XI JORNADA TÉCNICA DE ADA-SPAIN

DOCUMENTACIÓN

Miércoles 9 de abril de 2003
Salón de Actos del edificio López Araújo (C)
ETSI de Telecomunicación
Universidad Politécnica de Madrid

El "*UML Profile for Schedulability, Performance and Time*" en la representación de sistemas distribuidos de tiempo real[1]

Julio L. Medina, Michael González Harbour y José M. Drake

Departamento de Electrónica y Computadores, Universidad de Cantabria
Av. de los Castros s/n 39005 - Santander, ESPAÑA
{medinajl, mgh, drakej}@unican.es

RESUMEN

El Open Management Group (OMG) ha aprobado la especificación final del "*UML Profile for Schedulability, Performance and Time"* [15][14] y se encuentra ya en las últimas etapas del proceso para su estabilización como especificación formal. En este trabajo se describen algunos aspectos de este perfil que han sido reportados como mejorables y en particular se detallan los problemas y propuestas de solución para aquellos en los que se ha encontrado conflicto al pretender aplicar el perfil en la representación de modelos de análisis de planificabilidad para sistemas distribuidos de tiempo real.

1. INTRODUCCIÓN

En este documento plantea una serie de observaciones [1] sobre el *"UML Profile for Schedulability, Performance and Time"* [15][14], llamado perfil RT-UML[2], que fue propuesto por un grupo de empresas miembros del *Open Management Group* (OMG) en agosto de 2000, revisado en junio de 2001 y ha sido aceptado como especificación final en marzo de 2002, ello en respuesta a la solicitud de propuestas (RFP) lanzada por el *"Analysis and Design Platform Task Force"* del OMG en marzo de 1999 (OMG document ad/99-03-13). Este perfil se encuentra en fase de finalización, a la espera de resolver un número de puntos en cuestión que son aún objeto de debate [2].

Estas observaciones parten de la experiencia en el desarrollo del entorno MAST (*Modeling and Analysis Suite for Real-Time Applications*) [4][19], que es una metodología de modelado y una herramienta para el análisis de planificabilidad de sistemas distribuidos de tiempo real. Con ella se ha definido la herramienta UML-MAST [11][3] que formula los modelos de análisis utilizando UML y puede ser soportada desde cualquier herramienta CASE orientada a este lenguaje de modelado.

Aunque el ámbito del perfil RT-UML es más amplio que el que cubre el entorno MAST, los aspectos considerados por MAST quedan dentro del espectro de aplicación del RT-UML y quedan reflejados en varios de sus sub-perfiles, por lo que al desarrollar el meta-modelo de la herramienta UML-MAST se ha procurado mantener la compatibilidad semántica tanto como ha sido posible, haciéndolos especialmente próximos desde el punto de vista del dominio. Esto no ha podido realizarse de forma estricta ya que el perfil RT-UML limita aspectos relevantes del modelo conceptual de MAST, y adaptarse a él hubiera significado bien perder capacidad de modelado para diversos tipos de sistemas de tiempo real de interés, o bien redefinir de manera local conceptos de muy alto nivel de abstracción, que son claramente merecedores de entidad propia en un perfil como RT-UML. El objetivo de este documento es justamente identificar estos aspectos que hemos encontrado restrictivos y exponer las modificaciones que se han propuesto al perfil para incrementar su generalidad.

El documento se organiza de la siguiente manera, en el siguiente apartado se describe de manera muy sucinta los objetivos y alcance del perfil RT-UML, se muestra el sub-perfil *SAprofile* de análisis de planificabilidad y se resume el proceso de estandarización de una especificación formal en el OMG, se describen muy brevemente los condicionamientos de UML-MAST y se citan los aspectos del perfil RT-UML que han sido analizados por encontrarse en el ámbito de esta herramienta. En el apartado 3 se enumeran y describen las limitaciones encontradas y las correspondientes propuestas. En el apartado 4 se comentan algunos otros aspectos del perfil de alcance más general y para los que a pesar de considerarlos mejorables, no tenemos planteadas propuestas concretas. Finalmente el apartado 5 presenta algunas conclusiones.

---

1. Este trabajo está financiado en parte por la Comisión Interministerial de Ciencia y Tecnología del gobierno español en el proyecto: "Diseño Integrado de Sistemas de Tiempo Real Embarcados", del Plan Nacional de Investigación (TIC99-1043-C03-03).
2. En adelante le identificaremos como RT-UML por su utilidad en la representación de los aspectos necesarios para el modelado con UML de sistemas de alta integridad y en particular de sistemas de tiempo real.

2. UN PERFIL DE UML PARA TIEMPO REAL

2.1. El perfil RT-UML del OMG

2.1.1. Generalidades

Tal como se enuncia en el propio perfil, el RFP solicitaba propuestas para un perfil UML que defina paradigmas de uso estándar para el modelado de aspectos relacionados con el tiempo, la planificabilidad y la respuesta de sistemas de tiempo real, tal que: permitieran la construcción de modelos que pudieran ser usados para hacer predicciones cuantitativas sobre estas características, facilitaran la comunicación de propuestas de diseño entre desarrolladores de forma estandarizada y facultaran la interoperatividad entre distintas herramientas de análisis y diseño.

Los principios que se enuncian como guía de la propuesta actualmente aprobada son los siguientes:

- En lo posible no restringir las formas en que el modelador quiera emplear UML para describir su sistema tan sólo porque éste deba después ser analizado.
- El modelo resultante debe ser útil para analizar y predecir las propiedades de tiempo real relevantes, y debe poder hacerlo desde las primeras fases del ciclo de desarrollo del software.
- El uso de las técnicas de análisis no debe requerir del modelador un conocimiento exhaustivo de sus aspectos internos.
- Se deben soportar todas las actuales tecnologías, paradigmas de diseño y técnicas de análisis y a la vez permitir la incorporación de otras nuevas.
- Se debe poder construir de manera automática diferentes modelos de análisis a partir de un mismo modelo UML y las herramientas que lo hagan deben ser capaces de leer el modelo, procesarlo y retornar los resultados al mismo en los términos del modelo original.

Así mismo se suponen tres potenciales tipos de destinatarios y siete casos de uso en los que el perfil puede ser empleado, que se ilustran en la Fig. 1 tomada del apartado 2.3 del perfil [15].

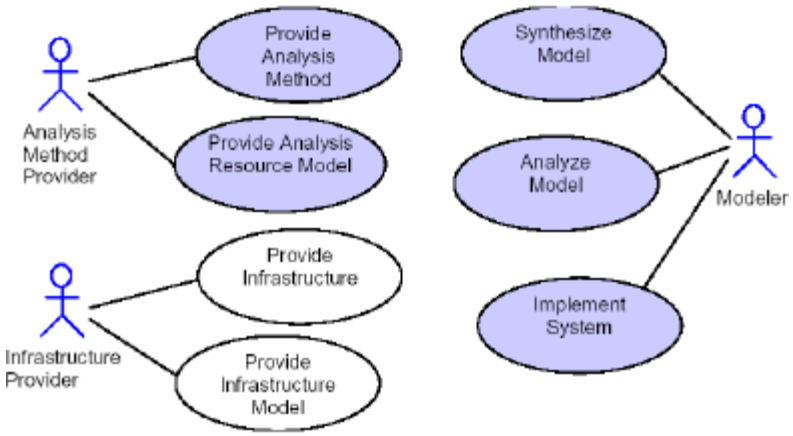

Fig. 1. Formas de uso del perfil RT-UML.

El cuerpo principal del perfil RT-UML es una estructura de sub-modelos que agrupan en dominios los conceptos que éste aporta, de manera que son utilizables o extensibles de forma independiente. Cada modelo de un dominio conceptual tiene asociado un sub-perfil que define el conjunto de estereotipos en los que se resumen los conceptos concretos que se proponen como de uso directo por el modelador, los "tagged values" propios de cada estereotipo con los que proporcionar valores a sus atributos y las restricciones que sean de aplicación. La idea central del perfil es pues tener un marco conceptual lo suficientemente rico y modular como para encajar con cierto grado de autonomía tanto otros métodos de análisis como nuevos dominios conceptuales que puedan aparecer mediante la especialización de los propuestos.

Otro aspecto de interés es que propone un marco de referencia para describir cómo se implementa dentro de una herramienta determinada el paradigma del procesado de modelos.

2.1.2. El sub-perfil SAprofile

De los sub-perfiles que incluye hemos utilizado en especial el de análisis de planificabilidad. En éste se proponen un conjunto de conceptos generales que pretende dar soporte a la información necesaria para aplicar métodos de análisis de planificabilidad. Se proponen también ciertas convenciones para la presentación de los elementos de modelado y se da una pauta sobre el tipo de diagramas en los que alojar el modelo.

El cuerpo principal del modelo del sub-perfil de planificabilidad, se muestra en la Fig. 2.

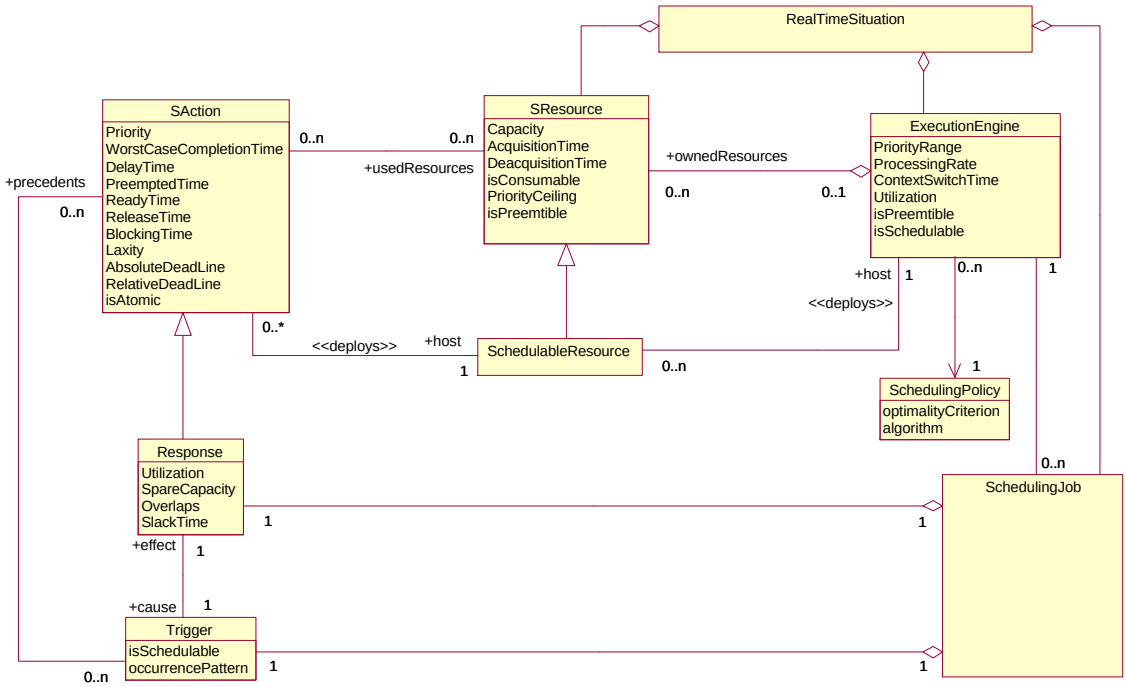

Fig. 2. Modelo fundamental del sub-perfil de análisis de planificabilidad.

Un aspecto destacable del apartado del documento en que se presenta este modelo, es la intención en él manifiesta de sus autores de plantearlo como base para una amplia variedad de métodos de análisis de planificabilidad, aún cuando por otra parte reconocen que bien sea por estar el mayor número de herramientas comerciales basadas en la teoría RMA [8], o por ser ésta además en la que tienen más experiencia, la especificación de este modelo puede estar influenciada por ella de manera sesgada.

2.1.3. El proceso de estandarización

Tan solo a manera de orientación sobre el camino que le queda al RT-UML para convertirse un una especificación formal, se enumeran de manera sucinta los términos que se emplean para identificar los documentos y los pasos que sigue una propuesta al OMG [20][21].

El primer paso suele ser el lanzamiento por parte del OMG de un RFP o *Request for Proposals*, mediante el cual se solicita propuestas concretas a los miembros del OMG, un paso previo opcional es el RFI o *Request for Information*, se trata de una prospección general que pretende recoger la opinión e información que sobre el tema se tenga en la industria.

Ante un RFP, las compañias deben responder en principio con un LOI o *Letter of Intent*, indicando su interes en presentar una propuesta, luego se abre un plazo para entregar las llamadas *Initial Submission*, se inicia entonces un proceso de revisión normalmente largo en el que se van consensuando las propuestas hasta quedar muy pocas a ser posible una *Revised Submission.*

De entre ellas después de un trabajo de negociación se vota y se aprueba así la reciente *Adopted Specification* y se establece un FTF *Finalization Task Force*, que es un grupo encargado de recibir y resolver enmiendas y propuestas, a la vez que se van realizando las implementaciones por parte de sus compañías. Este proceso puede alargarse más o menos en función de los *Issues* o asuntos pendientes que el FTF deba resolver y también de lo complejo que resulte implementar la

especificación, pues habitualmente las primeras versiones comerciales suelen aparecer a la par que el documento se convierte en una especificación formal. El RT-UML se encuentra actualmente en una de las últimas fases de este compás de espera entre implementación y aprobación formal, los últimos *Issues* aparecidos son de Febrero de 2003 sin embargo los anteriores son de Octubre de 2002.

2.2. UML-MAST

UML-MAST [11][3] es un entorno definido con UML que proporciona un conjunto de componentes de modelado para representar el comportamiento de tiempo real de sistemas distribuidos. Actualmente en el entorno MAST se han desarrollado los modelos relativos a sistemas mono-procesadores y distribuidos basados en prioridades fijas y con diferentes estrategias de planificación (expulsoras y no expulsoras, servidores esporádicos, polling, etc.) Las herramientas actualmente incorporadas permiten el análisis de sistemas de tiempo real basados en sistemas operativos y lenguajes comerciales de uso frecuente tales como POSIX y Ada.

UML_MAST persigue tres objetivos que coinciden exactamente con los casos de uso propuestos en el perfil RT-UML:

- Al diseñador ("*Modeler*") de sistemas de tiempo real, le proporciona los recursos que necesita para construir gradualmente el modelo de tiempo real del sistema que desarrolla. El modelo le sirve como base de aplicación de diferentes herramientas de análisis (análisis de planificabilidad, análisis de holguras, animación, etc.) y de otras herramientas de diseño (asignación óptima de prioridades, generación automática de código, etc.).
- Al proveedor de herramientas de desarrollo ("*Análisis and design method provider*") le proporciona un entorno estructurado de descripción de los sistemas de tiempo real, a partir del que se facilita la incorporación de nuevas herramientas de análisis o diseño, así como su comparación con otras ya validadas.
- Al proveedor de infraestructura (sistemas operativos, drivers, componentes, etc.), le proporciona una metodología consistente para el modelado del comportamiento de tiempo real de sus productos, y de especificación de sus capacidades de configuración.

Los modelos que se construyen en el entorno UML-MAST representan todas las características del sistema que se modela que son necesarias para poder predecir su comportamiento de tiempo real. La metodología que se utiliza modela independientemente los siguientes tres aspectos que condicionan el comportamiento de todo sistema de tiempo real:

- El modelo de la plataforma describe la capacidad de procesamiento y las restricciones operativas de los recursos de procesamiento hardware y software (procesadores, redes de comunicación, sistema operativo, drivers, etc.) que constituyen la plataforma sobre la que se ejecuta el sistema.
- El modelo de los componentes software describe la cantidad de procesado que requiere la ejecución de los módulos software (métodos y funciones definidos en las clases lógicas, primitivas de sincronización de threads, transferencia de mensajes por las redes, operaciones que realizan los dispositivos hardware, etc.) que se definen en el diseño lógico del sistema.
- El modelo de las situaciones de tiempo real que describen las configuraciones hardware/software que puede alcanzar el sistema, y para cada una de ellas la carga de trabajo que presenta, y los requerimientos de tiempo real que se le exigen.

Los modelos de tiempo real se formulan en UML y pueden ser gestionados por una herramienta CASE-UML, como una vista ("*Real-Time View*") complementaria a las utilizadas en el modelado y desarrollo del software. Las implementaciones que hemos desarrollado se han integrado en la herramienta ROSE'2000e de Rational.

2.3. Dominios del perfil RT-UML estudiados.

El entorno UML-MAST constituye una metodología y una herramienta de análisis representativa de las que trata de cubrir el perfil RT-UML. Específicamente es una implementación concreta destinada al modelado de sistemas de tiempo real para análisis de planificabilidad, implementa por tanto el sub-perfil "*SAprofile*" y emplea de los modelos contenidos en los demás sub-perfiles algunas secciones y conceptos que se enumeran a continuación con el propósito de enfatizar aquellos aspectos del perfil RT-UML sobre los cuales se han hecho la evaluacón y propuestas:

- *General Resource Model* (GRM): Proporciona el dominio básico de modelado de los componentes mas generales.
  - *Core ResourceModel*: Se utilizan de forma generalizada los conceptos de "*Instance-Descriptor*", "*Resource*" y "*QoScharacteristic*". No se utiliza el concepto "*ResourceService*" pues encontramos poco delimitada su diferencia con *Resource*.
  - *CausalityModel*: Se utilizan los conceptos "*Scenario*" y "*Stimulus*".

        - Solo se utiliza *Stimulus* como causa externa de disparo de los *Scenarios*.
        - El modelado basado en transacciones concurrentes que se utiliza hace que no se use el concepto de *Stimulus* como relación de causalidad entre *Scenarios*, .
        - Se utiliza el concepto de *State* y el de *EventOccurence* que se genera como elemento que describe la evolución del *scenario*. Al *State* se le asocia rol de efecto, pero no se le asocia rol de causa puesto que evolución de los estados del sistema es en realidad el mero transcurso del tiempo.
    - *Resource Usage Model*: Solo se le utiliza en su modelo dinámico. Se utilizan los conceptos "*AnalysisContext*", "*ResourceUsage*" ("*DynamicUsage*"), "*UsageDemand*" y "*ActionExecution*".
        - El modelo resulta insuficiente para diferenciar entre el *Descriptor* de un *ResourceUsage*, que se requiere para modelar un componente software (método, primitiva de sincronización, operación de un dispositivo, etc.) y la *Instance* de un *ResourceUsage* que se requiere para modelar la invocación de un componente software dentro de un *scenario* concreto.
        - La descripción de un *Scenario* como una secuencia ordenada de *ActionExecution* no es suficiente si se trata de describir un *Descriptor* de un *Scenario*.
    - *Resource Type Model*: Se utilizan todos los tipos de recursos propuestos en el modelo.
        - No se utilizan el concepto de *ExclusiveService*, ya que no se considera bien delimitado el concepto de *ResourceService*, pero su semántica y operatividad se transfieren al *ProtectedResource*.
    - *Resource Management Model*: No se utilizan los conceptos del modelo.
- *General Time Model*: Proporciona los conceptos básicos sobre la medida del tiempo y su resolución.
    - *Basic Time Model*: Se utilizan los conceptos de *TimeValue* y *TimeInterval* utilizando una medida de tiempo densa (derivada de *Float*).
    - *Timing Mechanism Model*: Se utilizan los conceptos *TimingMechanism*, *Clock* y *Timer*.
        - Se limitan a un solo *TimingMechanism* por *ActiveResource*.
    - *Timed Events Model*: Se utilizan los conceptos de *TimedStimulus* ,*TimeOut*, *ClockInterrupt*, *TimedEvent* y *TimedAction*.
        - No se dispone de capacidad de definir patrones probabilísticos de generación de *TimedStimulus*.
        - Solo se consideran *TimedAction* de duración determinista definida por un único *TimeInterval*.
- *Concurrency Domain Model*: No se hace uso del modelo de manera explícita. Se considera que la concurrencia es implícita entre los *ActiveResource* y está limitada por su comportamiento de *ProtectedResource*, respecto de las *ActionExecution* asignadas.
- *Schedulability Model*: Proporciona los conceptos para modelar sistemas de tiempo real a efectos de análisis de planificabilidad. Se utilizan todos los conceptos que allí se presentan con la misma semántica, considerando las limitaciones que se exponen en el apartado 3.

3. LIMITACIONES Y PROPUESTAS

La mayor parte de las limitaciones encontradas, devienen de la concepción de las técnicas de análisis tal y como se presentan en sus versiones originales. Las técnicas de análisis de planificabilidad han evolucionado significativamente a lo largo de la última década, y muy en particular las que se apoyan en mecanismos de planificación basados en prioridades fijas, al ser éste el mecanismo que más frecuentemente se ve incorporado en los sistemas operativos y lenguajes estándares y de mayor difusión comercial. Si bien es cierto que las primeras técnicas de análisis de planificabilidad basadas en prioridades fijas, que lograron una mayor difusión y amplia aceptación en la comunidad de desarrolladores de software de tiempo real, se avocaban a la evaluación de sistemas mono-procesadores [10][8][9][7][5], actualmente se dispone de una creciente gama de ellas que facilitan la evaluacón de sistemas de tiempo real de naturaleza distribuida [12][13][18][6]. De manera similar se dispone de la suficiente infrestructura de comunicaciones[16][17] para considerar útiles tales técnicas.

Sobre esta premisa y observando minuciosamente el modelo presentado en el SAprofile, se plantean las limitaciones que se describen a continuación.

3.1. Definición de *ExecutionEngine*

3.1.1. Limitación:

Se requiere que la *ExecutionEngine* sea un *Processor*, excluyendo que sea un *CommunicationResource* o un *Device*.

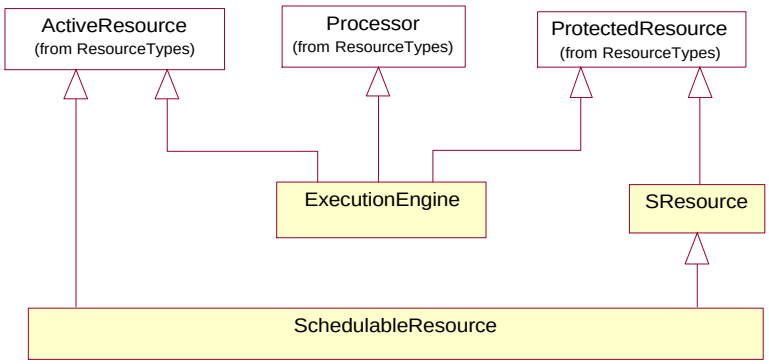

Fig. 3. Modelo de ExecutionEngine.

3.1.2. Análisis de la limitación:

El perfil está claramente limitado al análisis de planificabilidad dentro de un entorno monoprocesador. Y presupone que los únicos recursos con capacidad de ejecutar un *SchedulingJob* sean *processors*. Se debe extender el concepto de *ExecutionEngine* a todo recurso capaz de ejecutar cualquier actividad del *SchedulingJob* a ser planificado. Por ejemplo, si el *SchedulingJob* ha de ser ejecutado sobre un sistema distribuido, los procesos de transferencia de mensajes entre nudos que participan en su ejecución, son actividades que deben ser planificadas y que son llevadas a cabo (ejecutadas) por un *CommunicationResource*. Debe extenderse el concepto de *ExecutionEngine* para que pueda ser cualquier otro tipo de *ResourceInstance* que sea a su vez *ProtectedResource* y *ActiveResource* (*Processor*, *Device* o *CommunicationResource*).

3.1.3. Propuesta:

Una alternativa aunque simple poco específica es eliminar la herencia de *Processor*, con lo que *ExecutionEngine* sería un *ResourceInstance* general, activo y protegido. Otra alternativa, que es la que se ha propuesto [5706], es incluir una nueva superclase en la taxonomía de recursos que generalice los tipos de recursos que son capaces de ejecutar acciones, en la Fig. 4 se muestra la clase *ExecutingResource* y la redefinición de *ExecutionEngine* en base a ella.

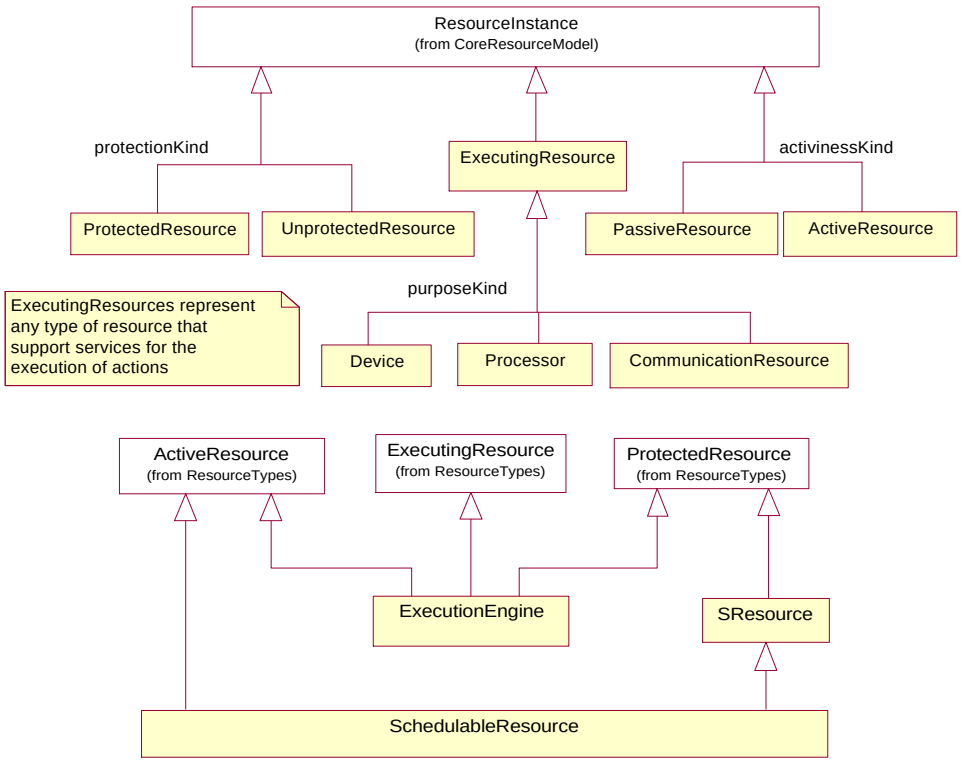

Fig. 4. Modelo que emplía ExecutionEngine para el análisis de sistemas distribuidos.

La definición propuesta de *ExecutionEngine* es: "An execution engine is an active, protected, executing-type resource that is allocated to the execution of schedulable resources, and hence any actions that use those schedulable resources to execute. In general, they are processor, network or device."

3.2. Relación entre *SchedulingJob* y *ExecutionEngine*

3.2.1. Limitación:

Cada *SchedulingJob* está unívocamente asignado a una *ExecutionEngine*.

3.2.2. Análisis de la limitación:

De nuevo esta limitación impide modelar aplicaciones distribuidas en las que deben planificarse *SchedulingJobs* que se ejecutan haciendo uso de diferentes nudos del sistema distribuido.

3.2.3. Propuesta:

Se propone hacer explícitamente múltiple [5713][5721] la asociación entre *SchedulingJob* y *ExecutionEngine*. Con lo cual además se debe redefinir el rol *ExecutionEngine* pasándolo al plural [5717].

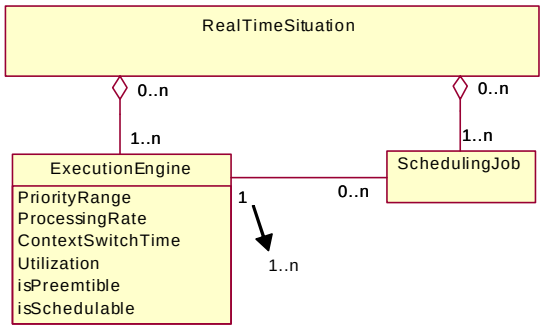

Fig. 5. Relación entre SchedulingJob y ExecutionEngine.

Otras opciones serían eliminar la asociación o hacerla opcional [1]. Si se eliminase, ésta se haría efectiva de manera indirecta entre un *SchedulingJob* y los múltiples *ExecutionEngine* a través de la asignación de las *SAction* a los correspondientes *SchedulableResource*, si se hace opcional, habría que aclarar mediante un comentario cual es el rol que representa le *ExecutionEngine* asociada, quizá podría contemplarse como aquella que inicia o recibe el evento que inicia el *SchedulingJob*.

3.3. Relación entre *SchedulingJob* y *Trigger*

3.3.1. Limitación:

La asociación entre un *SchedulingJob* y su *Trigger* es unívoca en ambos sentidos.

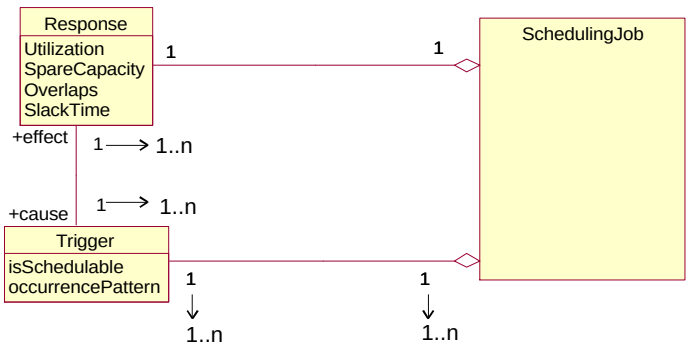

Fig. 6. Relación entre SchedulingJob y Trigger

3.3.2. Análisis de la limitación:

Ello significa que un *SchedulingJob* es excitado por un único *Trigger* y cada *Trigger* puede pertenecer a un único *SchedulingJob*. Existen casos sin embargo los que diferentes *Trigger* pueden intervenir en la activación de un *SchedulingJob* y viceversa:

- Un mismo *Trigger* pueden demandar diferentes *SchedulingJob*.
- Un *SchedulingJob* puede ser demandado alternativamente por diferentes *Triggers*.
- Un patrón complejo resultante de combinar diferentes eventos que se suceden a lo largo de la ejecución del *SchedulingJob* puede ser necesario para demandarlo en su totalidad.

3.3.3. Propuesta:

La asociación entre los *SchedulingJob* y sus *Trigger* debe ser múltiple en ambos sentidos [5713].

3.4. Encadenamiento de *SAction*

3.4.1. Limitacion:

Las *SAction* heredan de *TimedAction* y transitivamente de *Scenario*, sin embargo no heredan las propiedades de *ActionExecution*, en particular resultan así no encadenables ni demandantes de valores concretos de *QoSvalue* de sus recursos.

3.4.2. Análisis de la limitación:

Según el modelo de uso dinámico de recursos que se observa en la *Fig. 7* (derecha) un Scenario puede contener una secuencia ordenada de ActionExecutions, y estas además de encadenarse mediante los roles de sucesor y predecesor pueden a su vez contener a otras por ser especializaciones de Scenario.

El modelo deseable para una acción planificable es que ésta se pueda encadenar lo mismo que contener recursivamente a otras del mismo tipo y que pudieran además estar asignadas a distintos SchedulableResource y/o tener sus propios parámetros de planificabilidad.

3.4.3. Propuesta:

Hacer *TimedAction* especialización de *ActionExecution* [5720], en lugar de hacerlo directamente de *Scenario*, con lo que se admite en *SAction* ambos comportamientos.

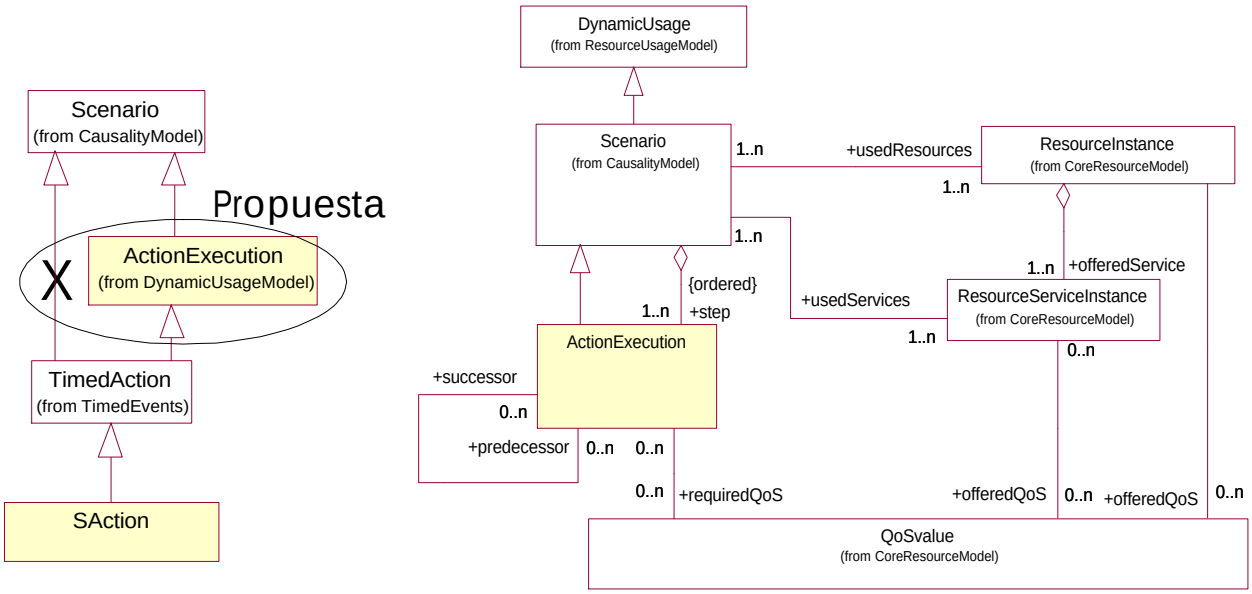

Fig. 7. Falta de encadenamiento de SAction

3.5. Relación entre SAction y SchedulableResource

3.5.1. Limitación:

Cada *SAction* está univocamente asociada a un *SchedulableResource*.

3.5.2. Análisis de la limitación:

Ello limita la recursividad en la definición de las *SAction*. Una *SAction* sólo puede estar definida en función de otras *SAction* que sean planificadas por el mismo *SchedulableResource*.

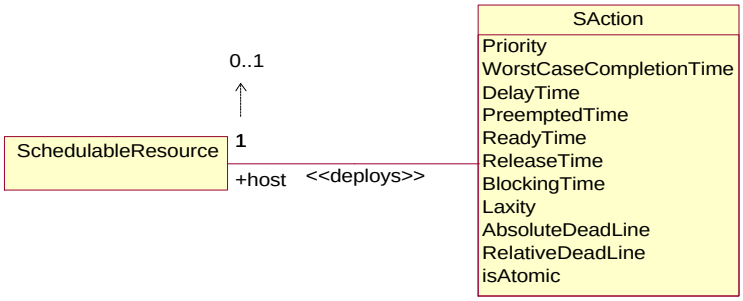

Fig. 8. Relación entre *SchedulableResource* y *SAction*

3.5.3. Propuesta:

Establecer como optativa la asociación entre la *SAction* y el *SchedulableResource*. Para la *SAction* que en su totalidad se planifica en un *SchedulableResource*, se establece el enlace, mientras que para aquellas que se planifican en varios *SchedulableResource* no se establece asociación y se deja la asignación a las *SAction* internas que correspondan [5721]. Además se sugiere añadir el siguiente parrafo a la definición del atributo correspondiente al rol *host*:

"the schedulable resource that the action executes on; this is only defined if all the internal SActions that constitute the SAction execute on the same schedulable resource" [5715].

3.6. Duración de una *SAction*

3.6.1. Limitación:

A través de herencia desde *TimedAction*, las *SAction* tienen un único atributo de temporización, ya sea expresado mediante *duration* o mediante *start* y *end*, el caso es que resulta insuficiente para el análisis de planificabilidad, pues las acciones tienen en general una duración variable.

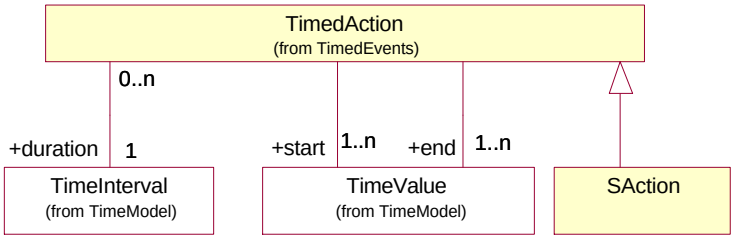

Fig. 9. Atributo de temporización de *SAction*

3.6.2. Análisis de la limitación:

Existen técnicas de análisis de planificabilidad que hacen uso de patrones mas complejos de caracterización de las *TimedAction*. Por ejemplo, en las técnicas de análisis de planificabilidad basadas en *Offsets* [13][12], se hace uso tanto de la máxima duración admisible de las *TimedAction* como de la mínima duración posible de las mismas.

3.6.3. Propuesta:

El atributo *duration* de *TimedAction* se puede convertir en una lista [5711] de carácter más general, a partir de la cual derivar los que fueran necesarios, las listas *start* y *end* pueden mantener su semántica o adaptarla de manera consistente.

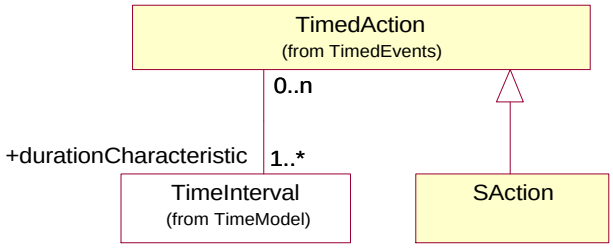

Fig. 10. Propuesta de asociación que generaliza los tipos de parámetros de temporización útiles en *SAction*

3.7. Modelado de secuencias optativas

Basándose en la presunción plausible de que se admita la propuesta del apartado 3.4.3. se encuentra la siguiente limitación en las variantes de conectividad que se otorgan a los roles *predecessor* y *successor* de *ActionExecution*.

3.7.1. Limitación:

Las posibilidades de establecer relaciones de causa-efecto entre *SAction*, que vienen heredadas de *ActionExecution*, sólo permiten representaciones relaciones de los tipos *joint* (AND de entrada) y *fork* (AND de salida), y no admiten relaciones *merge* (OR de entrada) y decisiones o *branch* (OR de salida).

3.7.2. Análisis de la limitación:

Esta limitación resulta relativamente consistente en el contexto general del perfil, puesto que se encuentra orientado muy particularmente a la representación de modelos de instancias, de modo tal que el modelo de análisis de una sección de código o de un cierto algoritmo, debería representar un caso concreto de ejecución. Sin embargo esta es una limitación innecesaria, pues aún a riesgo de introducir pesimismo en el modelo de análisis, puede resultar conveniente tener modeladas las secuencias alternativas de flujo de control que un escenario puede desplegar, para hacer diversos tipos de análisis a partir de una definición general, existen herramientas de análisis de planificabilidad que permiten la utilización de este tipo de modelos [6], y pueden incluso ser útiles a fin de aplicar herramientas de cálculo automático de tiempo de ejecución o de respuesta de peor caso.

3.7.3. Propuesta:

Redefinir el significado de los roles *successor* y *predecessor*, ampliándoles de modo que admitan además las posibilidades de *merge* (OR de entrada) y decisiones o *branch* (OR de salida).

4. OBJECIONES AL ESTILO SUBYACENTE EN EL PERFIL.

Se proponen objecciones relativas a los principios subyacentes en el modelo que restringe la capacidad de modelado de sistemas de tiempo real o dificulta la gestión de los modelos.

Estas objeciones son relativas a la propia concepción del perfil y afecta a la forma en que los autores han presentado el perfil. En consecuencia es difícil proponer modificaciones concretas.

4.1. Modelado de los componentes software como base del modelo de su ejecución.

En la formulación de perfil se considera que un modelo de tiempo real es básicamente un modelo de instancias.

"Because real-time analyses are instance based, in the rest of this model we will not define explicitly any additional descriptor concepts." (página. 24)

Aunque esto es cierto en el modelo de tiempo real final de una situación de tiempo real de un sistema, para elaboración de modelos de sistemas complejo es muy importante disponer de un descriptor genérico del *ResourceUsage* que representa

el modelo de tiempo real de un componente software, que pueda ser la base de la generación de las instancias concretas que modelan cada una de las ejecuciones del módulo software.

Esto puede hacerse definiendo descriptores de *ResourceUsage* que tienen parametrizadas ciertas asociaciones que conlleva y que representan su contexto. Cada instancia de *ResourceUsage* que se necesita en el modelo para representar una invocación del módulo, se obtiene asignando a los parámetros instancias concretas de otros elementos del modelo.

Este aspecto es necesario para el usuario que lo utilice como proveedor de infraestructura, que necesita proporcionar modelos del comportamiento de tiempo real de los componentes software o del software de base (sistema operativos, drivers, etc.) que suministra.

4.2. Modelado del software de sistema.

La capacidad de procesado de un recurso es función de la capacidad de su hardware, pero también del consumo de capacidad de procesamiento (cambios de contexto, gestión de timers, atención de drivers, etc) que lleva a cabo el software de base que se ejecuta en background y sobre el que se ejecuta la aplicación cuya planificabilidad se modela.

Aunque el software de sistema es modelable como lo es el de la aplicación, existe la diferencia fundamental de que sus detalles no son conocidos por el modelador, por lo que es conveniente considerarlo no como parte de la aplicación (*ResourceUsage*) sino como parte del modelo del recurso (*Resource*)

4.3. Representación del modelo de tiempo real como una nueva vista del modelo UML del sistema.

- En el perfil se propone la formalización del modelo de tiempo real en UML a través de la utilización de Estereotipos y Tags asociados a los entes de su descripción lógica, de componentes o de despliegue. Esta fórmula presenta varias dificultades:
- El modelo de tiempo real resulta muy disperso, sobre todo en aplicaciones grandes en las que sólo algunos componentes son relevantes a efecto de satisfacer los requerimientos de tiempo real. La dispersión es una complicación para el diseño de las herramientas que deben operar sobre el modelo y complica su expresividad de cara al modelador de tiempo real.
- La utilización de tags no está normalizada en UML. Las herramientas que traten el modelo de tiempo real deben obtener los datos a través del procesado del texto de las Tags y gestionar bases de datos externas a la herramienta para manejar el modelo de tiempo real.

Por ello consideramos mas conveniente un modelo diferente de representación:

- El modelo de tiempo real representa una caracterización complementaria (respecto de la lógica, la de componentes o la de despliegue) del sistema que se modela. Si se concentra en una nueva vista (*Real-Time view*) los componentes del modelo de tiempo-real, su mantenimiento y su gestión por las herramientas que lo procesan resultan mas simples.
- Las características de los componentes del modelo (*QoScharacteristics*) son atributos de los mismos, por lo que una estrategia de representación del modelo basada en clases para representar los descriptores y objetos para representar sus instancias, convenientemente esterotipadas para indicar el aspecto que la clase u objeto representan y en los atributos como forma de representar las *QoScharacteristics*, tiene las ventajas de que emplea recursos notacionales muy comunes en las herramientas UML, y como consecuencia de ello, su gestión es soportada por las propias herramientas CASE-UML.

5. CONCLUSIONES

El *"UML Profile for Schedulability, Performance and Time"* se encuentra a punto de convertirse en una especificación formal del OMG. Está destinado a soportar el grueso de los conceptos necesarios para establecer los modelos de análisis de aplicaciones e infraestructuras de software cuya principal característica es jústamente la necesidad de ser analizados, a fin de garantizar sus requisitos tanto funcionales como relativos a su respuesta temporal. Se han detectado defectos en el modelo que describe estos conceptos y se ha verificado que estos resultan particularmente limitantes cuando estas aplicaciones hacen uso de recursos de comunicación y tienen requisitos temporales que responden a modelos de análisis de respuesta *end-to-end*. Estos y otros aspectos mejorables, se han documentado y comunicado a quienes tienen la responsabilidad de estudiarlos al interior del OMG. Una vez estén superados, se prevé que el perfil se convertirá en una referencia de carácter general para la adopción de modelos de análisis basados en tecnologías UML, que previsiblemente facilitarán la interoperabilidad entre herramientas y que ayudará a la difusión a la vez que simplificará la aplicación de técnicas de análisis, que por su aridez o complejidad quizá han sido a menudo poco explotadas por la industria. Finalmente

cabe mencionar que si bien reconocemos éstas como ventajas, no quisiéramos en ningún caso considerar el estándar al punto de limitar la creatividad de nuestra comunidad, sea por tanto esta reflexión final para observar que los estándares son tanto un punto de encuentro de nuestros esfuerzos como un reto para superarlos.

BIBLIOGRAFÍA

[1] Miguel de Miguel and Julio Medina: "Evaluation of FTF: Scheduling, Performance, and Time". Internal Report submitted to the OMG Finalization Task Force for "UML Profile for Schedulability, Performance and Time". April 2002.

[2] *Issues* pendientes enviados a la *Finalization Task Force* del OMG correspondiente al perfil de "*Schedulability Performance and Time*", `http://www.omg.org/issues/uml-scheduling-ftf.html`.

[3] J.M. Drake and J.L. Medina: "UML-MAST Metamodel, specification, example of application and source code". `http://mast.unican.es/umlmast`

[4] M. González Harbour, J.J. Gutiérrez, J.C.Palencia and J.M. Drake: "MAST: Modeling and Analysis Suite for Real-Time Applications". Proceedings of 13th Euromicro Conference on Real-Time Systems, Delft, The Netherlands, IEEE Computer Society Press, pp. 125-134, June 2001

[5] M. González Harbour, M.H. Klein, and J.P. Lehoczky. "Fixed Priority Scheduling of Periodic Tasks with Varying Execution Priority". Proceedings of the IEEE Real-Time Systems Symposium, December 1991, pp. 116-128

[6] J.J. Gutiérrez García, J.C. Palencia Gutiérrez, and M. González Harbour, "Schedulability Analysis of Distributed Hard Real-Time Systems with Multiple-Event Synchronization". Euromicro Conference on Real-Time Systems, Stockholm, Sweden, 2000.

[7] M. Joseph and P. Pandya, "Finding Response Times in a Real-Time System," *BCS Computer Journal*, Vol. 29, no. 5, October 1986, pp 390-395.

[8] M. Klein, T. Ralya, B. Pollak, R. Obenza, and M. González Harbour, "A Practitioner's Handbook for Real-Time Systems Analysis". Kluwer Academic Pub., 1993.

[9] J.P. Lehoczky. "Fixed Priority Scheduling of Periodic Tasks Sets with Arbitrary Deadlines". *IEEE Real-Time Systems Symposium*, 1990.

[10] C.L. Liu, and J.W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment". Journal of the ACM, 20 (1), pp 46-61, 1973.

[11] J.L. Medina, M. González Harbour, and J.M. Drake: "MAST Real-Time View: A Graphic UML Tool for Modeling Object-Oriented Real-Time Systems" RTSS'01, London, December, 2001.

[12] J.C. Palencia, and M. González Harbour, "Schedulability Analysis for Tasks with Static and Dynamic Offsets". Proc. of the 19th IEEE Real-Time Systems Symposium, 1998.

[13] J.C.Palencia, and M. González Harbour, "Exploiting Precedence Relations in the Schedulability Analysis of Distributed Real-Time Systems". Proceedings of the 20th IEEE Real-Time Systems Symposium, 1999.

[14] B. Selic, "A Generic Framework for Modeling Resources with UML". IEEE Computer, Vol. 33, N. 6, pp. 64-69. June, 2000.

[15] Bran Selic, Alan Moore, Murray Woodside, Ben Watson, Morgan Bjorkander, Mark Gerhardt and Dorina Petriu: "UML Profile for Schedulability, Performance and Time". OMG document ptc/2002-03-02, March 2002. `http://www.omg.org/cgi-bin/doc?ptc/2002-03-02`

[16] J.K. Strosnider, T. Marchok, J.P. Lehoczky, "Advanced Real-Time Scheduling Using the IEEE 802.5 Token Ring". Proceedings of the IEEE Real-Time Systems Symposium, Huntsville, Alabama, USA, pp. 42-52, 1988.

[17] K. Tindell, A. Burns, and A.J. Wellings, "Calculating Controller Area Network (CAN) Message Response Times". Proceedings of the 1994 IFAC Workshop on Distributed Computer Control Systems (DCCS), Toledo, Spain, 1994.

[18] K. Tindell, and J. Clark, "Holistic Schedulability Analysis for Distributed Hard Real-Time Systems". Microprocessing & Microprogramming, Vol. 50, Nos.2-3, pp. 117-134, 1994.

[19] Grupo de Computadores y Tiempo Real de la Universidad de Cantabria. Página de documentación y descarga de MAST. `http://mast.unican.es/`

[20] Open Management Group. OMG's Technology Adoption Process. `http://www.omg.org/technology/documents/spec_tutorial.htm` y `http://www.omg.org/gettingstarted/processintro.htm`

[21] Fred Waskiewicz, "The Hitchhiker's Guide to the OMG Technology Adoption Process", `http://www.omg.org/cgi-bin/doc?omg/01-08-01.pdf`

[XXXX] *Issues* pendientes de resolución enviados al *OMG Finalization Task Force* del "*UML Profile for Schedulability Performance and Time*", `http://www.omg.org/issues/uml-scheduling-ftf.html#IssueXXXX`