Uso del lenguaje unificado en la modelización de sistemas operativos

Inés Casanovas, Nicanor Casas, Graciela De Luca
inescasanovas@gmail.com, pepecasas2002@yahoo.com.ar, graciela.edl@gmail.com
Universidad Tecnológica Nacional - FRBA
Departamento de Ingeniería de Sistemas

CONTEXTO

Este proyecto está siendo desarrollado en forma interdisciplinaria por las cátedras de Sistemas Operativos y Análisis de Sistemas de la carrera de Ingeniería de Sistemas de la Universidad Tecnológica Nacional- Regional Bs.As. Están involucradas las dos cátedras ya que en la primera se aplican todos los conceptos y prácticas de Sistemas Operativos y en la segunda se desarrollan los conceptos teóricos de Orientación a Objetos, UML en particular, ya que preliminarmente se pensó en la utilización de Rational como herramienta de modelado.

RESUMEN

El abordaje de la enseñanza de sistemas operativos orientados a objetos choca contra un elemento de singular peso: el escaso material de estudios que apunte en forma directa al centro del problema. Es por ello que en la actualidad las diferentes cátedras abordan la enseñanza de sistemas operativos tradicionales y muy pocas complementan con sistemas operativos orientados a objetos. En los trabajos existentes la arquitectura de integración del sistema está basada en una máquina abstracta que ofrece una funcionalidad mínima, y un sistema operativo que debe extender la funcionalidad de la máquina, chocando con la funcionalidad de arranque del sistema y en la utilización de herramientas de aplicación parcial, pues las estructuras existentes se basan en conceptos tradicionales de proceso, posición de memoria y archivo.
El objetivo es constatar que el modelado de un sistema operativo mediante técnicas de Lenguaje Unificado presenta inconvenientes para definir los procedimientos, que requiere un sistema operativo, en especial todo lo relacionado con la interacción de bajo nivel, siendo necesaria la creación o redefinición de herramientas que permitan la modelización de un sistema operativo general y que tenga la visión de un sistema integral orientado a objetos como parte fundante de un sistema operativo distribuido.

Palabras Claves: *UML (lenguaje unificado de modelado), Flexibilidad, Máquina abstracta, Encapsulamiento, Reusabilidad.*

1. INTRODUCCION

La característica principal del estudio a desarrollar es constatar que el modelado de un sistema operativo a través de técnicas de lenguaje unificado presenta inconvenientes para definir los procedimientos a bajo nivel que requiere un sistema operativo, en especial todo lo relacionado a la interacción entre lenguajes cercanos a la máquina, o lenguajes de bajo nivel.
Por otro lado todo software de base debe adaptarse a la modularidad de una máquina que en principio puede tener estructuras muy diferentes con otras que sirvan para la misma función.
Los modelos UML tienen significado para el análisis lógico y para la implementación física, facilitando con ello las construcciones de implementación, pero dificultando en muchos de los casos las estructuras de construcción.
Suzuki y Yamamoto [SUZ99], ya en 1999 proponían una extensión de UML para soportar aspectos, considerando a los aspectos como un clasificador en un meta-modelo.
Los objetos proporcionan modularidad y encapsulación, dado que separan claramente

la interfaz de construcción con la interfaz de implementación.
La comunicación cuenta con un mecanismo uniforme de alto nivel, la invocación de métodos o paso de mensajes entre objetos, que es bien conocido y para el que existen técnicas para conseguir implementaciones eficientes.
Los objetos proporcionan un medio adecuado para compartir información. La ventaja respecto a los ficheros compartidos es una semántica de más alto nivel y una mejor protección ante errores de programación.
Los sistemas estructurados en términos de objetos evolucionan fácilmente (más fáciles de extender y mantener), por reemplazo de sus partes. Además los subsistemas y objetos individuales podrán ser reutilizados con facilidad
Los sistemas de software de tiempo real, en particular los sistemas operativos, representan un desafío de modelización y diseño lógico porque tienen comportamientos orientados a eventos y contienen limitaciones temporales y de recursos. El análisis basado en escenarios (en todas sus modalidades) proporciona soluciones a las problemáticas de alto nivel, mientras que la programación orientada a objetos proporciona las adecuadas a las de bajo nivel. Queda pendiente de resolver el gap que se produce al tratar de modelizar estos sistemas en base a las técnicas propuestas por UML.
Por tanto entre los inconvenientes para definir los procedimientos a bajo nivel debemos tener en consideración especial todo lo relacionado con:

- la interacción entre lenguajes de bajo nivel
- las causas, efectos y responsabilidades a nivel de comportamiento de sistemas donde los escenarios pueden cambiar en tiempo de ejecución, y se requiere capacidad de refinamiento durante la variación de estos escenarios.

No importa cuan completo sea el conjunto de facilidades que proporcione un lenguaje de referencia, siempre se querrán realizar extensiones al mismo por varias razones que son muy entendibles, por ejemplo:

- Para mejorar la visión de los diagramas que se desarrollen
- Porque existen áreas oscuras que no pueden ser representadas por el lenguaje.
- Porque la evolución de las necesidades operativas requieran un mejor interacción entre las diferentes estructuras.

Por otro lado los estereotipos representan una nueva clase de elemento de modelado con la misma estructura de un elemento existente, pero con restricciones adicionales, un interpretación diferente de algún elemento, como ser un ícono y un tratamiento diferente de los generadores de código.
En la mayor parte de los sistemas operativos actuales, la aplicación de las herramientas de modelado y en especial las correspondientes a las de orientación a objetos se realiza de una manera parcial, dado que se sustentan en los conceptos tradicionales de proceso, posición de memoria y archivo. Por otro lado los mecanismos y políticas definidos por el sistema operativo están codificados de manera rígida en el mismo, y esto reduce la posibilidad de adaptarlos al entorno en el que va a prestar servicios.
Consideramos que una aproximación interesante a la integración de todas las tecnologías, la del modelado, las políticas y los mecanismos, es la creación de un sistema que ofrezca un soporte que permita la generación de un sistema operativo general y que tenga la visión de un sistema integral orientado a objetos como nervio central para el desarrollo de un sistema operativo distribuido.
En este sistema, todos los elementos, interfaces de usuario, aplicaciones, lenguajes, etc., comparten un mismo paradigma. La integración de la máquina abstracta y el sistema operativo se consigue a través de la arquitectura reflectiva de la primera, que permite definir la máquina como una estructura de objetos perfectamente cuantificables.

Uno de los inconvenientes que se registran en los trabajos que preceden es que la arquitectura para realizar la integración del sistema está basada en una máquina abstracta que puede o no estar orientada a objetos, pero que fundamentalmente ofrece una funcionalidad mínima, y un sistema operativo que tiende a ser orientado a objetos, que debe extender la funcionalidad de la máquina chocando con la funcionalidad de arranque del sistema.

Desde un punto de vista general un sistema operativo puede verse como la interacción entre dos componentes bien definidos: una base y un meta-nivel.

La base intenta solucionar el problema de las aplicaciones, tomándolas como entidades que reflejan la problemática real a resolverse y el meta-nivel es el responsable de que ese sistema base funcione manteniendo la información y describiendo de que manera debe resolver los conflictos el sistema base.

Si se integran los lenguajes de programación, como se indicó anteriormente, el programa fuente correspondería al sistema base y el intérprete del lenguaje corresponde al meta-nivel.

Al ser los sistemas de tiempo real orientados a eventos, la determinación de las especificaciones basadas en lenguaje natural y el proceso de análisis y modelización los diferencia notablemente del modelado de software de gestión, pues las técnicas tradicionales de determinación de requerimientos proporcionan modelos de alto nivel.

Se requiere un entendimiento detallado de dependencias, concurrencias, tiempo de respuesta y uso de recursos, es decir, cobran destacada importancia los requerimientos no funcionales. En este tipo de software los requerimientos han sido clasificados como requerimientos de comportamiento y requerimientos temporales (Ekelin C. y Jonson J, 1999, Real Time Systems Constrains, Alexandria, VA). Los primeros, como se relacionan con la funcionalidad, se identifican rápidamente; los temporales son el resultado de la interacción de los objetos componentes y mas difíciles de definir.

Una visión panorámica de los enfoques actualmente discutidos en los congresos y áreas de investigación del dominio de los Sistemas Operativos muestra los siguientes métodos:

a) UML Use Cases: es una aproximación basada en casos donde los actores definen requerimientos que son modelizados en casos de uso y expandidos en escenarios que esclarecen relaciones específicas entre actores y el sistema a nivel individual, usando lenguaje natural.

b) Diagramas de transición de estados: Glintz ha promovido el uso de estos diagramas como una mejora para la representación de los requerimientos de Sistemas Operativos a partir de un trabajo anterior de Harel. Este método propone la representación de escenarios sencillos, compuestos y abstractos, usando diagramas y texto formalizado, que permitiría vistas a nivel de detalle de eventos y sus transiciones. En su versión original no definía formas para manejar requerimientos temporales pero es posible de ser ampliado.

c) Notación jerárquica gráfica: fue propuesto por Regnell [REG96] y en este método, el escenario presenta un grafo de episodios conectados, donde se definieron notaciones de tiempo para mostrar excepciones e interrupciones, y cada evento a su vez es un diagrama de secuencia de mensajes. Se refleja también las dependencias entre escenarios y el entorno de ejecución.

d) Mapas de Casos de Uso (UCM): describen visualmente relaciones causales entre responsabilidades impuestas en estructuras de componentes abstractos Estas responsabilidades podrían ser tareas, acciones, operaciones etc. Mientras que los componentes pueden ser entidades de software (objetos, procesos etc.) o no (hardware, por ejemplo). Las relaciones causales vinculan

precondiciones o eventos disparadores, con postcondiciones y resultantes, secuencialmente o concurrentemente. Básicamente, son caminos que muestran el desarrollo o evolución temporal de un escenario dentro de un Caso de Uso, es decir es un modelo dinámico para representar las variaciones de comportamiento y de estructura.

2. LINEAS DE INVESTIGACION y DESARROLLO

El grupo de investigación desarrolla en la actualidad un sistema operativo de características didácticas el cual basas su análisis en través del uso de metodologías ágiles. Se presenta así, debido al intercambio en diferentes congresos, el interés de realizar un intento de llevar el mismo a diseño a través de UML, si bien existen muchos informes sobre las falencias que el lenguaje unificado de modelización tiene, pero los mismos no se refieren a sistemas operativos, ni a software de base.

3.RESULTADOS OBTENIDOS / ESPERADOS

3.1 Resultados alcanzados

Dado que el proyecto está en sus comienzos, hasta el momento no se han visto resultados, sino que todo se basa sobre el supuesto de que las técnicas de modelado, en función de la hipótesis de trabajo de este estudio, resultarían inadecuadas para representar claramente todos los atributos presentes en tiempo real.

3.2 Objetivos

Permitir que los alumnos puedan definir un universo de discurso, aplicando conceptos que son claves de la aplicación, sus propiedades internas y las relaciones entre cada una de ellas, generando a posteriori nuevas relaciones y establecer la factibilidad de cada una de ellas.

Permitir que los alumnos obtengan y generen conocimientos que permitan visualizar un sistema operativo en forma integrada a través de diagramas.

Permitir que los alumnos observen un sistema operativo bajo una herramienta de diseño de uso general ingresando diferentes modelos de máquinas y generando vistas estáticas de diagramas de clase.

Lograr que los alumnos puedan realizar estudios avanzados sobre sistemas operativos aprovechando los diferentes modelados básicos estructurados anticipadamente, para ser incrementados y corregidos.

Permitir implementar áreas estructurales, dinámicas y de gestión de modelos.

Permitir el análisis detallado de extensiones de UML que permitan salvar problemas, si los hubiera, y/o generar nuevos diagramas, no definidos, para facilitar la confección de software de base.

4. FORMACION DE RECURSOS HUMANOS

El grupo de investigación espera transferir los conocimientos alcanzados a los alumnos que cursan las materias de Análisis de Sistemas y Sistemas Operativos para proveerles de una estructura que enlace ambas materias.

Intercambiar información con Universidades Nacionales y Privadas del ámbito nacional e internacional publicando los avances que se realicen en diferentes congresos, teniendo como objetivo extender y realizar convenios de colaboración que permitan ampliar conocimientos de docentes e investigadores.

5. BIBLIOGRAFIA REFERENCIADA Y DE CONSULTA

[AG96] Allen Robert y David Garlan, “The Wright Architectural Description Language”, Technical Report, Carnegie Mellon University. 1996.

[ALL97] Allen Robert. “A formal approach to Software Architecture”. Technical Report, CMU-CS-97-144, 1997.

[BIN93] Binns Pam, Matt Englehart, Mike Jackson y Steve Vestal. "Domain- Specific Software Architectures for Guidance, Navigation, and Control". Technical report, Honeywell Technology Center, http://www-ast.tds-gn.lmco.com/arch/arch-ref.html, 1993.

[BUR98] Buhr R., 1998, Use Case Maps as Architectural Entities for Complex Systems, IEEE, Transactions on Software Engineering 24(12) pag. 1131-1155 - 1998.

[DH01] Eric Dashofy y André van der Hoek. "Representing Product Family Architectures in an Extensible Architecture Description Language". En Proceedings of the International Workshop on Product Family Engineering (PFE-4), Bilbao, España, 2001.

[HAR87] Harel D., State Charts: a visual formalism for complex systems. SCI Program, 1987

[JAC92] Jacobson J. et al., Object Oriented Software Engineering: a use case driven approach, Addison Wesley), 1992

[GLI00-a] Glinz Martin. "Problems and deficiencies of UML as a requirements specification language". En Proceedings of the 10th International Workshop of Software Specification and Design (IWSSD-10), pp. 11-22, San Diego, noviembre de 2000.

[GLI00-b] Glintz M., 2000, Improving the Quality of Requirements with Scenarios, Proceedings of the 2nd. World Congress for Software Quality, Japón

[Mon98] Monroe Robert. "Capturing software architecture design expertise with Armani". Technical Report CMU-CS-163, Carnegie Mellon University, Octubre de 1998.

[OOS99] OOSPLA'99, Workshop #23: "Rigorous modeling and analysis with UML: Challenges and Limitations". Denver, Noviembre de 1999.

[PIN89] Pinkert J. & L.Wear – Operating Systems: Concepts, Polices and Mechanisms, Englewood Cliffts NJ, Prentice Hall – First Edition – 1989

[REG96] Regnell et al., A Hierarchical Use Case Model with Graphical Representation, Proceedings of IEEE Internacional Symposium on Engineering of Computer Based Systems, 1996

[RUM99] Rumbaugh James, Ivar Jacobson, Grady Booch – El lenguaje unificado de modelado, Manual de Referencia – Addison Wesley - 1999

[SMC00] Standard Microsystems Corporation *Application note 6.12,* 2000

[SUZ99] Suzuki J. y Y. Yamamoto - Extending UML with Aspects: Aspect Support in the Design Phase - ECOOP Workshop on Aspect-Oriented Programming - 1999.

Revistas y Publicaciones

Diomidis Spinellis - Computer Standars & Interfaces 20:1-8 November 1998 "A Critique of the Windows Application Programming Interface" University of the Aegean

Internet

Akehurst David y Gill Waters. "UML deficiencies from the perspective of automatic performance model generation". OOSPLA'99 Workshop on Rigorous Modeling and Analysis with the UML: Challenges and Limitations, Denver, Noviembre de 1999.

http://www.cs.kent.ac.uk/pubs/1999/899/content.pdf,

Douglas Bruce Powel. "UML – The new language for real-timed embedded systems".

http://wooddes.intranet.gr/papers/Douglass.pdf, 2000.