Un Modelo Evolutivo para Sistemas Cooperativos

Nuria Medina
Dept. de Lenguajes y Sistemas Informáticos
ETS Ingeniería Informática
Univ. de Granada
18071 Granada
nmedina@ugr.es

Visitación Hurtado
Dept. de Lenguajes y Sistemas Informáticos
ETS Ingeniería Informática
Univ. de Granada
18071 Granada
mhurtado@ugr.es

Lina García Cabrera
Dept. de Informática
EPS Paraje Las Lagunillas
Univ. de Jaén
23071 Jaén
lina@ujaen.es

José Luis Garrido
Dept. de Lenguajes y Sistemas Informáticos
ETS Ingeniería Informática
Univ. de Granada
18071 Granada
jgarrido@ugr.es

Resumen

En este trabajo se presenta una propuesta para obtener una especificación dinámica y evolutiva de un sistema cooperativo. Dicha propuesta está basada en una extensión del modelo SEM-HP y toma como punto de partida para dicha especificación los diagramas de tareas utilizados en AMENITIES para el diseño de sistemas cooperativos.

1. Introducción

En la era de las nuevas tecnologías los sistemas orientados hacia la comunicación, la colaboración y la coordinación entre grupos tienen para las organizaciones un especial interés. En este sentido en el ámbito de la disciplina CSCW (Computer Supported Cooperative Work) [6], cabría destacar la relevancia que actualmente están teniendo estos sistemas.

Desde el punto de vista de la Ingeniería del Software y de Sistemas, existen propuestas metodológicas que apoyan el proceso de desarrollo de dichos sistemas. Entre ellas hemos considerado AMENITIES [2], como método estructurado para la construcción de modelos cooperativos, ya que permite integrar el análisis social con el diseño del sistema. Sin embargo no se debe olvidar que el éxito de estos sistemas depende en gran medida de su capacidad para soportar los cambios tanto del entorno como del grupo; tareas modeladas y resto de conceptos presentes en el dominio del problema (roles que desempeñan los actores, etc.).

En ese sentido, el objetivo final de este trabajo es presentar un marco metodológico que permita obtener una especificación dinámica y evolutiva de un sistema cooperativo, a través de un modelo hipermedia evolutivo y adaptativo basado en SEM-HP [1]. El modelo cuenta con mecanismos que garantizan la consistencia del sistema especificado tras un cambio y automatiza la propagación del mismo. Además, desde el punto de vista del usuario, permite una navegación dinámica filtrada por roles y adaptada con métodos de soporte a la orientación, mejorando de esta forma el trabajo en grupo.

En la siguiente sección se presenta un resumen de la metodología utilizada para la obtención de los diagramas de tareas que permitirán modelar las distintas actividades que se llevan a cabo en el sistema; en la sección 3 se describe brevemente los fundamentos y arquitectura del modelo SEM-HP. La sección 4 ilustra a través de un ejemplo, los pasos necesarios para obtener una especificación dinámica y evolutiva del sistema cooperativo, basándose en el modelo descrito en la sección previa. Por último, se presentan las conclusiones del trabajo.

2. Metodología AMENITIES

AMENITIES [2] es una metodología basada en modelos de comportamiento y tareas para el análisis, diseño y desarrollo de sistemas cooperativos.

La metodología parte de marcos teóricos cognitivos (teoría de la actividad, cognición distribuida, etc.) y metodológicos (ingeniería del software, análisis y modelado de tareas, etc.). Se centra en el concepto de grupo para reflejar aspectos relevantes de su comportamiento (tareas a realizar, dinámicas, etc.) y estructura (organización, roles sociales, etc.). El objetivo de la metodología es abordar de forma sistemática el análisis y diseño del sistema cooperativo facilitando el desarrollo posterior del software [3]. AMENITIES proporciona un marco de referencia conceptual y metodológico, pero además propone

una metodología concreta incluyendo las siguientes fases generales: análisis del sistema y obtención de requisitos, modelado y diseño del sistema, análisis de dicho modelo y desarrollo del software.

El modelo de sistema a construir, denominado *modelo cooperativo*, es el núcleo central de AMENITIES. Está formado por un conjunto de modelos de comportamiento y de tareas organizados jerárquicamente, los cuales son de utilidad para describir y representar cualquier sistema cooperativo desde el punto de vista de su estructura y funcionamiento. Su principal propósito es ofrecer una especificación del sistema independiente de su implementación, proporcionando así una mejor comprensión del dominio del problema.

El método estructurado que se propone para la construcción del *modelo cooperativo* es el paso intermedio que permite integrar el análisis social con el diseño del sistema. Para la construcción de este modelo se propone la notación COMO-UML [4] basada en los diagramas de estados y actividades del lenguaje UML [5], siendo por tanto una notación eminentemente gráfica que incluye ciertas partes declarativas:

- Los **diagramas de estados** de COMO-UML se utilizan para modelar el comportamiento de un grupo en base a los roles presentes en la organización y los posibles cambios dinámicos que pueden llevar a cabo los actores entre estos roles.
- Los **diagramas de actividades** de COMO-UML se utilizan para describir las tareas individuales y cooperativas. Un estado de actividad va a representar el estado de realización de una unidad de trabajo. Normalmente, la ejecución progresa sin eventos externos y los eventos de iniciación/terminación de estados de actividad se asumen implícitamente. Al igual que en el lenguaje UML, los diagramas de actividades pueden contener estados de subactividad y de acción, decisiones, división/reunión del control en/de actividades concurrentes (*fork-join*), y cualquier otro elemento de los diagramas de estados.

3. Evolución y Adaptación en SEM-HP

SEM-HP [1] (modelo SEMántico, Sistémico y Evolutivo para Sistemas Hipermedia) es un modelo evolutivo y adaptativo que asiste al autor en el proceso de desarrollo y mantenimiento de sistemas hipermedia. Proporciona al autor una arquitectura (figura 1) que separa los aspectos relacionados con el diseño de la estructura conceptual de los relacionados con la navegación del sistema hipermedia. Para ello, ofrece una doble división. Una vertical, formada por cuatro sistemas (de *memorización*, *presentación*, *navegación* y *aprendizaje*) interrelacionados y en interacción que asisten al autor en su proceso de desarrollo y al usuario en su navegación. La otra, horizontal, distingue dos niveles de abstracción en cada uno de los sistemas: *sistema* y *meta-sistema*.

Con el nivel menos abstracto, *sistema*, se recogen algunas de las características del modelo y se proporcionan ciertas funciones tanto a los restantes sistemas como a los usuarios (ver sección 3.1). El meta-sistema, por su parte, incluye los mecanismos evolutivos (acciones evolutivas, restricciones y propagación del cambio) que permiten integrar y propagar cambios en la estructura (y como consecuencia, en el funcionamiento) de cada sistema.

Con las *acciones evolutivas* el autor construye y modifica su diseño hipermedia. Para garantizar la integridad del sistema, el meta-sistema sólo realiza los cambios que garantizan su consistencia, esto es, aquellos que satisfacen las *restricciones* impuestas por el modelo y por el autor. Finalmente, la modificación de algún elemento de alguno de los sistemas puede requerir el cambio de otros elementos del mismo (*propagación interna*) o de los otros sistemas (*propagación externa*) para lograr una co-evolución completa.

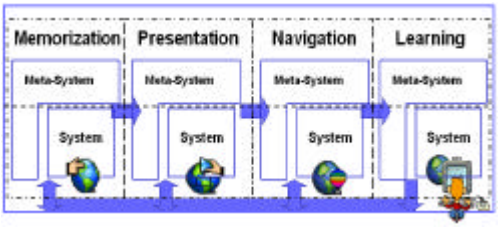

Figura 1. Arquitectura de SEM-HP

3.1. Los Sistemas de SEM-HP

El **Sistema de Memorización** almacena, estructura y mantiene el dominio conceptual y de información del sistema hipermedia, a través de la Estructura Conceptual (EC). Una EC es una representación ontológica o red semántica formada por nodos y arcos etiquetados. Los nodos pueden ser de dos tipos: conceptos e ítems. El conjunto de ítems de información se cataloga mediante los conceptos asociándolos a éstos, de esta manera, la navegación básica, queda determinada por las relaciones semánticas que existen entre los conceptos. Gracias a los mecanismos evolutivos de SEM-HP es posible cambiar la estructura conceptual y garantizar la consistencia de dicho cambio.

El **Sistema de Presentación** permite seleccionar vistas o presentaciones (subconjuntos) de la estructura conceptual. Durante la navegación se utilizará la información del *modelo del usuario* para elegir la parcela de conocimiento que más le guste o se adapte a los intereses del usuario.

El **Sistema de Navegación** determina el orden en el que se puede navegar la información ofrecida por las presentaciones. Para ello se proporcionan *restricciones de orden* que establecen si desde un ítem se puede ir a otro basándose en las relaciones conceptuales que el lector puede seguir y en los ítems visitados anteriormente.

Finalmente, el **Sistema de Aprendizaje** se encarga de modelar al usuario y de adaptar la estructura (elegir la presentación más adecuada) y la navegación del sistema hipermedia. Para ello, se basa en la información recogida en el *modelo de usuario* y en métodos adaptativos.

4. Evolución de Sistemas Cooperativos basada en SEM-HP

En esta sección se describen, a través de un ejemplo, los pasos necesarios para obtener una especificación que soporte la evolución de un sistema cooperativo, basándose para ello en el modelo SEM-HP anteriormente descrito. Concretamente, el sistema considerado es un Centro de Coordinación de Emergencias, encargado de distribuir tareas y organizar recursos en caso de accidentes graves, y en el ejemplo se mostrará la especificación, mediante SEM-HP, de parte de la actividad del sistema.

Se considera previamente definida la estructura de la organización (roles sociales presentes en el grupo) y las características estáticas y dinámicas de los roles implicados (conexión entre los actores y las tareas).

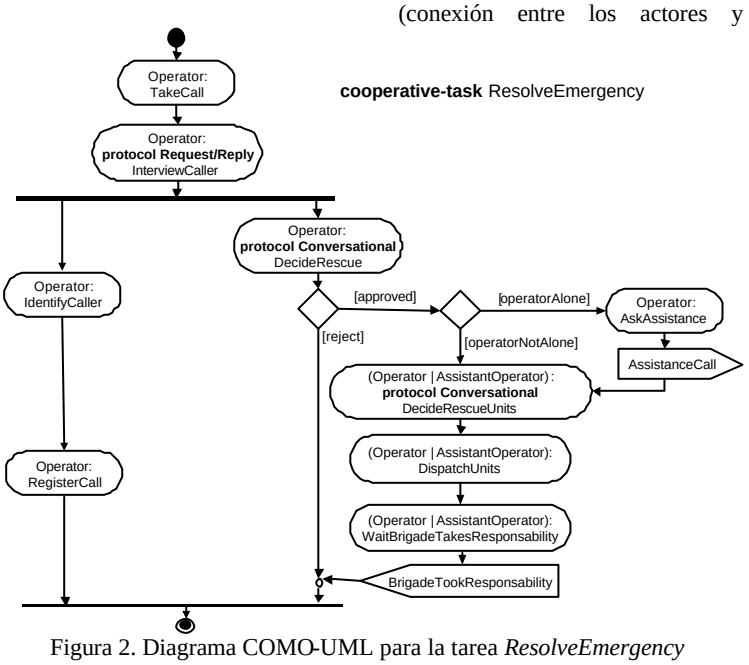

Figura 2. Diagrama COMO-UML para la tarea *ResolveEmergency*

Por ello, partimos del *diagrama de tarea* COMO-UML mostrado en la figura 2. Dicho diagrama especifica la tarea principal del sistema: *ResolveEmergency*, que comienza a raíz de una llamada de emergencia (evento *EmergencyCall*) y finaliza una vez que las unidades de salvamento (policía, bomberos, asistencia médica, etc.) llegan al escenario del accidente.

4.1. Representación con SEM-HP

Dada la especificación AMENITIES anterior, es posible obtener la representación correspondiente en SEM-HP. Dicha representación consta de dos componentes interrelacionadas: una Estructura Conceptual y un conjunto de reglas de orden. En este caso la Estructura Conceptual (EC) se transforma en una representación ontológica de las actividades consideradas y las relaciones existentes entre éstas. Las *reglas de orden* (Ro), por su parte, son las reglas lógicas obtenidas parcialmente a partir de la EC encargadas de definir las condiciones necesarias para la ejecución de cada actividad.

4.1.1 Estructura Conceptual

La figura 3 muestra la EC creada para la tarea *ResolveEmergency*. Su construcción es inmediata a partir de la especificación COMO-UML de la figura 2. Básicamente, consiste en insertar un concepto por cada actividad existente en el diagrama de tarea, y posteriormente seleccionar la relación adecuada que existe entre la actividad considerada y sus sucesoras.

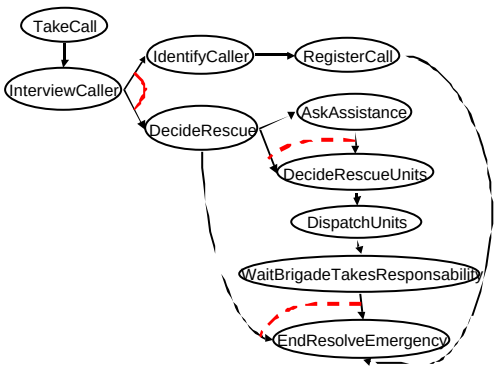

Figura 3. EC para la tarea *ResolveEmergency*

En la figura 3, se puede observar cómo a continuación de la actividad *DecideRescue* es posible ejecutar tres conjuntos distintos de actividades según las circunstancias que acontezcan en el sistema, comenzando cada uno de ellos por las actividades: *AskAssistance*, *DecidedRescueUnits* o *EndResolveEmergency*.

Asimismo, para la actividad *InterviewCaller*, existen dos actividades sucesoras, *IdentifyCaller* y *DecideRescue*, sin embargo a diferencia del caso anterior donde se ejecutará sólo una de las sucesoras, ahora deben ejecutarse ambas. Esta diferencia es expresada mediante un arco de línea continua que une las flechas que llegan a las citadas actividades de ejecución concurrente.

De modo parecido, cuando una actividad tiene varias actividades predecesoras, se asume que deben haberse realizado todas ellas, salvo que aparezcan unidas por un arco de línea discontinua. Así, por ejemplo, para la ejecución de la actividad *EndResolveEmergency* es necesario que se haya completado la actividad *RegisterCall*, y la actividad *DecideRescue* o la actividad *WaitBrigadeTakesResponsability*.

Con el propósito de separar los aspectos de representación y ejecución, las condiciones particulares que determinan la realización de una u otra de las actividades posibles no son expresadas en la EC, sino en las Ro.

4.1.2 Reglas de Orden

Una Ro está estructurada en dos partes: cabeza y cuerpo. La cabeza de la regla coincide con una actividad representada en la EC, mientras que el cuerpo especifica a través de predicados lógicos el responsable de esa actividad y las circunstancias necesarias para iniciar su ejecución. El actor responsable queda definido a través del rol requerido (predicado *rol*) junto con un conjunto de capacidades adicionales que éste debe poseer (predicados *capacidad*). Las circunstancias de ejecución incluyen predicados para expresar la actividad o actividades que deben haberse completado de forma previa (*previo*), la recepción de eventos externos (*evento*) y otros requisitos que afectan al estado del sistema o a eventos internos producidos en el mismo (*requisito*).

Las Ro son generadas inicialmente de forma automática a partir de la EC, pero deben ser completadas después por el desarrollador del

sistema, o en este caso que nos ocupa, derivadas del diagrama de la tarea. Una Ro inicial contiene únicamente los predicados *previo* deducidos a partir de las relaciones conceptuales presentes en la EC. En la tabla 1 pueden verse las Ro creadas por defecto para las actividades *TakeCall* y *DecideRescueUnits*.

Ro(*TakeCall*): true → *TakeCall*
Ro^1(*DecideRescueUnits*): previo(*AskAssistance*) → *DecideRescueUnits*
Ro^2(*DecideRescueUnits*): previo(*DecideRescue*) → *DecideRescueUnits*

Tabla 1. Ejemplo de reglas de orden iniciales

La actividad *TakeCall* no requiere ninguna actividad previa ya que es la primera en ejecutarse. Por su parte, la actividad *DecideRescueUnits* tiene dos reglas de orden, una para cada una de sus posibles actividades predecesoras. De acuerdo a la especificación de la figura 2, el desarrollador completaría las reglas como sigue:

Ro(*TakeCall*): rol(*Operator*) ∧ evento(*EmergencyCall*) → *TakeCall*
Ro^1(*DecideRescueUnits*): rol(*Operator*) ∧ requisito(*approved*) ∧ requisito(*operatorAlone*) ∧ evento(*AssistanceCall*) ∧ previo(*AskAssistance*) → *DecideRescueUnits*
Ro^2(*DecideRescueUnits*): rol(*Operator*) ∧ requisito(*approved*) ∧ requisito(*operatorNotAlone*) ∧ previo(*DecideRescue*) → *DecideRescueUnits*
Ro^3(*DecideRescueUnits*): rol(*AssistantOperator*) ∧ requisito(*approved*) ∧ requisito(*operatorAlone*) ∧ evento(*AssistanceCall*) ∧ previo(*AskAssistance*) → *DecideRescueUnits*
Ro^4(*DecideRescueUnits*): rol(*AssistantOperator*) ∧ requisito(*approved*) ∧ requisito(*operatorNotAlone*) ∧ previo(*DecideRescue*) → *DecideRescueUnits*

Tabla 2. Reglas de orden modificadas

Se observa que cuando una actividad puede ser desempeñada por dos roles diferentes, como en el caso de *DecideRescueUnits*, sus reglas de orden se duplican para cada rol: Ro^1 y Ro^2 para *Operator* y Ro^3 y Ro^4 para *AssistantOperator*.

4.2. Evolución con SEM-HP

Una vez obtenida la especificación de una tarea puede surgir la necesidad de realizar cambios como consecuencia de modificaciones en la organización o en los objetivos del sistema. Los mecanismos evolutivos de SEM-HP constituyen un soporte automático para garantizar y propagar dichos cambios.

Así por ejemplo, supongamos que el desarrollador decide eliminar la actividad *AskAssistance*. Entonces, automáticamente se pone en marcha el siguiente mecanismo de propagación del cambio encargado de mantener la consistencia del sistema modelado:

- En primer lugar, se borran en la EC todas las relaciones que parten o llegan a la actividad eliminada. En éste caso, la flecha que va desde *DecideRescue* hasta *AskAssistance* y la que parte de éste hacia *DecidedRescueUnits*.
- Si al hacerlo, alguna actividad queda desconectada se elimina también, repitiendo el proceso hasta que la EC no presente partes inconexas.
- Por otro lado, se descartan las Ro asociadas a la actividad eliminada. Así como, los requisitos *previo* definidos sobre ella en el resto de Ro.
- Finalmente se eliminan las Ro en cuyo cuerpo se exigía únicamente como actividad previa la actividad desaparecida. En el ejemplo, las reglas Ro^1 y Ro^3.

Además, de eliminar conceptos (actividades) y relaciones en la EC, existen otras formas de evolución. Algunas de ellas, pueden afectar a la especificación de una tarea, pero tener su origen en cambios producidos en la estructura del sistema. Por ejemplo, esto ocurriría cuando el desarrollador elimina un rol que ha dejado de ser válido en la organización:

- En este caso, la propagación del cambio implica borrar las Ro cuyo responsable es el rol eliminado (Ro^3 y Ro^4 si se elimina el rol *AssistantOperator*).
- De manera que si desaparecen todas las Ro asociadas a una actividad (esa actividad únicamente podía ser realizada por el rol

excluido) la actividad también se elimina en la forma anteriormente descrita.

4.3. Ejecución con SEM-HP

A la hora de ejecutar una tarea cooperativa, como por ejemplo *ResolveEmergency*, es muy útil proporcionar a cada actor la EC que modela la tarea, de modo que éste pueda visualizar gráficamente las actividades requeridas para llevarla a cabo.

Además, las Ro permiten al sistema determinar automáticamente cuáles de las actividades involucradas en la tarea pueden ser realizadas en un momento dado. De este modo, es posible guiar al actor a través de la secuencia correcta de ejecución para dicha tarea.

Al mismo tiempo, la visualización de la EC es filtrada para cada actor mostrándole únicamente las actividades que puede realizar en función de su rol y capacidades. De nuevo esta información se deduce a partir de las Ro, analizando el responsable de cada actividad.

Como puede observarse en la figura 4, la EC proporcionada a un actor con el rol *AssistantOperator* oculta en color gris claro las actividades en las que no participa. Marca con el símbolo de validación cada una de las actividades que han sido completadas adecuadamente (caso de *DecideRescueUnits* y *DispatchUnits*), con el símbolo de ejecución las que están en progreso y con el símbolo apuntador si están a la espera de que alguien las ejecute (*WaitBrigadeTakesResponsability*).

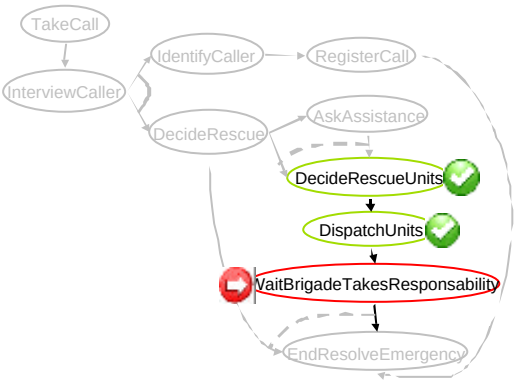

Figura 4. Ejecución de la tarea *ResolveEmergency* para un *AssistantOperator*

5. Conclusiones

La especificación de un sistema cooperativo es un proceso dinámico que exige realizar cambios como consecuencia de modificaciones en la organización o en los propios objetivos del sistema. La propuesta esbozada en este trabajo es novedosa en tanto en cuanto soporta dichos cambios por medio de un conjunto de mecanismos evolutivos y de adaptación, facilitando así la tarea del ingeniero de sistemas. Además, también es innovadora desde el punto de vista del usuario ya que proporciona un entorno gráfico en el que es posible la visualización de las actividades involucradas en cada una de las tareas, permitiendo un filtrado de las mismas en función de los roles y capacidades de los actores, asimismo cuenta con mecanismos que guían al actor a través de la secuencia correcta de ejecución para dicha tarea.

Referencias

[1] García-Cabrera, L. Rodríguez, M.J. Parets, J. Evolving Hypermedia Systems: a Layered Software Architecture. Journal of Software Maintenance and Evolution: Research and Practice. Wiley. Vol 14. pp: 389-406. 2002.

[2] Garrido, J.L., Gea, M., Rodríguez, M.L.: Requirements Engineering in Cooperative Systems. Requirements Engineering for Sociotechnical Systems. Chapter XIV. IDEA GROUP, Inc.USA (2005).

[3] Garrido, J.L., Paderewski, P., Rodríguez, M.L., Hornos, M.J. & Noguera, M. (2005) A Software Architecture Intended to Design High Quality Groupware Applications. 4th International Workshop on System/Software Architectures (IWSSA'05), USA.

[4] Garrido, J.L. *AMENITIES: Una metodología para el desarrollo de sistemas cooperativos basada en modelos de comportamiento y tarea.* Tesis Doctoral. Universidad de Granada, 2003.

[5] Object Management Group: *Unified Modelling Language Specification.* http://www.omg.org (September 2001)

[6] Greenberg, S. Computer-supported cooperative work and groupware. London: Academic Press Ltd. London (1991).