XV Jornadas de Ingeniería del Software y Bases de Datos
JISBD 2006
José Riquelme - Pere Botella (Eds)

EDROOM, HERRAMIENTA LIBRE DE MODELADO Y GENERACIÓN AUTOMÁTICA DE CÓDIGO PARA SISTEMAS DE TIEMPO REAL

P. Parra, A. Viana, O. Rodríguez, M. Knoblauch, F. Alcojor, S. Sánchez, J. Ignacio García, O. García y D. Meziat

Departamento de Automática
Escuela Politécnica Superior
Universidad de Alcalá
Carretera Madrid-Barcelona Km. 33,600
e-mail: {pablo,aitor,opolo,martin,falcojor,ssp,nach,ogarcia,meziat}@srg.aut.uah.es, web: http://srg.aut.uah.es

Palabras clave: Modelado, Sistemas de tiempo real, Generación automática de código

Resumen. *El desarrollo de sistemas de tiempo real es una tarea de considerable complejidad. Desde el área de la ingeniería del software se han propuesto, en las últimas décadas, diversos lenguajes de modelado que pretenden facilitar el proceso de desarrollo. Los modelos de los sistemas así obtenidos se basan en formalismos, estructuras o diagramas que proporcionan un nivel de abstracción adecuado al análisis y diseño, reduciendo el uso de los lenguajes clásicos de programación a la fase final de implementación. Para facilitar la definición del modelo, y especialmente, para asegurar la coherencia entre éste y su realización, se hace necesario el uso de herramientas de desarrollo asistido por ordenador (del inglés* Computer Aided Software Engineering *o CASE). Estas herramientas permiten construir el modelo del sistema empleando la sintaxis propia del lenguaje, generalmente de naturaleza gráfica, y generar automáticamente el esqueleto de las aplicaciones, lo que simplifica considerablemente la tarea del programador y facilita la posterior verificación. En este artículo se presenta la herramienta CASE de libre distribución EDROOM, basada en el lenguaje de modelado ROOM (*Real Time Object Oriented Modelling*). Los modelos construidos con EDROOM permiten describir la estructura, la topología de comunicación y el comportamiento de los sistemas de tiempo real empleando diagramas. La herramienta, además, genera código de manera automática para múltiples y variadas plataformas y ha sido empleada, entre otros proyectos, en el desarrollo del software de vuelo del satélite NANOSAT del Instituto Nacional de Técnica Aeroespacial (INTA).*

1 INTRODUCCIÓN

El desarrollo de software ha evolucionado considerablemente desde los primeros pasos de las ciencias de la computación. Esta evolución ha posibilitado el abandono cada vez más

generalizado de las técnicas de desarrollo no estructuradas, en favor de las metodologías de diseño organizado y del uso de herramientas de generación automática de código. En la actualidad existen numerosos formalismos para definir el comportamiento y la estructura de los distintos sistemas, incluidos los de tiempo real. Uno de estos formalismos es ROOM [9], cuya sintaxis gráfica para la definición de componentes ha sido incorporada en el estándar de modelado UML (*Universal Modelling Language*) Versión 2. Los modelos ROOM permiten al diseñador definir gráficamente la estructura de componentes de su sistema, el comportamiento de cada componente y la topología de comunicación entre ellos. En ROOM, los componentes se comunican exclusivamente a través de interfaces de comunicación que se denominan *puertos*, lo que proporciona una ocultación completa de la implementación de cada componente. El artículo presenta la herramienta de modelado gráfico de libre distribución EDROOM, inspirada en la metodología ROOM, desarrollada con el fin de facilitar el diseño e implementación de sistemas de tiempo real [5, 8]. A diferencia de otros entornos similares como Cadena [3], que genera cdigo Java para sistemas que usen el Modelo de Componentes CORBA (*CORBA Component Model* o CCM), EDROOM genera automáticamente, a partir del modelo, el código de la aplicación en C++ Empotrado (*Embedded* C++), un subconjunto del C++ estándar que pretende minimizar la utilización de recursos. Dicho código se soporta sobre una biblioteca de funciones denominada Biblioteca de Servicios EDROOM que está estructurada en dos capas o niveles. La capa superior, que proporciona la interfaz al código generado por EDROOM y la inferior que encapsula los detalles del sistema operativo. Esta última capa es de reducido tamaño y complejidad, factor que favorece su portabilidad a distintos sistemas operativos de tiempo real. Entre otros, la herramienta ha sido adaptada para trabajar sobre W32, RTKernel, VxWorks, CMX, ERCOS [8] y RTAI [4]. En las siguientes secciones describiremos más a fondo y con más detalle la herramienta y daremos cuenta de algunos proyectos en los que ha sido utilizada.

2 LA HERRAMIENTA EDROOM

EDROOM es una herramienta de desarrollo inspirada en el lenguaje de modelado orientado a objetos ROOM. Incluye un editor gráfico para definir la estructura de los distintos componentes, la topología de comunicación entre ellos, y su comportamiento. Además proporciona facilidades para gestionar los recursos e integra un generador automático de código C++, que traduce la representación gráfica del modelo a la invocación de servicios implementados mediante una biblioteca.

2.1 Estructura y Topología de Comunicación.

La estructura de los componentes de un modelo EDROOM se basa en la definición de clases de componentes y en las relaciones de agregación establecidas entre dichas clases. Una clase específica ocupa el nivel *Top* dentro del modelo, de manera que, a partir de la determinación de sus agregados inmediatos y sus correspondientes clases, se establece

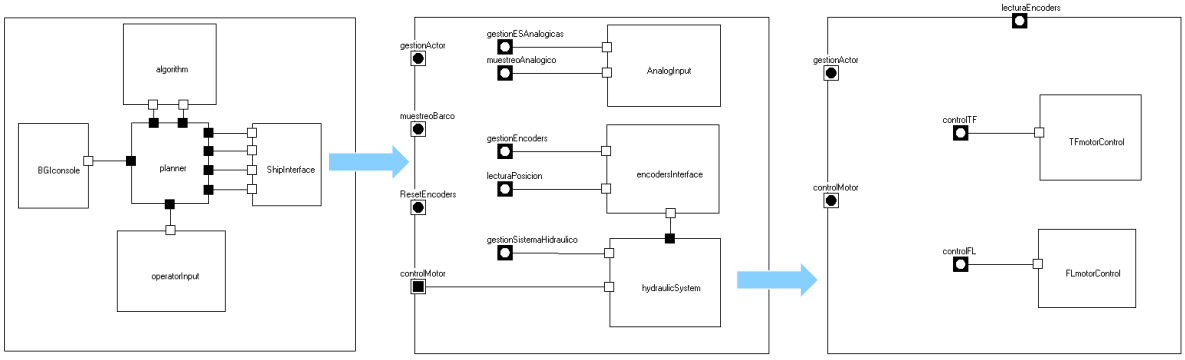

Figura 1: Estructura multinivel de un modelo EDROOM

la jerarquía completa de componentes. Esta jerarquía no tiene limitación en el número de niveles y proporciona una forma de encapsular cómo se distribuye la funcionalidad del sistema de un nivel en distintos componentes de los niveles inferiores.

La comunicación entre los componentes se realiza exclusivamente mediante intercambio de mensajes. Cada mensaje lleva una señal que lo identifica y un dato opcional. Los componentes envían y reciben los mensajes a través de unos puntos de conexión denominados *puertos*, que aislan completamente ambos lados de la comunicación. Los puertos de un componente quedan definidos en su clase, de forma que todas las instancias de una misma clase presentan el mismo interfaz de comunicación. Cada puerto, a su vez, es una instancia de una clase protocolo en la que se establecen los mensajes de entrada y salida que será posible recibir y transmitir a través de los puertos. Existe una forma especial de instanciación de los puertos a partir de una clase protocolo que se denomina instanciación conjugada. Este tipo de instanciación toma los mensajes de salida del protocolo como de entrada y viceversa.

Una vez definidos los puertos de todos los componentes, la topología de comunicación queda determinada mediante la conexión punto a punto entre ellos. Generalmente se establecen conexiones entre un puerto normal y otro conjugado de la misma clase protocolo.

La Figura 1 muestra un ejemplo de la estructura de componentes y la topología de comunicación de un modelo multinivel definido con EDROOM. Cada uno de los componentes aparece representado como un elemento rectangular. Los puertos se sitúan en los contornos de los componentes. El puerto que utiliza un protocolo de manera estándar recibe la forma de un pequeño cuadrado de color blanco. Por el contrario, el puerto que emplea el protocolo conjugado se representa mediante el color negro.

2.2 Comportamiento

El comportamiento de cada componente está definido dentro de su clase empleando un tipo de diagramas de estados denominados *ROOMCharts*. Estos diagramas, basados

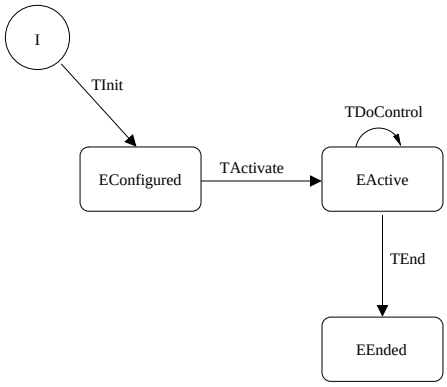

Figura 2: Ejemplo de diagramas de estados EDROOM

en los *StateCharts* de Harel [2], permiten la definición jerárquica del comportamiento, posibilitando la definición de nuevas máquinas de estados dentro de cada subestado. Esta facilidad permite encapsular los detalles del comportamiento reactivo y del control de la secuencia, que en los sistemas de tiempo real generalmente resultan especialmente complejos. Las transiciones entre los estados se disparan exclusivamente mediante la recepción de un mensaje, pudiéndose asociar, tanto a la transición, como a la entrada y salida de los subestados, la ejecución de acciones. Es en la definición de las acciones donde se integra la implementación a nivel de detalle en el comportamiento, pudiéndose invocar desde ellas servicios tales como el envío de un mensaje a través de un puerto, o la programación de un *timer* o la llamada a un *driver* de un dispositivo. En la Figura 2 se puede observar un ejemplo de diagramas de estados que definen el comportamiento de un componente.

2.3 Gestión de los recursos

Los sistemas de tiempo real no deben hacer uso de las clásicas funciones de gestión de la memoria dinámica *malloc* y *free* por no ser deterministas. Por este motivo EDROOM integra la gestión de *pools* de memoria que posibilitan el manejo de los mensajes y sus datos asociados. Los *pools* permiten que la memoria asociada a los mensajes y datos sea liberada de manera transparente una vez que éstos han sido tratados, emulando así la gestión de memoria dinámica que el modelo de envío de mensajes requiere. EDROOM, permite además, controlar recursos tales como las fuentes de excepciones e interrupciones, incluidas en estas últimas aquellas que se producen periódicamente y que constituyen el reloj monotónico del sistema. Para ello proporciona servicios de excepción, temporización e interrupción integrados en la implementación detallada del comportamiento.

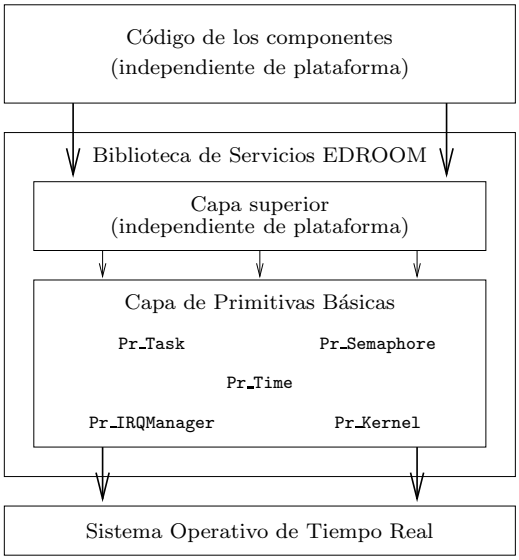

Figura 3: Estructura global de las aplicaciones generadas por EDROOM

2.4 Generación de código

El código generado por la aplicación se soporta sobre una serie de funciones que constituyen la denominada Biblioteca de Servicios EDROOM. Esta biblioteca está definida en dos capas o niveles: una capa superior independiente de la plataforma que proporciona la interfaz de servicios al código de las aplicaciones generado por EDROOM y una capa inferior que está en contacto directo con el sistema operativo de tiempo real. El esquema de la biblioteca es el que muestra en la Figura 3. La capa inferior recibe el nombre de capa de primitivas básicas. A continuación se describe la funcionalidad que proporcionan ambos niveles de la biblioteca.

2.5 Capa de primitivas básicas

Las primitivas de la capa inferior están definidas en forma de clases C++. Las más importantes son: `Pr_Task`, que engloba todas las operaciones relativas a las tareas, su creación y manejo; `Pr_Semaphore`, encargada de todas las operaciones relativas a los semáforos y la sincronización; y `Pr_Time`, que gestiona los servicios de temporización. Además de estas tres, también se definen otras dos clases, llamadas `Pr_IRQManager` y `Pr_Kernel`. La primera se encarga de interaccionar con los servicios de manejo de interrupciones. La segunda se utiliza para configurar e inicializar las rutinas que sean necesarias y los servicios de planificación. Este diseño, dividido en dos niveles, facilita enormemente la portabilidad de la aplicación a distintos sistemas operativos y entornos de programación. Las clases de la capa de primitivas básicas son la única parte del código EDROOM dependiente de la plataforma. Su reducido tamaño y su sencillez han posi-

bilitado su adaptación a distintos sistemas operativos de tiempo real, como RTKernel o RTAI, entre otros.

2.6 Capa superior de la biblioteca de servicios

Esta capa proporciona la interfaz necesaria para su funcionamiento al código de los componentes generado por EDROOM. Los servicios que proporciona se pueden agrupar en primitivas de comunicaciones, temporización, gestión de memoria y manejo de interrupciones y de excepciones. En las siguientes secciones se describen de forma general todos estos servicios.

2.6.1 Servicio de comunicaciones

La comunicación entre los distintos componentes se puede realizar de forma síncrona o asíncrona. La biblioteca proporciona la primitiva de comunicación `send` para enviar de forma asíncrona a través de un puerto un mensaje con una determinada prioridad. El número de niveles de prioridad de los mensajes y los componentes pueden fijarse a conveniencia. Para implementar la comunicación síncrona, el sistema provee dos primitivas: `invoke` y `reply`. La primera envía el mensaje al componente de destino y se bloquea a la espera de recibir la respuesta. El componente destino responde al mensaje con la segunda primitiva.

2.6.2 Servicio de temporización

Este servicio permite programar temporizadores que provoquen el envío de un mensaje en un instante de tiempo determinado. Se soporta sobre dos primitivas llamadas `InformIn`, para establecer un intervalo de tiempo hasta el envío relativo al instante actual, e `InformAt`, que en fija el instante temporal concreto en el que el mensaje será enviado.

2.6.3 Gestión de memoria

El intercambio de información entre las tareas implica un acceso compartido a la memoria y, por lo tanto, es necesario establecer mecanismos que aseguren la exclusión mutua. La gestión de la memoria empleada por los distintos componentes para transmitir los mensajes se realiza mediante pilas (*pools*) de elementos que emulan la memoria dinámica. El uso de la memoria dinámica no es aconsejable en este tipo de sistemas, ya que la reserva tiene un tiempo de latencia grande y, sobre todo, no determinista, lo que afecta notablemente al rendimiento de los sistemas de tiempo real. Además, el mecanismo de liberación de la memoria por parte del receptor puede ser una fuente de errores que podría llegar incluso a dejar al sistema sin memoria. Cada componente cuenta con un conjunto de *pools* cuyo tamaño es definido según las necesidades de comunicación para ese tipo determinado de datos. El servicio de control de la memoria se encarga de reservar la memoria de los

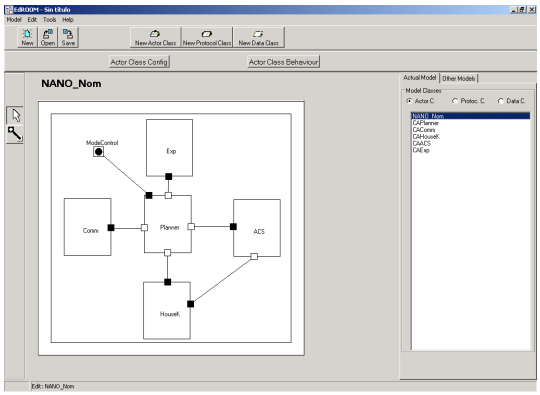

Figura 4: Ejemplo de ejecución de la herramienta gráfica EDROOM

pools en el momento de la inicialización del sistema. Cuando un componente desea enviar un dato en un mensaje debe pedir al *pool* correspondiente el dato, completar sus campos y enviar el mensaje utilizando una de las ya mencionadas primitivas de comunicación. El receptor por lo tanto no debe de ocuparse de liberar la memoria ya que la biblioteca de servicios realiza esa gestión de manera automática.

2.6.4 Manejo de interrupciones y excepciones

Con el fin de poder capturar los eventos provocados por las interrupciones, la herramienta incluye un tipo especial de clases protocolo destinado a tratar este tipo de eventos. El servicio de interrupciones gestiona los puertos asociados a estas clases protocolo de forma que a través de ellos se recibe un mensaje cada vez que la interrupción correspondiente se dispare. El servicio de manejo de excepciones ha sido incluido con el fin de posibilitar su tratamiento dentro del comportamiento de los componentes. Cuando el servicio es activado para un componente, los eventos debidos a excepciones que se produzcan durante la ejecución de su comportamiento son capturados y notificados al componente mediante el envío de un mensaje especial.

2.7 Interfaz gráfica

La herramienta gráfica tiene la interfaz mostrada en la Figura 4. En ella se puede ver un ejemplo de un sistema formado por varios componentes con sus respectivos puertos y conexiones. Actualmente EDROOM solamente funciona como entorno de desarrollo cruzado en sistemas basados en Microsoft Windows, aunque se está trabajando en adaptar la aplicación principal para que funcione en otros entornos y de manera independiente de la plataforma. Además de la herramienta de generación automática de código, el entorno EDROOM dispone de una segunda herramienta para facilitar el trazado de las aplicaciones. En la siguiente sección se describe su funcionamiento.

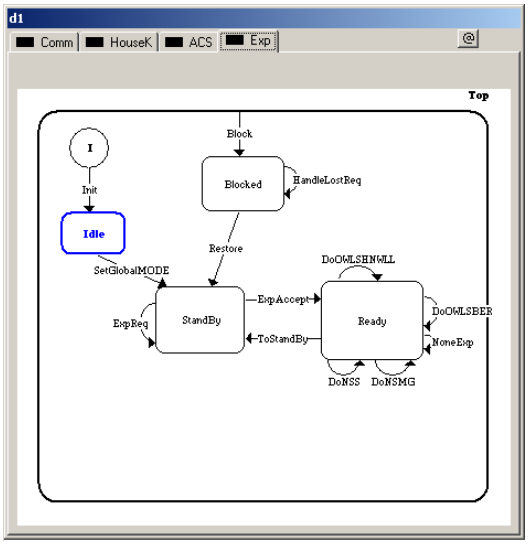

Figura 5: Ejemplo de ejecución de la aplicación de trazado

2.8 Herramienta de trazado

La herramienta EDROOM permite incluir en el código generado, de manera opcional, la funcionalidad necesaria para obtener, en tiempo de ejecución u *offline* una traza del sistema. Dicha traza almacena información referente al nivel de comportamiento EDROOM, esto es, sobre los eventos asociados con el diseño de los distintos componentes. Dicha información está formada por los detalles de las transiciones que ocurren entre los estados de los componentes, su identificador y una marca del tiempo en que se producen. Esta funcionalidad especial permite al desarrollador verificar el diseño gráfico del sistema y comprobar que se cumplen todas las restricciones temporales impuestas. Para procesar la información de traza, EDROOM dispone de una aplicación que, a partir de dicha traza, permite conocer de manera gráfica el estado en que se encuentra el sistema en cada momento, mostrando en cada caso todas las transiciones que tengan lugar. En la Figura 5 se puede observar una ejecución de esta aplicación. El estado marcado es el estado actual. Cuando se produce una transición, el cambio queda registrado, y la aplicación pasa a indicar el nuevo estado del sistema.

3 EJEMPLOS DE APLICACIÓN

La herramienta EDROOM ha sido utilizada para desarrollar el software de control de una maqueta de un ferry rápido [7]. El objetivo del estudio era la investigación sobre la posible reducción de la aceleración vertical de los ferrys rápidos empleando como actuadores un timón delantero y unos estabilizadores traseros, y cuyo ángulo de ataque era controlado por motores paso a paso. También se ha empleado para diseñar e implementar el software de vuelo del nano-satélite español NANOSAT del Instituto Nacional de Técnica

Aeroespacial (INTA) [6]. Este satélite se encuentra actualmente en funcionamiento y tiene como misión la realización de diversos experimentos con el objetivo de probar y validar nuevas tecnologías espaciales relacionadas con la nanotecnología. Durante ambos proyectos la experiencia de utilización de la herramienta ha sido muy positiva, destacando en las siguientes facetas:

- El modelado gráfico ha facilitado la comprensión del sistema por los distintos miembros del equipo de desarrollo.
- La incorporación de nuevos requisitos en fases avanzadas del proyecto se ha realizado con un reducido impacto gracias a la posibilidad ampliar la jerarquía de estados y los protocolos de comunicaciones.
- La portabilidad del modelo sobre distintas plataformas nos ha permitido anticipar la validación de gran parte de la funcionalidad sobre un PC cuando el *target* final aún no estaba disponible.

Los últimos trabajos que se están llevando a cabo van encaminados a fundir la herramienta EDROOM con un sistema operativo de tiempo real llamado ERCOS-RT diseñado expresamente para dar soporte a las aplicaciones generadas por ésta. Este sistema operativo ha sido desarrollado para ejecutarse sobre la arquitectura ERC32 [1], el estándar de la Agencia Espacial Europea (*European Space Agency* o ESA) para las misiones espaciales. El objetivo es la utilización conjunta del binomio EDROOM - ERCOS-RT para implementar aplicaciones de tiempo real empotradas en entornos con recursos limitados de memoria.

4 CONCLUSIONES

El desarrollo de sistemas de tiempo real es una tarea especialmente difícil. El empleo de formalismos y metodologías de diseño se torna fundamental a la hora de llevar a cabo el desarrollo y la implementación de estos tipos de software. Uno de estos formalismos es ROOM, un lenguaje de modelado basado en objetos.

EDROOM es una herramienta CASE de libre distribución inspirada en ROOM que facilita el diseño e implementación de sistemas de tiempo real. EDROOM permite editar un modelo que describa la estructura, la topología de comunicación y el comportamiento del sistema, y a partir de él, generar de manera automática el código de la aplicación. Dicho código se soporta sobre una Biblioteca de Servicios EDROOM que implementa las entidades y servicios correspondientes al nivel de abstracción del modelo. Esta biblioteca está dividida en dos capas, una independiente de plataforma y otra que da soporte a la primera y contiene el conjunto de primitivas básicas. Esta última capa encapsula al sistema operativo y son sólo sus primitivas las que deben reimplementarse a la hora de realizar una adaptación a un nuevo sistema operativo.

La herramienta EDROOM ha sido empleada, entre otros proyectos, para el diseño y la implementación del software de vuelo del satélite español NANOSAT del INTA.

La experiencia de uso ha sido muy positiva y ha facilitado, entre otras cosas, la mejor comprensión del sistema por parte de los integrantes del equipo, la incorporación de nuevos requisitos en etapas avanzadas del desarrollo y la validación de gran parte de la funcionalidad en plataformas basadas en PC.

REFERENCIAS

[1] SAAB Ericcson Space. 32-bit microprocessor and computer development programme - Final. ESA Contractor Report, 1997.

[2] Harel and David. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, (8):231–274, 1987.

[3] John Hatcliff, William Deng, Matthew B. Dwyer, Georg Jung, and Venkatesh Prasad Ranganath. Cadena: An integrated development, analysis, and verification environment for component-based systems. In *25th International Conference on Software Engineering (ICSE'03)*, 2003.

[4] Maria Herranz Molinero, Oscar Rodriguez Polo, Luis Pinuel, and Jesus Manuel de la Cruz. EDROOM: A case tool for the room based modelling and automatic code generation of real-time systems running under RTAI. In *Proceedings of the fifth real-time linux workshop*, November 2003.

[5] Oscar R. Polo, De la Cruz J. M., Giron-Sierra J.M., and Esteban S. Edroom. automatic c++ code generator for real-time systems modelled with room. In *NTCC2001 IFAC Conference*, November 2001.

[6] Oscar Rodriguez Polo, Luis de Salvador, Manuel Angulo, and Jesus Manuel de la Cruz. Development plan of the on board satellite software based on room modelling and evolution of component based prototypes. In *27th IFAC/IFIP Workshop on Real-Time Programming.*, 2003.

[7] Oscar Rodriguez Polo, S. Esteban, A. Grau, and J. M. de la Cruz. Control code generator used for control experiments in ship scale model. In *Proceedings of the CAMS2001 IFAC Conference*, 2001.

[8] Aitor Viana Sánchez, Oscar Rodriguez Polo, Oscar López Gómez, Martin Knoblauch Revuelta, Sebastián Sánchez Prieto, and Daniel Meziat Luna. EDROOM: A free tool for the uml2 component based design and automatic code generation of tiny embedded real time system. In *Proceedings of the 3rd European Congress on Embedded Real-Time Software ERTS*, 2006.

[9] Selic, B., Gulleckson, G., and Ward P.T. *Real-Time Object Oriented Modelling.* John Wiley and Sons, 1994.