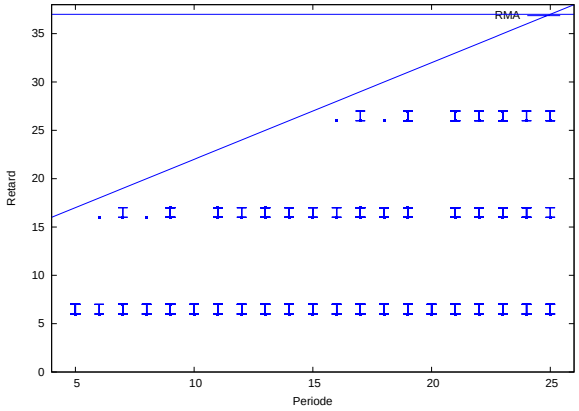

FIG. 4 – Temps de réponse du pilote [1,2], Période application 20, Temps de réponse dans [6,7]

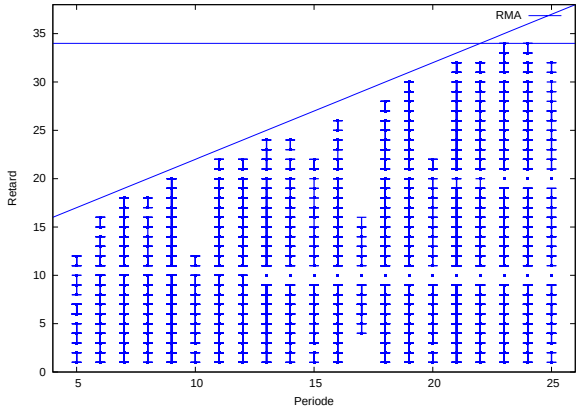

FIG. 5 – Temps de réponse du pilote [1,2], Période application 17, Temps de réponse : [6,7]

des occurences des données au sein du flot de données. L'exemple traité montre la grande richesse d'expression des modèles proposés. En particulier, il est possible de prendre en compte l'existence de propriétés temporelles définies sous forme d'intervalles.

Références

[1] I. S. 9945-1. Information technology-portable operating system interface (posix)-part 1 : System application : Program interface (api) [c language]. 1996.

[2] M. Belarbi, J.-P. Babau, and J.-J. Schwarz. Temporal verification of real-time multitasking application properties based on communicating timed automata. In *DS-RT '04 : Proceedings of the Eighth IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT'04)*, pages 188–195, Washington, DC, USA, 2004. IEEE Computer Society.

[3] A. Bouajjani, R. Echahed, and R. Robbana. Verification of context-free timed systems using linear hybrid observers. In *CAV '94 : Proceedings of the 6th International Conference on Computer Aided Verification*, pages 118–131, London, UK, 1994. Springer-Verlag.

[4] D. M. Bozga. *Symbolic verification for communication protocols*. PhD thesis, L'UNIVERSITE JOSEPH FOURRIER - GRENOBLE I, 1999.

[5] D. M. Bozga, S. Graf, and L. Mounier. If-2.0 : A validation environment for component-based real-time systems. In K. L. Ed Brinksma, editor, *Proceedings of CAV'02 (Copenhagen, Denmark)*, volume 2404 of *LNCS*, pages 343–348. Springer-Verlag, July 2002.

[6] A. David, G. Behrmann, K. Larsen, and W. Yi. A tool architecture for the next generation of uppaal, 2003.

[7] W. Fokkink, N. Ioustinova, E. Kesseler, J. Pol, Y. Usenko, and Y. Yushtein. Refinement and verification applied to an in-flight data acquisition unit, 2002.

[8] B. B. Hedia, F. Jumel, and J.-P. Babau. Evaluation de la qualité de services des pilotes d'équipement pour les systémes d'acquisition. *Journal européen des systémes automatisés, MSR'05 colloque Modélisation des Systémes Réactifs No5*, 39(1-3 (1 p.3/4) :47–62, 2005.

[9] Z. MAMMERI. Expressing and deriving time constraints in real-time applications. 1998.

[10] J. Migge, A. Jean-Marie, and N. Navet. Timing analysis of compound scheduling policies : Application to posix, 2001.

[11] P. Ramanathan. Overload management in real-time control applications using (m, k)-firm guarantee. *IEEE Transactions on Parallel and Distributed Systems*, 10(6) :549–559, 1999.

[12] S. Vestal. Modeling and verification of real-time software using extended linear hybrid automata. In *Proceedings Fifth NASA Langley Formal Methods Workshop*, 2000.

[13] A. Zahir and P. Palmieri. Osek/vdx - operating systems for automotive applications. *IEE Seminar Digests*, 1998(523) :4–4, 1998.