Proyecto DYNAMICA

II Jornadas de Trabajo DYNAMICA

(DYNamic and Aspect-Oriented Modeling for Integrated Component-based Architectures)

En Málaga, 11 de noviembre de 2004

Organización:

Grupo de Ingeniería del Software y Sistemas de Información

ISeI

Colaboradores:

Universidad Politécnica de Valencia

Departamento de Sistemas Informáticos y Computación

ACTAS

II Jornadas de Trabajo DYNAMICA

(DYNamic and Aspect-Oriented Modeling for Integrated Component-based Architectures)

En Málaga, 11 de noviembre de 2004

Editores:

Nour Ali
Artur Boronat
Patricio Letelier
Jenifer Pérez

Grupo de Ingeniería del Software
y Sistemas de Información

ISaI

Financiado por el CICYT (Centro de Investigación Científica Y Tecnológica), Proyecto DYNAMICA (DYNamic and Aspect-Oriented Modeling for Integrated Component-based Architectures)

Presidente de las jornadas

Isidro Ramos Salavert

Comité Organizador

Presidente: **Patricio Letelier Torres**

Vocales: **Nour Ali Irshaid**

Artur Boronat Moll

Jennifer Pérez Benedí

Comité Técnico

Isidro Ramos Salavert, *Universidad Politécnica de Valencia*

Bárbara Álvarez Torres, *Universidad Politécnica de Cartagena*

Coral Calero Muñoz, *Universidad de Castilla-La Mancha*

Francisco José Rodríguez Urbano, *Universidad Carlos III*

Joaquín Lasheras, *Universidad de Murcia*

Subproyectos y Grupos Participantes

PRISMA – Plataforma OASIS para Modelos Arquitectónicos
TIC2003-07804-C05-01
Grupo de Investigación "Ingeniería de Software y Sistemas de Información" (ISSI)
Departamento de Sistemas Informáticos y Computación (DSIC)
Universidad Politécnica de Valencia (UPV)

SUBPROYECTO: ANCLA - Arquitecturas DiNámiCas para Sistemas de TeLeoperAción
TIC2003-07804-C05-02
Grupo DSIE (División de Sistemas e Ingeniería Electrónica
Departamento de Tecnologías de la Información y Comunicaciones
Universidad Politécnica de Cartagena (UPCT)

SUBPROYECTO: CALIPO: CALIdad en POrtales
TIC2003-07804-C05-03
Grupo ALARCOS
Departamento de Informática
Universidad de Castilla-La Mancha (UCLM)

SUBPROYECTO: CARATE - Controlador de Arquitectura Reconfigurable para aplicaciones de Teleoperación
TIC 2003-07804-C05-04
Departamento de Ingeniería de Sistemas y Automática (ISA)
Universidad Carlos III (UC3M)

SUBPROYECTO: PRESSURE - PREcise Software modelS and reqUirements REuse
TIC 2003-07804-C05-05
Grupo de Investigación en Ingeniería del Software (GIS)
Departamento de Informática y Sistemas
Universidad de Murcia (UMU)

Presentación

El Proyecto DYNAMICA (**DYN**amic and **A**spect-Oriented **M**odeling for **I**ntegrated **C**omponent-based **A**rchitectures) es un proyecto coordinado de tres años de duración financiado por el Ministerio de Ciencia y Tecnología. DYNAMICA surge por los intereses comunes de cinco grupos de investigación españoles: el grupo de Ingeniería del Software y Sistemas de Información (ISSI) de la Universidad Politécnica de Valencia (UPV), el grupo de División de Sistemas e Ingeniería Electrónica (DSIE) de la Universidad Politécnica de Cartagena (UPCT), el grupo ALARCOS de la Universidad de Castilla-La Mancha (UCLM), el grupo del Departamento de Ingeniería de Sistemas y Automática (ISA) de la Universidad Carlos III (UC3M) y el Grupo de Investigación en Ingeniería del Software (GIS) de la Universidad de Murcia (UMU).

Las segundas jornadas DYNAMICA nos proveen un espacio para compartir los resultados de este primer año del proyecto. Así, el objetivo de estas jornadas es promover la discusión de soluciones para enfrentar la dinámica del software desde una visión arquitectónica en dominios con problemas reales como los sistemas de teleoperación, los sistemas hidráulicos y los portales Web.

Se han recibido 16 contribuciones desde los diferentes nodos, reflejando el espíritu e ilusión que tenemos invertido en este proyecto. Además, las relaciones inter-nodos han sido intensas, mostrando que somos conscientes que es el único modo de conseguir un avance hacia un objetivo común es mediante el trabajo colectivo. La amistad y sinergia entre todos nosotros permitirá un excelente trabajo y un gran resultado en este proyecto.

Esperamos que las discusiones sean intensas y las disfrutemos al máximo.

Los Editores.

Índice

Sección 1: Análisis, Formalización y Transformación de Modelos

Sección 2: Métricas y Calidad

Sección 3: Arquitecturas Software y Dominios de Aplicación

Sección 5: Ingeniería de Requisitos

Una aproximación formal a las métricas para diagramas de estados UML

Cristina Vigueras, Marcela Genero, José Luis Fernández, José Antonio Cruz-Lemus, Ambrosio Toval, Mario Piattini

Grupo de Investigación de Ingeniería del Software.
Departamento de Informática y Sistemas, Universidad de Murcia.
Campus de Espinardo, 30071, Murcia
{vigueras, aleman, atoval}@um.es

Grupo ALARCOS
Departamento de Informática, Universidad de Castilla-La Mancha
Paseo de la Universidad, 4, 13071, Ciudad Real
{Marcela.Genero, JoseAntonio.Cruz, Mario.Piattini}@uclm.es

Resumen

En este trabajo presentamos la formalización de un conjunto de métricas de complejidad estructural y de tamaño de los diagramas de estados UML, utilizando el lenguaje de especificaciones formales Maude. Lo que se pretende es dar un carácter formal a un conjunto de métricas intuitivas, obteniendo métricas bien definidas y no ambiguas. Esta definición formal evitará problemas de interpretación a la hora de aplicar las métricas, lo que llevará a una correcta y mayor utilización de las métricas en la práctica y además permitirá la posibilidad de automatizar su cálculo. Para llevar a cabo esta tarea se ha utilizado Maude, un lenguaje que permite dar este carácter formal y que a su vez ya está implementado en un entorno de programación amigable.

Keywords: métricas, diagramas de estados UML, Maude

1.- Introducción

El lenguaje UML es un lenguaje de modelado gráfico, que ha sido especialmente diseñado para visualizar, especificar, construir y documentar los distintos aspectos – o vistas- de un sistema, de forma que se utilizan distintos tipos de diagramas para describir las distintas perspectivas del sistema. En este artículo nos centraremos en uno de estos tipos de diagramas, los diagramas de estados. Los diagramas de estados UML se utilizan para describir el comportamiento dinámico del sistema, y son de los más utilizados entre todos los diagramas de estados, de ahí su interés.

Debido al amplio uso de los diagramas de estados UML, hay un interés creciente, [1] [2], en medir la calidad de estos diagramas. Por lo tanto, deben estar bien expresados en cuanto a sintaxis, deben ser legibles y comprensibles, y deben representar adecuadamente el comportamiento del sistema (es decir, han de tener la semántica correcta). Este artículo se centra en un aspecto de la calidad que es la comprensibilidad. Si bien una forma complementaria de medir la calidad de los diagramas sería comprobar que tienen una sintaxis correcta, como se propone en [3] y [13], esto esta fuera del alcance de este artículo y lo planteamos como trabajo futuro.

Con el fin de obtener diagramas de estados que sean más comprensibles, hemos tomado y ampliado un conjunto de métricas definidas en [6]. Estas métricas miden la complejidad estructural y el tamaño de los diagramas de estados UML de forma que

cuanto mayor valor tengan estas métricas, más complejos serán los diagramas y por tanto menos comprensibles.

Si bien estas métricas fueron definidas y validadas tanto teórica como empíricamente [6], su definición se ha planteado de manera intuitiva o informal utilizando lenguaje natural. Esta definición informal puede ser ambigua en muchos casos y puede llevar, como menciona Baroni et al. [1], a errores en la interpretación a la hora de aplicar las métricas en la práctica. Dos equipos distintos pueden obtener resultados completamente diferentes al aplicar una métrica particular al mismo diagrama. Por ello el principal objetivo de este trabajo es la definición formal de las métricas utilizando el lenguaje de especificaciones formales Maude [12], partiendo de las especificaciones algebraicas para los diagramas de estados de UML [4].

Este artículo consta de las siguientes secciones: en la sección 2 se ofrece una visión general de los diagramas de estados UML, haciendo énfasis en las características que más nos interesan. En la siguiente sección se da una descripción de las métricas propuestas para la complejidad estructural y tamaño de los diagramas de estados UML. En la sección 4 se muestra la formalización de éstas métricas. La sección 5 presenta una breve revisión de algunos trabajos relacionados con nuestro artículo. Por último en la sección 6 se establecen las conclusiones de nuestro trabajo y las líneas sobre las que seguimos trabajando.

2.- Los Diagramas de Estados UML

En [5] se encuentra una descripción completa y detallada de UML v1.5 y en particular de los diagramas de estados UML. En este artículo solo daremos una visión general de estos diagramas centrándonos en aquellos aspectos directamente relacionados con las métricas formalizadas.

Los **diagramas de estados UML** representan el comportamiento dinámico de entidades, especificando su respuesta a la recepción de instancias de eventos. Normalmente se utilizan para especificar el comportamiento de una clase. Gráficamente se representan mediante un grafo. Estos diagramas están formados por los siguientes componentes:

1) Estados, un estado es una condición durante la vida de un objeto o una situación durante la cual se satisface alguna condición, se realiza alguna acción, o se espera algún evento. Conceptualmente un objeto permanece en un estado durante un intervalo de tiempo. Especial mención merecen los *estados compuestos*, que son aquellos se pueden descomponer en dos o mas subestados concurrentes (llamados regiones) o bien en subestados disjuntos mutuamente excluyentes.

2) Eventos, un evento es una ocurrencia de un suceso que puede disparar una transición entre estados. Los eventos pueden ser de varios tipos:

- Eventos de cambio (Change Event). Tienen asociada una expresión booleana y el evento ocurre cada vez que el valor de la expresión cambia de falso a verdadero.
- Eventos de señal (Signal Event). Es un evento que provoca el disparo de una transición.
- Eventos de llamada (Call Event). Cuando se recibe una llamada para ejecutar una operación.
- Eventos de tiempo (Time Event). Tiene asociada una expresión de tiempo.

3) Transiciones, una transición es una relación entre dos estados que indica que una instancia en el primer estado entrará en el segundo estado y realizará las acciones especificadas, cuando ocurra un evento concreto. Cuando ocurre este cambio de estado se dice que la transición se "dispara". El disparador de una transición es la ocurrencia del evento que etiqueta dicha transición.

3.- Visión general de las métricas existentes para los diagramas de estados UML

En [6] se hace un estudio sobre el estado del arte en cuanto a las métricas que se pueden aplicar a diagramas de estados UML [5]. En dicho estudio se obtuvieron las siguientes conclusiones:

- La mayoría de estas propuestas de métricas no han ido más allá de la etapa de definición.
- No hay métricas definidas para los diagramas de estados UML.
- El trabajo relacionado con su validación teórica y empírica es escaso.

Además queremos destacar que en todas las propuestas las métricas se definieron de manera intuitiva o informal, utilizando en la mayoría de los casos lenguaje natural.

Viendo la carencia de métricas para medir aspectos estructurales de los diagramas de estado UML, en [6] se definen metodológicamente un nuevo conjunto de métricas para la complejidad estructural y el tamaño de los diagramas de estados UML (ver Tabla 1). Para asegurar la validez de las mismas, se sometieron a un proceso de validación teórica y empírica:

- Como resultado de la validación empírica, siguiendo el marco DISTANCE [14], se obtuvo que todas las métricas realmente miden el atributo que pretenden medir y además que estas métricas están definidas en la escala de ratio, lo que garantiza su utilización en estudios empíricos.
- Como resultado de la validación empírica se ha obtenido que la mayoría de las métricas parecen ser buenos indicadores de la comprensibilidad de los diagramas de estados UML. Esto significa que un mayor valor de estas métricas llevará a que estos diagramas que sean más difíciles de comprender.

	Nombre de la Métrica	Definición de la Métrica
Métricas de tamaño	NEntryA(Número de acciones de Entrada)	El número total de acciones de entrada, i.e. las acciones realizadas cada vez que se introduce un estado.
	NExitA(Número de acciones de Salida)	El número total de acciones de salida, i.e. las acciones realizadas cada vez que se abandona un estado.
	NA(Número de Actividades)	El número total de actividades (do/Activity)
	NSS (Número de estados simples)	El número total de estados considerando también los estados simples dentro de los estados compuestos.
	NCS(Número de estados compuestos)	El número total de estados compuestos.
	NE(Número de eventos)	El número total de eventos.
	NG(Número de centinelas)	El número total de condiciones centinela.
Métricas de complejidad estructural	McCabe(Número ciclomático de McCabe)	Se define como |NSS| - |NT| + 2
	NT(Número de Transiciones)	El número total de transiciones, considerando las transiciones comunes (aquellas cuyos estados fuente y destino son distintos), las transiciones inicial y final, las auto-transiciones (aquellas cuyo estado fuente y destino es el mismo) y las transiciones internas (transiciones dentro de un estado que responden a un evento pero sin abandonar el estado).

Tabla 1. Resumen de las métricas propuestas

Como se puede observar en la Tabla 1, la definición de las métricas se ha realizado de manera informal utilizando lenguaje natural.

Al examinar la métricas definidas se observa que las métricas propuestas son contadores de los elementos que pueden aparecer en un diagrama de estados UML, pero en dicha enumeración se echan en falta dos contadores, ambos referentes a las transiciones, que hemos añadido como nuevas métricas. Por una parte no se tienen en cuenta las acciones que se deben ejecutar al disparar una transición; a esta métrica la hemos llamado NIA, y por otra parte, igual que distinguimos entre estados simples y estados compuestos, también podemos hacer esta distinción entre transiciones simples, que son aquéllas que se disparan siempre (gráficamente se distinguen porque sólo tienen un nombre en la transición) y aquéllas que conllevan una condición de guarda o centinela y además implican un conjunto de acciones. Por ello aparece un nuevo contador de transiciones complejas NTC, que se divide en 3:

- NTCG1, contador de las transiciones que tienen asociado un evento y una condición de guarda o centinela.
- NTCG2, contador de transiciones que tienen asociado un evento y una lista de acciones internas
- NTCG3, contador de transiciones que tienen asociado un evento, una condición de guarda y una lista de acciones internas.

4.- Formalización

La elección de Maude como lenguaje para formalizar estas métricas obedece a varios motivos, entre ellos podemos destacar los siguientes:

- Se trata de un lenguaje de especificaciones formal, por lo que se elimina la posible ambigüedad en la definición de las métricas.
- Se tiene experiencia en el uso de este lenguaje, de hecho ya se posee una especificación algebraica de los diagramas de estados UML, aunque en una versión anterior de Maude
- Maude es ejecutable, es decir podemos calcular los valores de estas métricas, una vez que los diagramas estén transcritos a Maude. Además, actualmente existe un entorno de programación con un depurador, aún en periodo de pruebas, que facilita bastante la programación en este lenguaje.

Hace un tiempo, trabajar con Maude implicaba trabajar en Linux, lo cual podía suponer un inconveniente para algunos, pero actualmente también se puede ejecutar bajo Windows.

4.1 Formalización de los diagramas de estados

Para la formalización de los diagramas de estados UML hemos partido de las especificaciones formales construidas en Maude de la versión UML 1.3 [4], utilizada para comprobar la propiedad de ortogonalidad en un diagrama de estados UML. El objetivo es combinar la apariencia intuitiva de la notación gráfica con la precisión de los lenguajes de especificación formal.
El uso de Modelos Formales proporciona una serie de ventajas, de entre ellas podemos destacar la generación automática de prototipos.

Basándonos en el trabajo realizado en [4], para el modelado de los diagramas de estados UML v1.5 en Maude hemos utilizado la jerarquía de módulos que se muestra en la Figura 1.

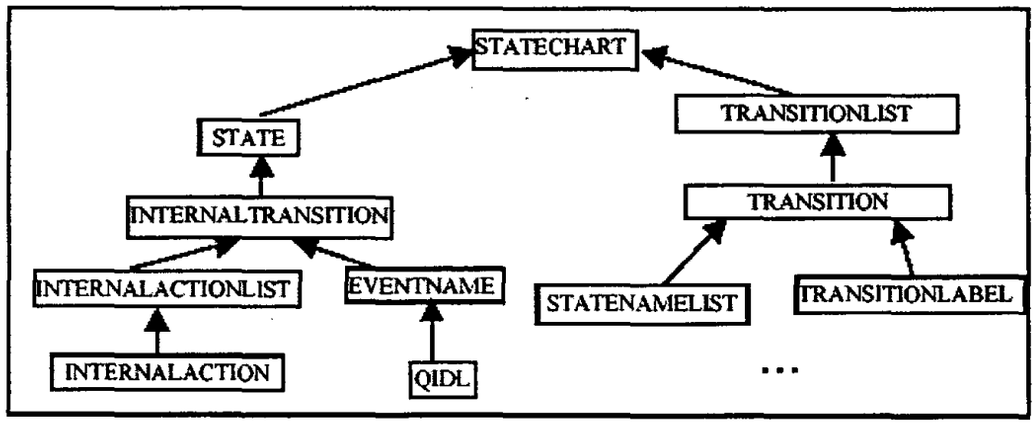

Figura 1. Jerarquía de módulos Maude para la especificación algebraica de los diagramas de estados UML

Supongamos un diagrama de estados UML complejo [11], como el de la Figura 2.

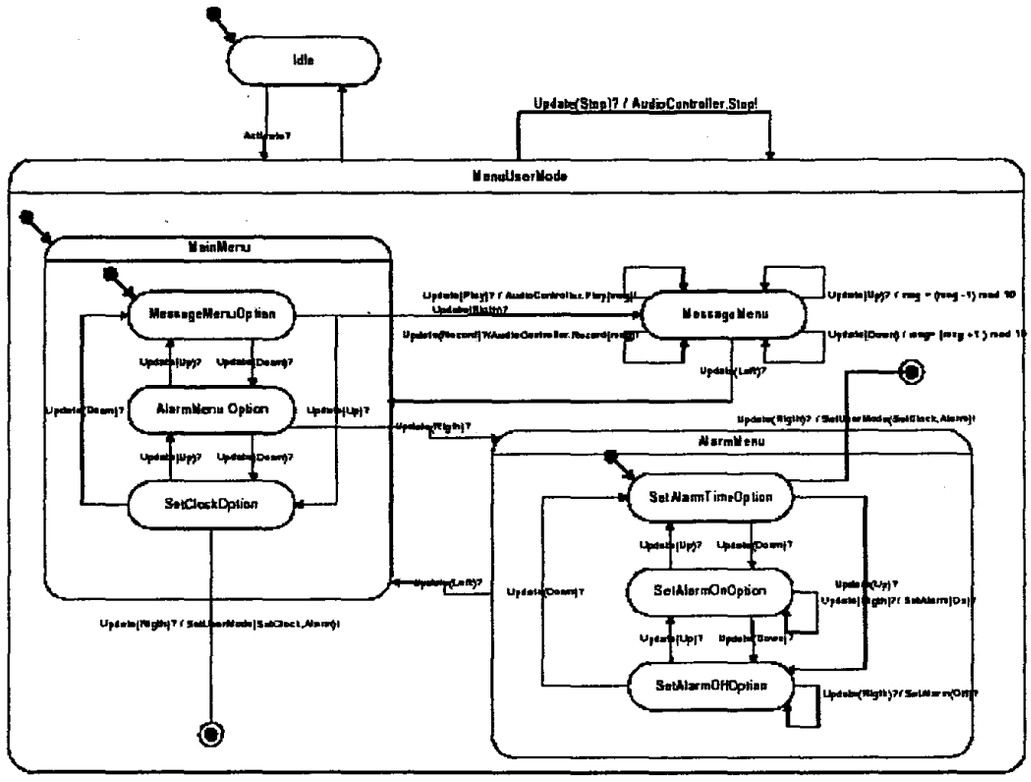

Figura 2. Diagrama de estados de un menú de usuario de un reloj despertador

Para este diagrama parte de su especificación algebraica sería:

```
      op Idle : -> StateVertex .
      op MenuUserMode : -> StateVertex .
      op MainMenu : -> StateVertex .
      op MessageMenu : -> StateVertex .
      op AlarmMenu : -> StateVertex .
      op transitions : -> TransitionList .
      op SD : -> StatechartDiagram .

      eq MainMenu = sequentialState('MainMenu, noInternalAction,
                     initialState ('i3)
                    simpleState('MessageMenuOption, noInternalAction)
                    simpleState('AlarmMenuOption, noInternalAction)
                    simpleState('SetClockOption, noInternalAction)
                    finalState ('f1, noInternalAction)) .
      eq transitions =
            transition (noTransitionName, 'i1, 'Idle,
      transitionLabel (noEvent, noGuardCondition, noActionExpression ))
            transition (noTransitionName, 'Idle, 'MenuUserMode,
      transitionLabel (event('TakeDown, noParameter), noGuardCondition,
      noActionExpression ))
      ...
            transition (noTransitionName, 'i3, 'MessageMenuOption,
      transitionLabel (noEvent, noGuardCondition, noActionExpression ))
            transition (noTransitionName, 'MessageMenuOption, 'MessageMenu,
      transitionLabel (event('Update?, parameter(in, 'Right, 'Button)),
      noGuardCondition, noActionExpression ))
      ...
eq SD = statechartDiagram ( Window, transitions) .
```

4.2 Formalización de las métricas

Las métricas formalizadas son las de la Tabla 1 y las nuevas propuestas. Como se comentó anteriormente son contadores, y por tanto sus especificaciones son muy similares. Como ejemplo, la especificación algebraica para la métrica NIA es la siguiente:

```
...
op NIA : StatechartDiagram -> Int .
op analizarITA : TransitionList -> Int .
...
  eq NIA(statechartDiagram (S, TL)) = analizarITA(TL) .
...
eq analizarITA(noTransition) = 0 .
eq analizarITA(transition(TN, NEVNL, NEVNL2, noTransitionLabel)) = 0 .
eq analizarITA(transition(TN, NEVNL, NEVNL2, transitionLabel(E, GC, AEL))) =
       if (AEL = = noActionExpression) then
           0
       else
           length(AEL)
       fi .
eq analizarITA(transition(TN, NEVNL, NEVNL2, noTransitionLabel) TL ) =
      analizarITA(TL) .
eq analizarITA(transition(TN, NEVNL, NEVNL2, transitionLabel(E, GC, AEL))T ) =
      if  (AEL = = noActionExpression) then
            analizarITA(TL)
      else
            length(AEL) + analizarITA(TL)
      fi .
...
```

Así pues, si calculamos estas métricas para el diagrama de estados UML de la Figura 2 obtendríamos los resultados mostrados en la Tabla 2.

Metrics	Value	Explanation
NentryA	0	No hay acciones de entrada en los estados del diagrama.
NexitA	0	No hay acciones de salida en los estados del diagrama.
NA	0	No hay acciones de actividad (do/Activity) en los estados del diagrama.
NSS	8	Idle, MessageMenuOption, AlarmMenuOption, SetClockOption, MessageMenu, SetAlarmTimeOption, SetAlarmOnOption, SetAlarmOffOption.
NCS	3	MenuUserMode, MainMenu, AlarmOption
NE	25	Uno por cada flecha etiquetada que conecta dos estados en el diagrama
NG	0	No hay condiciones de guarda.
NT	30	Una por cada flecha que conecta dos estados en el diagrama
McCabe	20	|NSS-NT+2| = |8-30+2| = |-20| = 20
NTG1	0	No hay transiciones con evento y condición de guarda.
NTG2	9	De las 29 transiciones hay 8 que además de un evento tienen una acción interna asociada.
NTG3	0	No hay transiciones con evento, condición de guarda y lista de acciones internas.
NIA	9	Hay 8 transiciones con una acción interna cada una.

Tabla 2. Resultados de las métricas para el ejemplo de la Figura 2

5.- Trabajos relacionados

Recientemente, diversos enfoques han abordado el tema de la definición formal de métricas, planteando diversas soluciones. Uno de ellos, [1], propone una formalización de las métricas basada en el meta-modelo de UML, para ello describe diferentes conjuntos de métricas orientadas a objetos utilizando OCL (Object Constraint Language), lenguaje que pertenece al estándar UML. Este enfoque permitiría dar métricas no ambiguas, así como establecer comparaciones entre conjuntos formalizados de métricas. Lo que pretende es buscar un compromiso entre el carácter formal de la especificación (utilizando un lenguaje semiformal) y que a su vez sea comprensible.
Por otra parte hay otro trabajo, [2], que ofrece un nuevo enfoque para la especificación y análisis y razonamiento sobre los diagramas de estados. Este estudio combina tres líneas, por un lado los diagramas de estados UML, por otro el π-calculus y la tercera línea es un comprobador de modelos (*model checker*) llamado NuSMV. El diseño de un sistema primero se especifica en diagramas de estados UML, después se formaliza mediante π-calculus y finalmente se verifica automáticamente mediante NuSMV.
Nuestra propuesta ofrece la ventaja de que al utilizar un lenguaje formal como Maude tanto para los diagramas de estados como para las métricas garantiza por un lado la corrección de los mismos (debida al propio lenguaje) y también la no ambigüedad. Además, a pesar de que Maude es un lenguaje para especificaciones formales, se están realizando esfuerzos para facilitar su uso. Así pues, este lenguaje ya está disponible para la plataforma Windows, y además hay una versión de prueba, bastante estable, de un entorno de trabajo que ofrece un depurador.

6.- Conclusiones y trabajos futuros

En este trabajo presentamos la formalización de un conjunto de métricas de complejidad estructural y de tamaño de los diagramas de estados UML, utilizando el lenguaje de especificaciones formales Maude [12]. Para realizar este trabajo, se ha utilizado la versión 2.1 de Maude, bajo el sistema operativo Linux. Maude ofrece entornos de programación tanto para Linux como, recientemente, para Windows.

Este trabajo puede integrarse en una herramienta CASE como puede ser RIVIERA [15], de forma que la generación de las especificaciones formales de los diagramas de estados se haga automáticamente, como ya se hace con los diagramas de clases UML. Así, también podemos corregir los posibles errores de sintaxis que pudiese haber en los diagramas. Teniendo en cuenta todo lo anterior se podrían comparar distintos diagramas, con una sintaxis correcta, para un mismo sistema y elegir aquel que ofrezca un menor valor para estas métricas, garantizando así que es el más comprensible.

Otra línea de trabajo que hemos de afianzar es trabajar sobre la equivalencia de diagramas, pues una vez que podamos medir la calidad de los diagramas con este conjunto de métricas y las nuevas que vayan surgiendo, y podamos establecer si dos diagramas de estados son equivalentes, podremos determinar cual de los dos es mejor o por lo menos más sencillo y por tanto ese deberá ser el que sigamos utilizando en el resto de etapas de desarrollo del sistema. En esta línea de trabajo puede ser interesante

la investigación de D. Latella [10], que propone un método para establecer si dos diagramas son equivalentes mediante simulación.

Otro enfoque en el que nos gustaría trabajar es construir métricas para medir la calidad de la semántica de los diagramas. Una vía prometedora es el Model Checking ([7], [8], [9]), ya que es mediante simulación como podemos comprobar como se comportará el sistema.

Agradecimientos

Este trabajo de investigación esta parcialmente financiados por los proyecto CALIPO (TIC2003-07804-C05-03) y PRESSURE (TIC2003-07804-C05-05), ambos concedidos por la "Dirección General de Investigación del Ministerio de Ciencia y Tecnología".

Referencias

[1] A.L. Baroni, S. Braz, F. Brito. "Using OCL to Formalize Object-Oriented Design Metrics Definitions". *in ECOOP'02 Workshop on Quantitative Approaches in Object-Oriented Software Engineering,* 2002

[2] V. Lam, J. Padget. "Symbolic Model Checking of UML Statechart Diagrams with an Integrated Approach", Proceedings of the 11th *IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*, 2004.

[3] C. Denger, M. Ciolkowski. "High Quality Statecharts through Tailored, Perspective-Based Inspections". Proceedings of the 29th EUROMICRO Conference "New Waves in System Architecture", 2003.

[4] J. L. Fernández, A. Toval. "Can Intuition become Rigorous? Foundations for UML Model Verification Tools". Proceedings of the *4th IEEE International Symposium on Software Reliability Engineering*, 2000.

[5] "Guía de Notación UML" – OMG- Unified Modeling Lenguaje, v1.5", 2003 http://www.omg.org/technology/documents/formal/uml_2.htm

[6] J.A. Cruz-Lemus, M. Genero, M. Piattini. "Metrics for UML Statechart Diagrams". Capítulo del libro: "*Conceptual Model Metrics*". Genero, M., Piattini, M. y Calero, C. (eds.), Ed. Collage Press, 2004

[7] S. Gensi, D.Latella, M. Massink. "Model Checking UML Statechart Diagrams using JACK". Proceedings of the *4th IEEE International Symposium on High-Assurance System Engineering*, 1999.

[8] W. Dong, J. Wang, X. Qi, Z Qi. "Model Checking UML Statecharts". Proceedings of the 8th *Asia-Pacific Software Engineering Conference*, 2001.

[9] G. Csertán, G. Huszerl, I. Majzik, Z. Pap, A. Pataricza. "VIATRA- Visual Automated Transformations for Formal Verification and Validation of UML Models". Proceedings of the 17th *IEEE International Conference on Automated Software Engineering*, 2002.

[10] D.Latella, M. Massink. "A Formal Testing Framework for UML Statechart Diagrams Behaviours : From Theory to Automatic Verification". Proceedings of the 6th *IEEE International Symposium on Hig Assurance System Engineering*, 2001.

[11] I. Porres, J. Lilius "Digital Sound Recorder: A case study on designing embedded systems using the UML notation", 1999.

[12] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, Carolyn Talcott (2003). "Maude 2.1.1 Manual. Versión 1.0". Web: http://maude.csl.sri.com/

[13] Reggio G. y Wieringa R. J., "Thirty one Problems in the Semantics of UML 1.3 Dynamics". Workshop on Rigorous Modeling and Analysis of the UML Challenges and Limitations, conjuntamente con OOPSLA'99, Denver, Colorado, USA, November 2, 1999.
http://www.disi.unige.it/person/ReggioG/PublicationData.html#ReggioWieringa99a

[14] Poels G. and Dedene G. (1999). DISTANCE: A Framework for Software Measure Construction. *Research Report DTEW9937*, Dept. Applied Economics, Katholieke Universiteit Leuven,

[15] http://riviera.dif.um.es/