IX Jornadas de Ciencia y Tecnología
de Facultades de Ingeniería del NOA
Santiago del Estero, 3 y 4 de Octubre de 2013

Modelado de sistemas de Tiempo Real con perfil MARTE: Un caso de estudio.

Patricio O. Condorí[1], Pablo A. Vilte[1] & María del P. Galvez[1]

(1) Facultad de Ingeniería, Universidad Nacional de Jujuy.
patricioomarcondori@hotmail.com
mpgalvezd@fi.unju.edu.ar

RESUMEN: El presente trabajo expone conceptos elementales del perfil UML MARTE para el modelado y análisis de sistemas de tiempo real y embebidos, de su estructura y, fundamentalmente del paquete de análisis de modelos. Se describe un caso de estudio: sistema de garaje inteligente, y se presentan distintos diagramas UML con notaciones MARTE que los proveen con información cuantitativa para un posterior análisis de rendimiento del sistema.

1 INTRODUCCIÓN

Los lenguajes de modelado para desarrollar sistemas tradicionales no son apropiados para trabajar con Sistemas de Tiempo Real (STR), porque adolecen de especificidad para resolver problemas, como la interacción con los dispositivos físicos, la concurrencia del sistema, la gestión del tiempo o el análisis de planificabilidad y de rendimiento.
Por ello, es importante examinar cómo se complementan los lenguajes de modelado generales para abordar el desarrollo de STR permitiendo gestionar su complejidad intrínseca y apoyar el análisis cuantitativo con el fin de validar sus propiedades no funcionales.
El paradigma orientado a objetos es muy eficaz para hacer frente a la complejidad del software y promueve la descomposición y la abstracción en el proceso de diseño e implementación de sistemas. En este sentido, el modelado orientado a objetos aparece como una herramienta potente para capturar las diferentes características y requisitos de los STR.
Debido a que el lenguaje unificado de modelado (UML) se ha convertido en la notación dominante para el análisis y diseño orientados a objetos, y que una de sus características importantes es su capacidad de adaptarse a las particularidades de un sistemas específico mediante sus mecanismos de extensibilidad incorporados: estereotipos, valores etiquetados y restricciones (agrupados en Perfiles), se considera que UML es una buena alternativa para modelar y diseñar STR.
MARTE es el actual perfil que adapta UML, no solo para el diseño de este tipo de sistemas, sino también para el análisis basado en modelos.

Este trabajo tiene como objetivo presentar el perfil MARTE y su aplicación en el modelado del análisis de rendimiento del caso de estudio SGI: Sistema de Garage Inteligente.
En el apartado 2 se expone el concepto de STR, se describen brevemente los Sistemas Embebidos de Control de Tiempo Real y su modelado, en el apartado 3 se expone el perfil UML detallando sus mecanismos de extensión, en el apartado 4 se presentan las características y estructura del perfil MARTE, en el apartado 5 se describe el caso de estudio SIG y se presentan los diagramas UML con notaciones MARTE para análisis de rendimiento del sistema, en el apartado 6 se presentan las conclusiones y trabajo futuro y en el apartado 7 las referencias.

2 SISTEMAS DE TIEMPO REAL

Un STR es un sistema informático en el cual su buen funcionamiento depende no solo de la corrección lógica de los algoritmos y de las respuestas obtenidas, sino también del momento en que éstas están disponibles (Stankovic, 1993). Por esto, para considerar que el sistema está operando apropiadamente es obligatorio que proporcione los resultados dentro de un intervalo de tiempo conocido.

2.1 *Sistemas de Control Embebidos de Tiempo Real*

Los Sistemas de Control Embebidos de Tiempo Real (SCETR) están diseñados para llevar a cabo una función específica y suelen estar limitados en cuanto a los recursos disponibles y no suelen ser visibles. En general son sistemas de computación con un hardware acoplado herméticamente y

software integrado. Sus funciones son (Buttazzo, 1997):
Monitoreo: obtienen información acerca del estado actual del entorno físico del sistema.
Control: realizan los cálculos necesarios para permitir controlar el proceso de acuerdo a los valores leídos.
Actuación: modificar el estado actual del entorno físico del sistema

2.2 *Modelado de STR*

Con el aumento de la complejidad de los sistemas informáticos, y en especial los STR, se ha pasado de la preocupación que había por cuestiones de implementación, a dedicar más tiempo y esfuerzo al diseño y modelado de sistemas.
Esta complejidad se basa en dos cuestiones fundamentales: la construcción de la solución correcta a un sistema complejo no se puede llevar a cabo a través de modelos mentales y es difícil establecer a priori si el sistema funcionará correctamente una vez construido, lo que es especialmente grave en aquellos sistemas cuyo costo es muy elevado, o llevan a cabo tareas muy delicadas o peligrosas. Por esta razón, es imprescindible el uso de modelos en la rama de Ingeniería del Software, ya que permiten describir y representar los modelos mentales en modelos teóricos intuitivos, abordando sus conceptos más relevantes antes de proponer soluciones de implementación, como también permiten realizar un análisis previo de las características y funcionamiento del sistema (Akhalaki, 2008).
Aspectos como las restricciones temporales, la concurrencia, la criticidad, la sincronización y coordinación de tareas, la interacción con el hardware, son algunos aspectos importantes a tener en cuenta a la hora de modelar y diseñar STR. Para ello se hace necesario formular modelos que, además de describir la funcionalidad del sistema, permitan representar los aspectos propios de los STR, y fundamentalmente, predecir y evaluar cualitativa y cuantitativamente el comportamiento temporal de estos sistemas, esto es hacer predecibles los instantes en que se producen sus respuestas (López, 2010).

3 PERFIL UML

El lenguaje de modelado UML es el estándar más utilizado para especificar y documentar cualquier sistema de forma precisa, sin embargo, el hecho de que sea una notación de propósito general obliga a que muchas veces sea deseable poder contar con algún lenguaje más específico para modelar y representar los conceptos de ciertos dominios particulares. Los Perfiles UML constituyen el mecanismo que proporciona UML para extender su sintaxis y su semántica para expresar los conceptos específicos de un determinado dominio de aplicación.
Un Perfil UML es un paquete que se apoya en los mecanismos de extensión que se definen en el metamodelo de UML: estereotipos, valores etiquetados y restricciones, que se aplican a cualquier elemento de los modelos UML, proveyéndolos de una nueva semántica y adaptándolos a dominios, procesos o plataformas específicos (Fuentes et al., 2007).

3.1 *Mecanismos de extensión UML*

Estereotipos*:* permiten extender el vocabulario de UML con nuevos elementos de construcción que representan conceptos específicos de un dominio en particular. Gráficamente, un estereotipo se representa como un nombre con comillas <<estereotipo>>.
Valores etiquetados*:* extienden las propiedades de los estereotipos aplicados en los elementos del modelo, añadiendo nueva información. Se pueden modelar solamente como atributos definidos para estereotipos y se representan como una cadena encerrada entre llaves, por ejemplo: {versión=3}.
Restricciones: representan un conjunto de propiedades que definen las condiciones que deben cumplir los elementos de modelado para que estén correctos.

4 MARTE

En 2009 el OMG (Grupo de Administración de Objetos) emite la primera versión oficial de este Perfil, MARTE 1.0 (Modelado y Análisis de Sistemas de Tiempo Real y Embebidos (STR/E)). Actualmente, está en vigencia desde junio de 2011 la versión MARTE 1.1 (MARTE, 2011).
Este Perfil añade capacidades de UML para el desarrollo de modelos de STR/E y proporciona soporte durante las etapas de especificación, diseño y verificación/validación.
MARTE proporciona construcciones para el diseño de STR/E y para el análisis basado en modelos.

4.1 *Características de MARTE*

Las características con las cuales MARTE fue especificado son muy variadas, y permiten que los diferentes usuarios del Perfil, cuenten con las siguientes posibilidades (MARTE, 2011), (Serna, 2011):

- Permite construir modelos que caractericen los STR/E en forma cualitativa

(comunicación, sincronización, paralelismos, etc.) o cuantitativa (plazos, periodicidad, etc.).

- Permite realizar análisis, en fases tempranas del ciclo de desarrollo, de las características del software y el hardware en STR/E en forma cuantitativa y segmentada.
- Proporciona un marco común de modelado de los aspectos hardware y software (en forma independiente) de un STR/E con el fin de mejorar la comunicación entre los desarrolladores.
- Permite la interoperabilidad entre las herramientas de desarrollo utilizadas para la especificación, diseño, verificación, generación de código, simulación.
- Fomenta la construcción de modelos que se pueden utilizar para realizar predicciones cuantitativas acerca de las características de STR/E.
- Permite que los modeladores apliquen diferentes técnicas de análisis, sin necesidad de que tengan un profundo conocimiento de las mismas.

4.2 *Estructura de MARTE*

Como muestra la Fig.1, MARTE se estructura en tres paquetes principales (MARTE, 2011), (Serna, 2011):

1) Paquete MARTE Foundations: define los conceptos, para STR/E, necesarios para el modelado de aplicaciones genéricas y artefactos de la plataforma. Está integrado por los siguientes subpaquetes: Elementos Básicos (CoreElements), Modelado de Propiedades No Funcionales (NFP), Modelado del Tiempo (Time), Modelado de Recursos Genéricos (GRM) y Modelado de Asignación (Alloc).
2) Paquete MARTE Design Model: proporciona soporte necesario para una variedad de actividades, desde la especificación hasta el diseño detallado de STR/E. Se compone de los siguientes sub-paquetes: Modelado Genérico de Componentes (GCM), Modelado de Aplicaciones de Alto nivel (HLAM) y Modelado Detallado de Recursos (DRM).
3) Paquete MARTE Analysis Model: ofrece conceptos para anotar modelos UML con información cuantitativa para un posterior análisis de los mismos. Se compone de tres sub-paquetes: Modelado de Análisis Cuantitativo Genérico (GQAM), Modelado de Análisis de Planificabilidad (SAM) y Modelado de Análisis de Rendimiento (PAM).

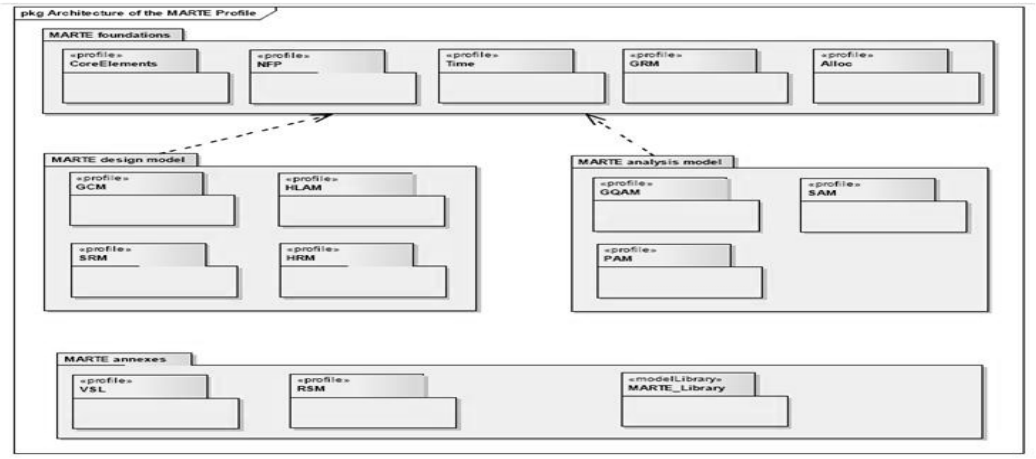

Figura 1: Estructura del Perfil UML MARTE

4.3 *Modelo de análisis con MARTE*

Existen básicamente dos enfoques para evaluar el rendimiento de un sistema: estudiar un modelo del sistema o medirlo una vez implantado. Ambos enfoques son complementarios: un modelo analítico reduce el riesgo de implementar una arquitectura de software ineficiente, y medir el rendimiento del sistema una vez implementado proporciona mayor precisión y nivel de detalle.

MARTE permite realizar análisis de rendimiento y de planificabilidad, pero su modelo de análisis es tan flexible que permite definir otros como consumo de energía, uso de la memoria, confiabilidad, y seguridad. Es posible realizar evaluaciones precisas y fehacientes de modelos de STR/E, utilizando análisis cuantitativo formal basado en modelos matemáticos, lo cual permite al diseñador detectar problemas en etapas previas

del desarrollo, reduciendo costos y riesgos asociados al proyecto.

4.4 *Modelado para el Análisis de Rendimiento con MARTE*

El modelado del sistema para análisis de rendimiento, debe permitir estimar el rendimiento de una instancia del sistema, de acuerdo al modelo del sistema, e igualmente, debe facilitar la mejora del mismo, determinando tiempos de respuesta, plazos no cumplidos, recursos utilizados, tamaño de las colas, recursos críticos (MARTE, 2011), (Serna, 2011). Para modelar el rendimiento con MARTE, se usan una variedad de conceptos propios del perfil (representados como estereotipos y valores etiquetados en UML). Las Tablas 1 y 2, muestran las notaciones MARTE empleadas en el modelado de rendimiento del caso de uso que se expone en el apartado 5.4.

Tabla 1: Conceptos del Perfil MARTE en la vista del dominio y en la representación UML.

Conceptos MARTE	Estereotipos UML
BehaviorScenario	<<GaScenario>>
WorkloadBehaviour	<<GaWorkloadBehaviour>>
Step	<<GaStep>>
AnalysisContext	<<GaAnalysisContext>>
Resources	<<Resources>>
CommunicationChannel	<<GaCommChannel>>
CommunicationStep	<<GaCommStep>>
ExecutionHost	<<GaExecHost>>
ResourcesPlatform	<<GaResourcesPlatform>>
WorkloadEvent	<<GaWorkloadEvent>>

Tabla 2: Propiedades para analizar el rendimiento.

NFP (valor etiquetado)	Step (estereotipo)
repetition: NFP_Real[*]	Número de veces que el step se repite una vez disparado.
probability: NFP_Real[*]	Probabilidad de que el step sea ejecutado.
hostDemand: NFP_Duration[*]	Demanda de CPU en unidades de tiempo.
hostDemandOps: NFP_Real[*]	Demanda de CPU en unidades de operaciones.
priority: NFP_Integer[*]	Prioridad de un step en el host.
respTime: NFP_Duration[*]	Tiempo que tarda la ejecución de un step desde el evento disparador hasta que termina.
execTime: NFP_Duration[*]	respTime menos cualquier retardo.
troughput: NFP_Frequency[*]	Frecuencia promedio a la que se inicia un step.
utilizationOnHost: NFP_Real[*]	Fracción de tiempo en que el host está ocupado ejecutando un step.
blockingTime: NFP_Duration[*]	Retardo del step.

5 CASO DE ESTUDIO: SISTEMA DE GARAJE INTELIGENTE (SGI)

El SGI, es un sistema de control embebido de tiempo real que controla la apertura y cierre de un portón automático y la luminosidad de un garaje (Fig.2).

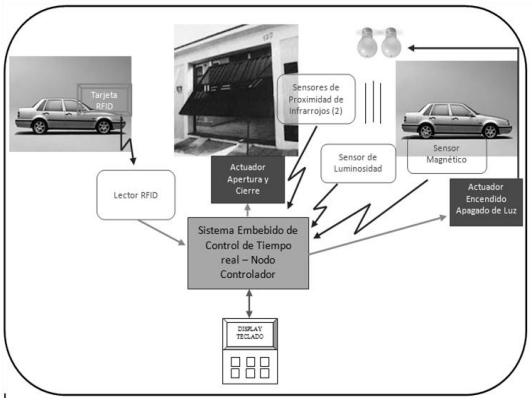

Figura 2: Sistema de Garaje Inteligente

5.1 *Arquitectura del SGI*

La arquitectura del sistema embebido de tiempo real para el caso de estudio SGI, se estructura a partir de dispositivos que pueden clasificarse en los siguientes grupos:

- Controladores*:* dispositivos que gestionan el sistema. Son los encargados de llevar el control del sistema: procesar la información enviada por los sensores, tomar decisiones y transmitir los comandos de control necesarios a los actuadores.
- Sensores*:* dispositivos que monitorean el entorno captando información que transmiten al sistema.
- Actuadores*:* dispositivos capaces de ejecutar y/o recibir una orden del controlador y realizar una acción sobre un dispositivo o ambiente sobre el cual actúa el sistema.
- Buses*:* medio de transmisión que transporta la información entre los distintos dispositivos finales.
- Interfaces*:* son los dispositivos y los formatos en que se muestra la información del sistema para los usuarios (u otros sistemas) y donde los mismos pueden interactuar con el sistema.

Como muestra la Fig.3, el SGI está compuesto básicamente por:

- Un controlador.
- Una interfaz usuario (pantalla LCD y teclado).
- Dispositivos terminales sensores:
  - Un Lector RFID: que lee el código inserto en una etiqueta RFID ubicada en el parabrisas de un automóvil.

- Dos Sensores de Proximidad: que monitorean el cruce de un automóvil a través del portón.
- Un Sensor de Luminosidad: que lee el nivel de luminosidad del interior del garaje.
- Un Sensor Magnético; que detecta el ingreso, encendido de motor y egreso del automóvil a través de la variación del campo magnético en el ambiente.

- Dispositivos terminales actuadores:
  - Un dispositivo motor: que realiza la apertura y cierre del portón (Actuador A/CP).
  - Un contactor: que enciende y apaga las luces (Actuador E/AL).

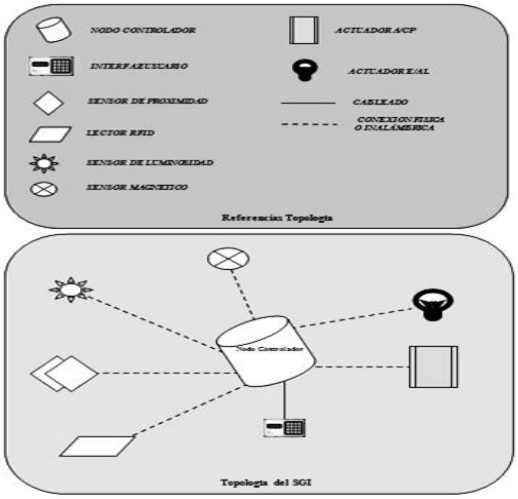

Figura 3: Topología del SGI

5.2 *Requisitos funcionales del SGI*

Los requerimientos funcionales del SGI se pueden clasificar de acuerdo a tres procesos:

1. Ingreso del automóvil al garaje

El sistema debe monitorear la presencia de automóviles con etiqueta RFID. Cuando detecte y valide el código de una etiqueta RFID debe:

- Activar el sensor de luminosidad, hacer lectura del porcentaje de luminosidad en el interior del garaje.
- Activar el Actuador E/AL y encender las luces, si el nivel de luminosidad es menor al 70%.
- Activar el Actuador A/CP e iniciar la apertura del portón.
- Monitorear el cruce del automóvil a través del portón.
- Detectar la presencia del vehículo en el interior del garaje.
- Activar el Actuador A/CP e iniciar el cierre del portón, cuando el vehículo cruza totalmente el portón.
- Interrumpir el cierre e invertir el proceso (abrir el portón), si durante el proceso de cierre del portón algún objeto (un niño, un animal, el automóvil, etc.) se interpone.
- Activar el Actuador E/AL y apagar las luces, después de 30 s del cierre del portón.

2. Egreso del automóvil del garaje

El sistema debe monitorear el interior del garaje. Cuando detecte el encendido del motor de un automóvil debe:

- Activar el sensor de luminosidad, hacer lectura del porcentaje de luminosidad en el interior del garaje.
- Activar el Actuador E/AL y encender las luces, si el nivel de luminosidad es menor al 70%.
- Activar el Actuador A/CP e iniciar la apertura del portón.
- Monitorear el cruce del automóvil a través del portón.
- Detectar la ausencia del vehículo en el interior del garaje.
- Activar el Actuador A/CP e iniciar el cierre del portón, cuando el vehículo cruza totalmente el portón.
- Interrumpir el cierre e invertir el proceso (abrir el portón), si durante el proceso de cierre del portón algún objeto (un niño, un animal, el automóvil, etc.) se interpone.
- Activar el Actuador E/AL y apagar las luces, después de 30 s del cierre del portón.

5.3 *Requisitos no funcionales del SGI*

Requisitos Temporales

- El lector RFID y el sensor magnético monitorearán el ambiente en ciclos de 50 ms.
- Los sensores de proximidad y de luminosidad monitorearán a solicitud del controlador.
- Las lecturas de los sensores y el lector RFID se realizarán en menos de 20 ms.
- La activación y desactivación de los sensores y lector RFID se realizarán en menos de 20 ms.
- La activación del Actuador A/CP se realizará en menos de 20 ms.
- La activación del Actuador E/AL se realizará en menos de 20 ms.
- El controlador monitoreará el cruce a través del portón en ciclos de 500 ms.
- Una vez autorizado el ingreso o egreso del automóvil, el sistema debe garantizar que:
  - La Apertura y el cierre del portón se realice en 10 s como máximo.
- Las luces deben ser apagadas a los 30 s del cierre del portón.
- El tiempo de cruce a través del portón es de 30 s.

5.4 *Modelado del SGI con notaciones del perfil MARTE*

En este apartado se expone el modelado de análisis de rendimiento del SGI, estos es, diagramas con notaciones MARTE que pueden ser usados para un análisis de rendimiento del sistema.

5.4.1 *Diagrama de despliegue con notación MARTE para el análisis de rendimiento*

En la Fig.4 se observa que el contexto de rendimiento del sistema hace énfasis en la cantidad de microcontroladores, *in$NMicroc*, Lectores RFID *in$NLRFID*, Sensores de Proximidad, *in$NSP*, Sensores de Luminosidad, *in$NSL*, Sensores Magnéticos, *in$NSM*, Actuadores: *in$NAACP* y *in$NAEAL*, así mismo, se especifica el tamaño máximo de la trama de comunicación entre el coordinador y los dispositivos finales, *in$maxSizeMsg*.

El estereotipo, <<GaAnalysisContext>>, que reúne la especificación del contexto de rendimiento, tiene la siguiente forma:

<<GaAnalysisContext >>
{contextParams={in$NMicroc, in$NLRFID, inNSP, inNSL, inNSM, inNAACP, in$NAEAL, in$maxSizeMsg}, paramValues={1,1,2,1,1,1,1,(128,Bytes)}}

En esta especificación se indica que el contexto de despliegue del SGI tendrá 1 microcontrolador, 1 lector RFID, 2 sensores de proximidad, 1 sensor de luminosidad, 1 sensor magnético, 1 actuador para apertura/cierre de portón, 1 actuador para encendido/apagado de luces, y la trama de comunicación entre el microcontrolador y los dispositivos finales tendrá un tamaño máximo de 128 Bytes.

El nodo *Microcontrolador* etiquetado con el estereotipo <<GaExecHost>>, en el cual se ejecuta el software del sistema, contiene los artefactos <<artefact>>: para la planificación del SGI, gestión de los timers, comunicación mediante el protocolo ZigBee, control de los sensores y actuadores y para controlar el SGI. El estereotipo <<GaExecHost>> representa un procesador u otro dispositivo que ejecuta operaciones especificadas en el modelo.

El estereotipo <<GaCommHost>> representa el medio de comunicación entre el microcontrolador y los sensores y actuadores.

Los sensores y actuadores están etiquetados con los estereotipos <<HW_Sensor>> y <<HW_Actuator>>, y conectados al microcontrolador en forma inalámbrica mediante la tecnología ZigBee (ZigBee, 2010), el canal de comunicación, por limitaciones físicas del estándar IEEE802.15, tiene una capacidad de 250Kb/s. La latencia de la WPAN es típicamente del orden de los 20ms. Las conecciones están etiquetadas con los estereotipos <<ZigBee>> y <<RS-232>>.

Los dispositivos finales y el microcontrolador tienen propiedades de sobrecarga de transmisión y recepción de mensajes, dadas en ms/KB.

El teclado y la pantalla se comunican con el microcontrolador con el protocolo RS-232 una velocidad de transmisión de 200Mb/s.

Estas notaciones MARTE son útiles para modelar y analizar el rendimiento del sistema.

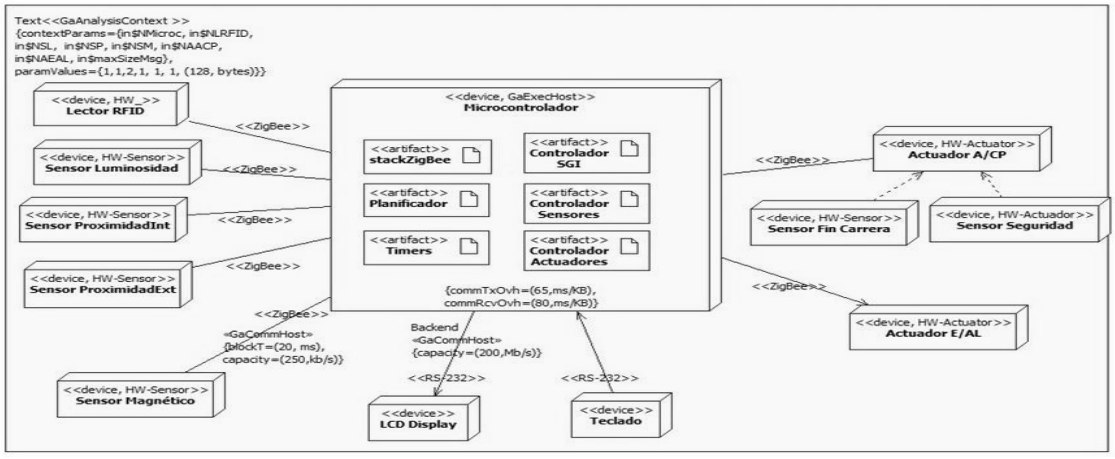

Figura 4: Diagrama de despliegue del SGI con notación MARTE para análisis de rendimiento

5.4.2 *Diagrama de actividades con notación MARTE para el análisis de rendimiento*

El diagrama de actividades de la Fig.5 detalla los requisitos de tiempo del SGI que se definieron para cumplir la función de monitoreo del portón. El estereotipo <<GaWorkloadEvent>> dispara un nuevo evento que inicia el monitoreo en forma periódica, cada 500 ms. *GaWorkloadEvent* sólo produce el evento disparador del monitoreo, por lo tanto, las actividades que deben ejecutarse previas al evento, no deben tardar más que el periodo del *GaWorkloadEvent*.

Las actividades en este diagrama están estereotipadas con <<GaScenario>>, de esta forma, es posible asociarles atributos definidos en MARTE, que son útiles para modelar y analizar rendimiento. Los escenarios del <<loop>>, son aquellos que se ejecutarán periódicamente mientras se detecte que el automóvil está cruzando a través del portón. Estos escenarios tienen *respT* (tiempos de respuesta), definidos en los requisitos temporales, que permiten verificar el cumplimiento del periodo de monitoreo del portón. Algunos escenarios también tienen asociados el atributo *throughput*, el cual garantiza que estos escenarios deben ejecutarse 2 veces por segundo.

Los tiempos de respuesta de los escenarios *Iniciar Timer Monitoreo*, *Leer Sensores Proximidad* (40 ms y 20 ms, respectivamente) y *Evaluar Cruce* (10 ms), no deben en conjunto sumar un tiempo de respuesta mayor a 500 ms. En este caso, de acuerdo a los requisitos de tiempo especificados, el tiempo total es menor que el requerido por cada evento del *GaWorkloadEvent.*

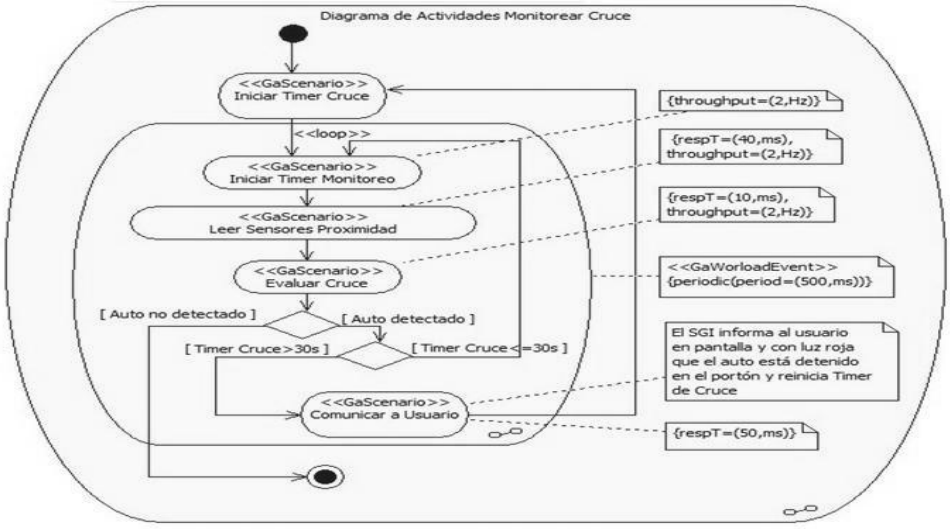

Figura 5: Diagrama de actividades Monitorear Cruce utilizando restricciones de tiempo – MARTE

5.4.3 *Diagrama de secuencia con restricciones temporales MARTE*

La Fig.6 muestra el diagrama de secuencia Abrir Portón con extensiones MARTE. El diagrama muestra el proceso normal de apertura del portón. Para este propósito, el diagrama de secuencia muestra las clases Controlador SGI y Actuador A/CP.

El diagrama de secuencia se concentra en la representación del tiempo. Para esto se aplica el estereotipo <<timedConstraint>> que permite especificar la restricción de la duración de la ejecución de los métodos, para un posterior análisis de rendimiento del sistema.

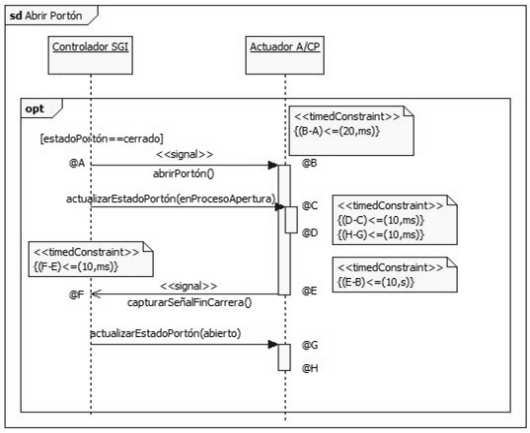

Figura 6: Diagrama de secuencia con restricciones temporales – MARTE

6 CONCLUSIONES Y TRABAJO FUTURO

- MARTE es un perfil que permite el modelado de las propiedades específicas de los STR/E y promueve el análisis cuantitativo de modelos para validar las NFPs de estos sistemas.
- MARTE es un Perfil muy potente y complejo, y la aplicación de todos sus elementos en un proyecto es improbable. Dependerá de los aspectos que se quieran abordar, para decidir qué paquetes de MARTE usar.
- La aplicación industrial de MARTE es aún muy incipiente y no muy conocida. Es necesario un análisis más profundo sobre su aplicabilidad, en el marco de futuros trabajos.
- El SGI necesita cumplir sus requisitos de rendimiento. Por lo cual es necesario evaluar el sistema analizando los modelos anotados con MARTE, actividad a realizarse en un trabajo futuro.
- El producto obtenido en el caso de estudio: especificación de requisitos, modelado de análisis de rendimiento con MARTE, es una guía de gran utilidad para futuras actividades de esta naturaleza, ya que es insuficiente la bibliografía disponible en el ámbito académico u otros medios, que contengan ejemplos prácticos de aplicación de este perfil.

7 REFERENCIAS

Akhalaki, K. B., *Medistam-RT: Metodología de Diseño y Análisis de Sistemas de Tiempo Real,* Tesis Doctoral, Universidad de Granada, 2008.

Álvarez Palomo, J. M., *Desarrollo de Software para Sistemas de Tiempo Real basado en UML. Un Enfoque Formal basado en Metamodelado,* Tesis Doctoral, Universidad de Málaga, España, 2006.

Burns, A & A. Wellings, *Sistemas de Tiempo Real y Lenguajes de Programación,* Tercera edición, Editorial: Addison, Wesley, 2003.

Buttazzo G., *Hard real-time computing system*, Scuola Superiore S. Anna, Kluwer Academic Publishers, 1997.

Douglass, B. P. *Real Time UML Third Edition: Advances in The UML for Real-Time Systems,* Editorial Addison-Wesley, Boston, 2007.

El poder semántico de UML 2.0 en la práctica, http://www.epidataconsulting.com/tikiwiki/tiki-read_article.php?articleId=31#**,** noviembre de 2012.

Fuentes, L. & A. Vallecillo, *Una Introducción a los Perfiles UML*, Depto. de Lenguajes y Ciencias de la Computación, Universidad de Málaga, Campus de Teatinos, E29071- Málaga (SPAIN), 2007.

López Martínez, P, *Desarrollo de sistemas de tiempo real basado en componentes utilizando modelos de comportamiento reactivos,* Tesis Doctoral, Universidad de Cantabria, Granada, 2010.

MARTE OMG, *UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems. Version 1.1*, OMG Document Number: formal/2011-06-02, June 2011. Standard document URL: http://www.omg.org/spec/MARTE/1.1/

Medina, J. L., *Metodología y Herramientas UML para el Modelado y Análisis de Sistemas de Tiempo Real Orientados a Objetos,* Tesis Doctoral, Pasaje Santander, 2005.

OMG Unified Modeling Language (OMG UML) Versión 2.4.1, http://www.omg.org/spec/UML/2.4.1/**.**

Selic, B. & J. Rumbaugh, *Using UML for modeling complex real-time systems,* Rational white paper, 1998.

Serna, S., *Especificación formal de requisitos temporales no funcionales*, Facultad de Minas, Universidad Nacional de Colombia. 2011.

Stankovic, J. & k. Ramamritham, *What is the Predictability for Real-Time System? Advances in Real-Time Systems*, Edited by: John A. Stankovic and Kriti Ramamritham, California. 1993.

Wainer, G. A. *Sistemas de Tiempo Real, Conceptos y Aplicaciones,* Editorial Nueva Librería, 2002.

ZigBee, http://www.zigbees.com/, noviembre 2012.