Aproximación al Modelado de Tareas para Verificación de Tiempo Real Siguiendo un Enfoque Dirigido por Modelos

Juan F. Inglés-Romero, Cristina Vicente-Chicote, Diego Alonso Cáceres
División de Sistemas e Ingeniería Electrónica (DSIE), Universidad Politécnica de Cartagena
Campus Muralla del Mar, s/n. Cartagena 30202, Spain
{juanfran.ingles, cristina.vicente, diego.alonso}@upct.es

Resumen

V^{3}CMM *(3-View Component Meta-Model)* es un lenguaje de modelado basado en componentes que permite el diseño de aplicaciones de forma independiente de la plataforma sobre la que éstas se implementarán finalmente. En su versión inicial, V^{3}CMM no soportaba el modelado de restricciones temporales asociadas al comportamiento de sus componentes.

Este artículo presenta algunos resultados preliminares sobre: (1) la extensión de V^{3}CMM para abordar el modelado de requisitos temporales, (2) un procedimiento para la generación de modelos de tareas a partir de las especificaciones realizadas con el nuevo V^{3}CMM extendido y (3) la posibilidad de interoperar con herramientas de análisis existentes para verificar la planificabilidad de los modelos de tareas anteriores, todo ello, siguiendo un enfoque dirigido por modelos.

1. Introducción

En el panorama actual, existe una creciente necesidad de abordar el desarrollo de sistemas cada vez más complejos. Algunos de estos sistemas presentan requisitos temporales estrictos (Hard Real-Time) o flexibles (Soft Real-Time).

El grupo de investigación *División de Sistemas e Ingeniería Electrónica* (DSIE) acumula una amplia experiencia en el desarrollo de software para sistemas reactivos aplicando, entre otros, enfoques como el de Desarrollo de Software Basado en Componentes (DSBC) [15] y el de Desarrollo de Software Dirigido por Modelos (DSDM) [3]. Fruto de estos trabajos se ha definido V^{3}CMM [1], como un lenguaje de modelado basado en componentes para el diseño de sistemas de forma independiente de la plataforma. Sin embargo, V^{3}CMM no permite modelar los requisitos temporales asociados a ciertos componentes. Esta limitación impide analizar el comportamiento temporal de los modelos para adoptar decisiones de diseño.

Este trabajo presenta algunas ideas y resultados preliminares sobre la extensión del lenguaje V^{3}CMM para modelar sistemas con requisitos de tiempo real, permitiendo la generación de modelos de tareas equivalentes con el fin de poder analizar su planificabilidad, haciendo uso de alguna de las múltiples herramientas existentes para ello.

El resto de este artículo se estructura como sigue. La sección 2 describe las principales características de V^{3}CMM y la sección 3 las extensiones realizadas sobre él para permitir el modelado de aspectos temporales. La sección 4 describe el procedimiento de generación de conjuntos de tareas que se ajusten a los requisitos temporales modelados con la extensión de V^{3}CMM para su posterior análisis. La sección 5 aborda algunos trabajos relacionados y, por último, la sección 6 presenta algunas conclusiones y líneas de trabajo futuras.

2. El meta-modelo de componentes V^{3}CMM

V^{3}CMM proporciona a los diseñadores un lenguaje de modelado basados en componentes expresivo y simple, que permite realizar diseños arquitectónicos (describiendo tanto la estructura como el comportamiento de los componentes del sistema) de forma independiente de la plataforma y del dominio concreto de aplicación. Los componentes V^{3}CMM tienen una semántica sencilla, lo que facilita su transformación a otros modelos (por ejemplo, para análisis o simulación), así como la generación de código.

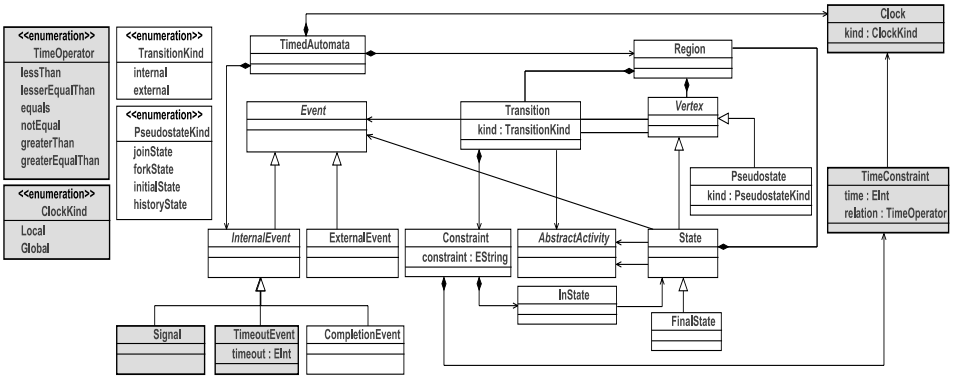

Figura 1. Extensión de la vista de comportamiento de V^{3}CMM para modelar autómatas temporizados. Los nuevos conceptos añadidos al meta-modelo se han destacado en color gris.

La simplicidad, la economía de conceptos y la reutilización son las principales guías de diseño tras V^{3}CMM.

V^{3}CMM incluye un conjunto reducido de conceptos, suficientes para modelar aplicaciones basadas en componentes. V^{3}CMM no "reinventa la rueda", sino que adopta y adapta algunos de los conceptos incluidos en el estándar UML 2, lo que facilita su uso por parte de la comunidad de los diseñadores de software.

V^{3}CMM define tres vistas complementarias, altamente cohesionadas y débilmente acopladas: (1) una *vista estructural*, para describir la estructura estática de los componentes que integran la arquitectura de la aplicación (tanto simples como compuestos), (2) una *vista de coordinación* (basada en máquinas de estados), para modelar el comportamiento de los componentes frente a los mensajes que reciben del exterior, y (3) una *vista algorítmica* (basada en diagramas de actividad), para describir los algoritmos ejecutados por el componente dependiendo del estado en que se encuentre. Existe una cuarta vista asociada a las tres anteriores, en la que se definen los elementos comunes al sistema como, por ejemplo, los tipos de datos y las interfaces. Una descripción más detallada de V^{3}CMM junto con un ejemplo de aplicación en el dominio de la Robótica puede consultarse en [1].

3. Extensión de V^{3}CMM para modelar aspectos temporales

El modelado de aplicaciones de tiempo real requiere que se contemplen, de forma explícita, los requisitos temporales del sistema, algo que la versión anterior de V^{3}CMM no permitía. El modelado de estos requisitos permitirá analizar el comportamiento temporal del código generado y verificar que efectivamente se ajusta al diseño. La Figura 1 muestra la extensión realizada sobre el meta-modelo de V^{3}CMM (en particular, sobre su vista de coordinación), para incluir el modelado de aspectos temporales. Como se puede observar, se ha optado por adoptar un enfoque basado en autómatas temporizados [2], dada su cercanía conceptual a las máquinas de estado finitas empleadas en V^{3}CMM.

Un autómata temporizado es, en esencia, un autómata finito extendido con restricciones temporales descritas en términos de los relojes lógicos del sistema, que son incrementados síncronamente a medida que pasa el tiempo. Estos relojes pueden reiniciarse bajo demanda y sus valores pueden utilizarse tanto en las guardas de las transiciones como para generar ciertos eventos (por ejemplo, de tipo *time-out*). Además, existen varias teorías de análisis y verificación de propiedades basadas en el uso de autómatas temporizados, cuya aplicación puede facilitar el prototipado rápido de aplicaciones cuyas características temporales hayan sido verificadas previamente en tiempo de diseño.

4. Generación de tareas a partir de modelos V³CMM temporizados

En esta sección se detalla el procedimiento desarrollado para obtener modelos de tareas equivalentes a los modelos diseñados con la versión extendida de V³CMM, descrita en la sección anterior. Una vez obtenidos estos modelos de tareas, el objetivo es poder analizar su planificabilidad, empleando para ello alguna de las herramientas de análisis de tiempo real existentes en la actualidad.

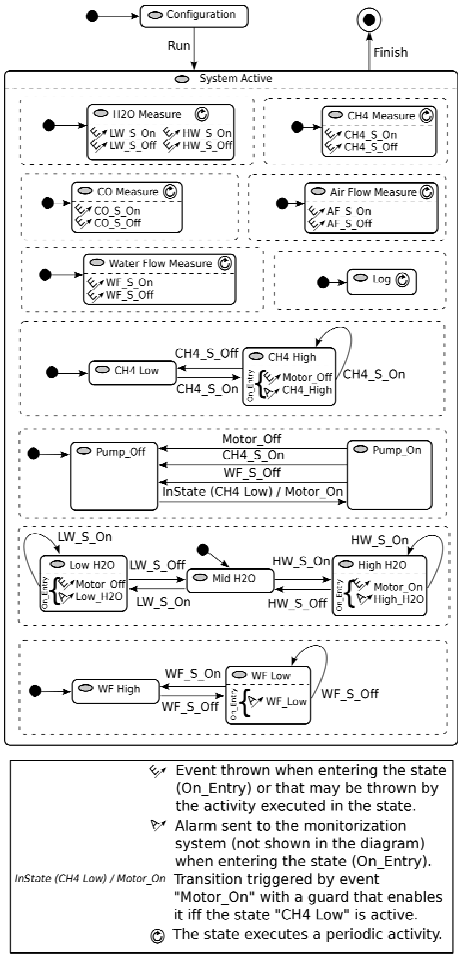

Figura 2. Máquina de estados del caso de estudio

4.1. Descripción del caso de estudio

La aplicación del método propuesto se ilustra mediante el caso de estudio de un sistema de drenaje de una mina. Este ejemplo aparece referenciado en numerosas ocasiones en la literatura relacionada con los sistemas de tiempo real [5]. En la Figura 2 se muestra la máquina de estados asociada al ejemplo. Supondremos que ésta conforma la vista de comportamiento de un único componente V³CMM de alto nivel, que representa al sistema completo.

La función del sistema es bombear a la superficie el agua que se acumula en el pozo de una mina. Para ello, el sistema cuenta con sensores que miden el nivel del agua del pozo (representado por el estado *H2O Measure*). La bomba se enciende cuando el agua alcanza cierto nivel (estado *High H20*) y se detiene al descender por debajo de otro (*Low H2O*). Un requisito importante es el que establece que, cuando la concentración de metano alcanza un determinado nivel (*CH4 High*), la bomba no debe funcionar ya que podría producirse una explosión. El sistema también cuenta con sensores de flujo de agua (*Water Flow Measure, WF High* y *WF Low*), de aire (*Air Flow Measure*) y de monóxido de carbono (*CO Measure*). Todos los eventos son registrados periódicamente en una base de datos (estado *Log*). Los requisitos temporales asociados a cada una de las actividades del sistema quedan recogidos a continuación en la Tabla 1.

Estado	Actividad	C	AT
Configuration	System Init	S	--, 30, 100
H2O Measure	H2O Sensor Polling	H	80, 5, 30
Water Flow Measure	WF Sensor Polling	H	1000, 8, 40
CO Measure	CO Sensor Polling	H	100, 5, 60
CH4 Measure	CH4 Sensor Polling	H	80, 2, 30
Air Flow Measure	AF Sensor Polling	H	100, 12, 100
CH4 Low	--	--	--
CH4 High	Send Alarm	S	--, 2, 30
Pump On	Motor On activity	H	--, 2, 80
Pump Off	Motor Off activity	H	--, 2, 80
Low H2O	Send Alarm	S	--, 4, 30
Mid H2O	--	--	--
High H2O	Send Alarm	S	--, 4, 30
WF High	--	--	--
WF Low	Send Alarm	S	--, 4, 30
Log	Log activity	S	2000, 20, 200

Tabla 1. Actividades asociadas a los estados. *C* (Criticidad): *H* (Hard Real-Time), S (Soft Real-Time), *AT* (Aspectos Temporales): *Periodo, Coste, Plazo*.

4.2. Selección de los perfiles de criticidad

Un sistema suele presentar una combinación de actividades con diferentes grados de criticidad. En el ejemplo, se observa cómo aparecen actividades que deben cumplir sus restricciones temporales de forma estricta (etiquetadas como *H* en la Tabla 1) junto a otras cuyo cumplimiento puede ser más relajado (etiquetadas como *S* en la Tabla 1). Así, el objetivo en este primer paso es particionar el sistema según el nivel de criticidad de sus actividades. De este modo, tras aplicar el procedimiento completo, se obtendrán distintos perfiles de tareas (uno por cada nivel de criticidad considerado), cuya planificabilidad podrá ser analizada aplicando distintos algoritmos, más o menos estrictos.

Cada uno de estos perfiles contendrá las actividades cuyo nivel de criticidad esté por encima de un cierto valor, de modo que, aquéllas que no superen este umbral serán sustituidas por actividades de coste nulo. En el ejemplo, se han considerado dos perfiles: (1) *HRT* (Hard Real-Time), en el que se han incluido todas las actividades marcadas como *H* en la Tabla 1, y (2) *SRT* (Soft Real-Time) en el que se han incluido todas las actividades (tanto *H* como *S*). La creación de estos perfiles se lleva a cabo automáticamente mediante la aplicación de una o más transformaciones Modelo-A-Modelo (M2M). Estas transformaciones conservan las actividades del nivel de criticidad seleccionado y sustituyen las demás por actividades de coste nulo, tal y como se ilustra en la Figura 3.

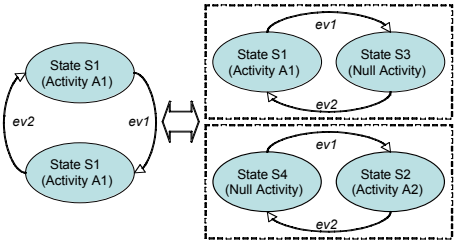

Figura 3. Equivalencia de las regiones ortogonales.

4.3. Búsqueda del Peor Caso

Definimos como peor caso del sistema aquella configuración de estados en la que confluye la ejecución de las actividades más exigentes (en cuanto al tiempo de cómputo) para la plataforma.

Para hallar el peor caso de un determinado perfil, no basta con seleccionar el estado asociado a la actividad de mayor coste en cada región ortogonal, ya que puede ocurrir que ciertas configuraciones de estados no resulten viables dada la lógica de transiciones de la máquina de estados. La búsqueda del peor caso viable para un determinado perfil, puede llevarse acabo aplicando técnicas de *Model Checking*, para lo que existen numerosas herramientas formales como, por ejemplo, Maude [8].

En el ejemplo, el peor caso del perfil HRT ocurre cuando confluyen los estados: *Water Flow Measure, Air Flow Measure, CO Measure, H2O Measure, CH4 Measure* y *Pump On/Off*. En el perfil SRT se añadirían, además, los estados *CH4 High, WF Low* y *Low H2O*.

4.4. Separación de las actividades aperiódicas

Suponiendo un perfil de criticidad concreto, podemos encontrar tanto actividades de carácter periódico como aperiódico. Revisando la literatura afín, se observa que las actividades aperiódicas suponen una clara dificultad para los planificadores de tiempo real, dado que su ocurrencia es no determinista. Por ello, suelen tener un tratamiento especial empleando *Servers*, que las tratan como procesos periódicos [6]. Así, proponemos la separación de las actividades aperiódicas para ser modeladas por separado en función del criterio considerado.

De forma similar al primer paso, consideramos una transformación M2M que, para cada perfil de criticidad, separa aquellos estados que realizan actividades periódicas de aquellos que llevan a cabo actividades no periódicas. A su vez, éstos últimos pueden agruparse por prioridades en distintas regiones ortogonales.

En el caso de estudio, en el perfil HRT sólo habría un estado con una actividad aperiódica asociada (*Motor Activity*). Haciendo uso de un *Server*, ésta sería ejecutada por una tarea periódica de periodo 80, coste 2 y plazo 80. En el caso del perfil SRT, se añadiría otro *Server* de periodo 30, coste 14 y plazo 30 en el que se agruparían todas las actividades *Send Alarm* ya que, como se muestra en la Tabla 1, tienen todas la misma prioridad.

4.5. Generación de esquemas de tareas

A continuación, se procede a identificar las posibles configuraciones de tareas equivalentes al modelo funcional del componente para cada perfil de criticidad. De esta forma, un mismo perfil podría implementarse con distintos conjuntos de tareas. Por ejemplo, sería factible agrupar varias tareas de igual periodo en una sola, que mantendría el periodo y el plazo y cuyo coste se obtendría como la suma de los costes individuales, siempre éste no sobrepasase el plazo fijado.

4.6. Análisis de planificabilidad

Como último paso, tras haber generado las distintas posibles configuraciones de tareas y con el fin de verificar, por ejemplo, si éstas son planificables, podemos hacer uso de alguna de las muchas herramientas de análisis de tiempo real existentes en la actualidad. En nuestro caso, nos hemos decantado por la herramienta Times Tool [18], que toma como entrada un fichero en formato XML en el que se describe el conjunto de tareas a analizar y sus especificaciones temporales. Así, resulta sencillo aplicar una transformación automática Modelo-A-Texto (M2T) que genere estos ficheros XML a partir de los modelos de tareas generados a lo largo del proceso descrito previamente. En la Figura 4 se observa cómo, para una posible configuración de tareas de nivel de criticidad H y considerando un planificador basado en el algoritmo *Rate Monotonic,* el análisis se satisface. Sin embargo, tal y como se muestra en la Figura 5, si este mismo esquema se aplica a otra configuración de tareas de nivel de criticidad S, el análisis no se satisface.

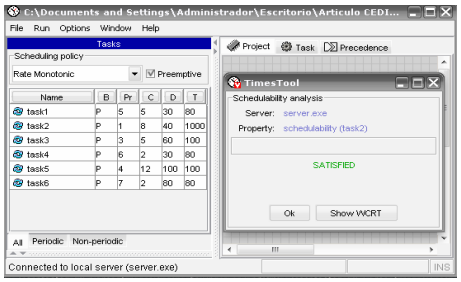

Figura 4. Análisis HRT con Times Tool

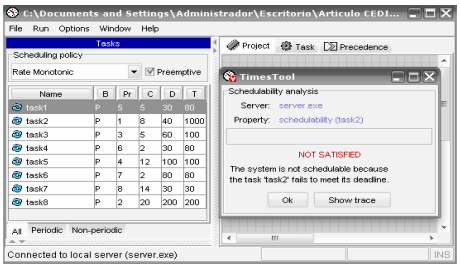

Figura 5. Análisis SRT con Times Tool

5. Trabajos relacionados

En los últimos años, el modelado y análisis de sistemas de tiempo real empleando un enfoque de DSDM ha dado lugar al desarrollo de numerosos proyectos de carácter internacional [10][17]. En este sentido, Burmester propone en [4] una visión similar a la planteada en este trabajo, en la que se parte de la especificación del sistema desde una perspectiva independiente de la plataforma (con diagramas de componentes y máquinas de estado anotadas con requisitos temporales) y se plantea como objetivo verificar las restricciones temporales a alto nivel para realizar un proceso de división y mapeo de las especificaciones a modelos dependientes de la plataforma. A diferencia de este trabajo, centrado en obtener una implementación del sistema consistente con los requisitos temporales incluidos en los modelos de diseño, el trabajo que se presenta en este artículo persigue la verificación temprana de los requisitos temporales utilizando para ello los mecanismos necesarios para lograr la interoperabilidad con algunas de las herramientas de análisis de tiempo real ya existentes.

Entre las aproximaciones asociadas al análisis de los aspectos temporales en sistemas basados en componentes, cabe destacar [11] y [13]. Gu Z., et al. proponen una extensión de los modelos Rational Rose con la información necesaria para el despliegue del sistema y un método para distribuir los componentes automáticamente en múltiples procesadores garantizando su planificabilidad. No obstante, esta aproximación queda limitada, dado que no aborda la sincronización entre componentes.

La extensión temporal de los diagramas de estado UML, conocida como *Hierarchical Timed*

Automata (HTA) [9], proporciona un formalismo que permite la validación y la verificación de sistemas de tiempo real con UPPAAL [14]. Relacionadas con UPPAL, se han desarrollado diversas herramientas como Times Tool [18] (para el modelado, análisis de planificabilidad y síntesis de código de sistemas empotrados), o CATS [7] (para la composición temporal y el análisis del rendimiento de sistemas de tiempo real).

Por último, también cabe destacar el estándar MARTE [16], del *Object Management Group* (OMG), que define un perfil UML para el modelado de sistemas empotrados con restricciones de tiempo real, así como la existencia de ciertas metodologías orientadas al desarrollo de sistemas de tiempo real, tales como UML-MAST [19] o HRT-HOOD [5].

6. Conclusiones y trabajos futuros

El DSDM proporciona los mecanismos necesarios para definir nuevos lenguajes de modelado, precisos y completamente adaptados a cada dominio de aplicación. En este sentido, V^{3}CMM aprovecha las bondades de este enfoque, proporcionando un lenguaje independiente de la plataforma para modelar sistemas basados en componentes. En este trabajo hemos abordado la extensión de este lenguaje para permitir el modelado de las restricciones de tiempo real que pueden tener asociados ciertos componentes del sistema. También se ha presentado un mecanismo que permite transformar los modelos de máquinas de estado temporizadas, descritos con V^{3}CMM en modelos de tareas equivalentes, para su análisis temprano y formal utilizando algunas de las herramientas existentes para ello en la actualidad. Dado que se trata de un trabajo preliminar, algunos aspectos deben ser completados y mejorados. Por ejemplo, la extensión del análisis considerando la necesaria sincronización entre componentes, o el desarrollo del análisis probabilístico para los perfiles SRT, considerando la distribución de probabilidad asociada a los distintos eventos, la variabilidad asociada al tiempo de ejecución de las actividades, o la confluencia estadística de actividades aperiódicas.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el proyecto EXPLORE (TIN2009-08572).

Referencias

[1] Alonso D., et al., "*V^{3}CMM: a 3-View Component Meta-Model for Model-Driven Robotic Software Development*", Journal of Software Engineering for Robotics (JOSER), 1(1), 3–17, 2010.

[2] Baier C. and Katoen J., "*Principles of Model Checking*", The MIT Press, 2008.

[3] Bézivin J., "*On the unification power of models*", Journal of Systems and Software, 4(2), 171–188, 2005.

[4] Burmester S., et al., "*Model-Driven Architecture for Hard Real-Time Systems: From Platform Independent Models to Code*", European Conf. on Model Driven Architecture - Foundations and Applications, 2005.

[5] Burns A. and Wellings A., "*HRT-HOOD: A Structured Design Method for Hard Real-Time Ada Systems*", Real-Time Safety Critical Systems, vol.3, ed. Elsevier, 1995.

[6] Buttazzo G., "*Hard Real-time Computing Systems. Predictable Scheduling Algorithms and Applications*", 2nd ed. Kluwer, 2000.

[7] CATS, http://www.timestool.com/cats

[8] Maude, http://maude.cs.uiuc.edu

[9] David A., et al., "*Formal Verification of UML Statecharts with Real-Time Extensions*", 5th Int'l Conf. on Fundamental Approaches to Software Engineering (FASE), 2002

[10] FRESCOR, http://www.frescor.org/

[11] Gu Z., et al., "*A Model-Based Approach to System-Level Dependency and Real-Time Analysis of Embedded Software"*, 9th IEEE Real-Time Embedded Tech. and App., 2003.

[12] Jin X. et al., *"Real-Time Component Composition Using Hierarchical Timed Automata"*, Proc. 7th Int'l Conf. on Quality Software (QSIC 2007), 2007.

[13] Kim G. Larsen, et al. "*Uppaal in a Nutshell"*. Int. Journal on Software Tools for Technology Transfer, 1(1-2), 134–152, 1997.

[14] Lau K. and Wang Z., "*Software component models*", IEEE Trans. Software Eng., 33(10), 709–724, 2007.

[15] MARTE, http://www.omgmarte.org

[16] QUASIMODO, www.quasimodo.aau.dk

[17] Times Tool. http://www.timestool.com

[18] UML-MAST, http://mast.unican.es/umlmast