

Modelado de Aplicaciones con Procesos Concurrentes y Distribuidos

Universidad Nacional de la Matanza, Departamento. de Ingeniería e Investigaciones Tecnológicas

Mg. Giulianelli, Daniel Alberto
dgiulian@unlam.edu.ar

Ing. Rodríguez, Rocío Andrea
rrodri@unlam.edu.ar

Ing. Vera, Pablo Martín
pablovera@unlam.edu.ar

Resumen

En este paper se presenta al DEA "Diagrama de Estados Activos", el cual fue desarrollado por el equipo de trabajo, para poder modelar todos los aspectos de las aplicaciones con procesos concurrentes y distribuidos. Como punto de partida se evaluaron los distintos diagramas de UML 2.0 y viendo que ninguno de ellos se adaptaba por completo a las necesidades presentadas, se elabora éste modelo que toma las características de algunos diagramas de UML (Lenguaje Unificado de Modelado) y agrega elementos necesarios para este tipo de aplicaciones. El DEA permite visualizar en un único diagrama aspectos que de modelarse con UML implicarían la construcción de varios diagramas y el uso de estereotipos.

Palabras Claves

UML, DEA, DTE, Semáforos, Monitores, Cardinalidad, Concurrencia, Distribuido, Procesos, Modelado, Software, Actividades, Estados.

Antecedentes

Con la intención de modelar aplicaciones con procesos concurrentes y distribuidos usando UML 2.0 (ver introducción [6] y [2]), nace en el año 2005 este trabajo de investigación. Viendo que para modelar dichas aplicaciones era necesario utilizar un conjunto de diagramas y estereotipos, surge la idea de desarrollar un modelo que permita unificar todos los conceptos necesarios para una correcta representación

en un único diagrama. Producto de la interacción con pares de la disciplina y luego de varios trabajos preliminares surge dicho diagrama al que hemos denominado DEA (Diagrama de Estados Activos).

A fin de difundir los resultados obtenidos en las distintas versiones preliminares del presente trabajo, realizamos ponencias en los siguientes eventos [10]:

- 1) Congreso Argentino de Ciencias de la Computación. Universidad Nacional de Entre Ríos (CACIC 2005).
- 2) Jornada de Jóvenes Investigadores de Universidades Nacionales organizada por la Universidad Nacional de San Luis (JI2005).
- 3) Congreso Argentino de Ciencias de la Computación organizado por la Universidad Nacional de San Luis (CACIC 2006).
- 4) Jornada Iberoamericana de Ingeniería de Software e Ingeniería del Conocimiento, organizado por la Pontificia Universidad Católica del Perú (JIISIC07).

Uno de los objetivos del presente trabajo es presentar las mejoras realizadas sobre el
Uno de los objetivos del presente trabajo es presentar las mejoras realizadas sobre el DEA y realizar una comparativa de las ventajas de nuestra propuesta con respecto a

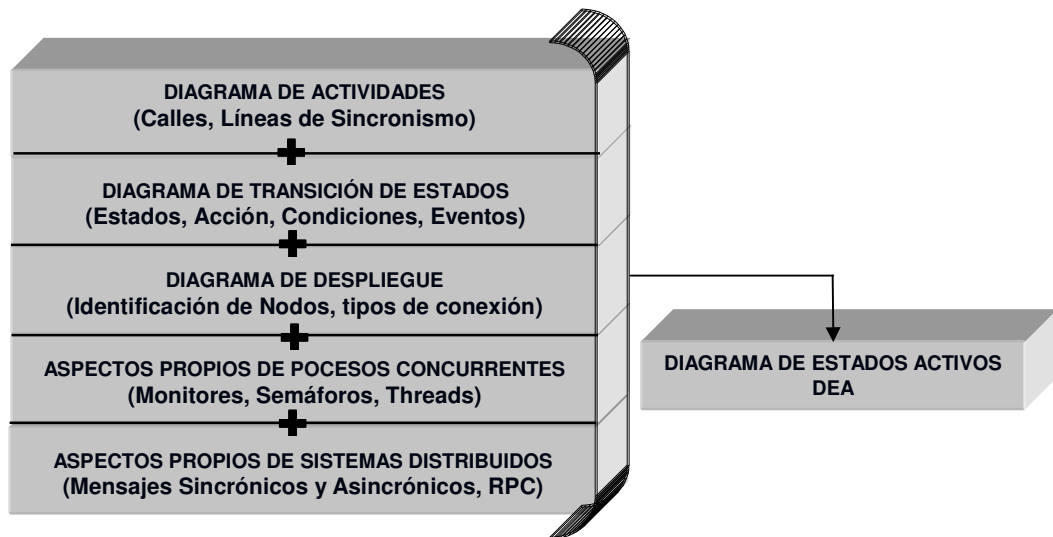


Figura 1: Elementos que conforman al DEA

UML 2.0. Con este fin se presenta el modelado de una misma aplicación utilizando ambas metodologías.

Introducción

Tomando como referencia a UML nos propusimos construir un Diagrama que reuniera las características necesarias para poder con él modelar procesos concurrentes y distribuidos.

Para ello tomamos las particularidades del Diagrama de Transición de Estados (que cubren la vista dinámica de un sistema), las correspondientes al Diagrama de Actividades (que muestra el flujo de control entre actividades), algunas características del Diagrama de Despliegue (que cubre la vista estática de un sistema) e incorporando características propias de los sistemas con procesos concurrentes y distribuidos, construimos un modelo al que hemos denominado “Diagrama de Estados Activos (DEA)”.

Este modelo que proponemos lo consideramos adecuado para:

1) Representar un proceso distribuido: Se utilizan las calles que caracterizan al clásico Diagrama de actividades [7] [14], para determinar en que nodo se realizará la ejecución de ciertos procesos. Para aquellos casos donde el medio de comunicación entre los nodos sea de relevancia, se indicará con una línea entre ambos el tipo

de vínculo utilizado, tal como se haría en UML en un Diagrama de Despliegue.

2) Representar la concurrencia de procesos: Se utilizan las líneas de sincronismo del Diagrama de actividades.

3) Detallar la información de los estados: Además de indicarse el nombre del estado se puede detallar como en el clásico DTE acciones de entrada y salida, transiciones internas y eventos diferidos.

4) Mostrar el cambio de estados: Se utilizan las transiciones del DTE indicando la o las condiciones, así como el evento y en el caso que existan, las acciones que permiten el paso de estado.

Particularidades de la propuesta

Del análisis del modelo presentado surgen las siguientes conclusiones:

1) Al agregar al clásico Diagrama de Transición de Estados las calles, queda indicado si el proceso comenzado en cierto estado al pasar a otro involucra o no un cambio de nodo. Es decir que al cumplirse cierta condición podrá continuarse el proceso en otro nodo (calle). Es importante destacar que la acción de cambio de nodo puede no estar asociada a una condición, en ese caso el cambio de nodo se producirá sin evaluar condiciones.

2) Las líneas de sincronismo pueden compartirse entre varios nodos por lo tanto el flujo del procesamiento se continúa en un nodo específico cuando este obtenga una

respuesta de los otros nodos que están procesando en forma paralela, ya sea por necesitar un resultado o por la finalización de los mismos

Particularidades del modelado de procesos distribuidos: El procesamiento distribuido requiere que los procesos alojados en distintos host se comuniquen de alguna manera para poder intercambiar información. Existen dos formas distintas de realizar dicha comunicación: mediante el envío de mensajes y por medio de la utilización de RPC (Remote Procedure Call – Llamadas a procedimientos remotos).

A su vez los mensajes pueden ser asincrónicos o sincrónicos. Un mensaje sincrónico obliga al emisor a esperar una respuesta antes de continuar con sus tareas. Por el contrario si es asincrónico se envía el mensaje y se continúa con el resto de las tareas. Cabe destacar que los RPC son en su mayoría llamadas sincrónicas ya que actúan como simples llamadas a procedimientos como si estuvieran en una misma computadora.

Para modelar la comunicación entre los procesos se mantiene la nomenclatura habitual de UML donde un mensaje sincrónico se representa con una punta de flecha rellena y un mensaje asincrónico con una punta flecha abierta (es recomendable consultar el manual de especificación de UML 2.0 [12]). Respetando esta nomenclatura proponemos para mayor claridad diferenciar los RPC de los mensajes usando la nomenclatura mostrada en la Figura 2.


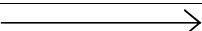
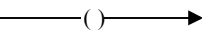
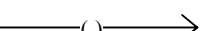
Mensaje Sincrónico	
Mensaje Asincrónico	
RPC Sincrónico	
RPC Asincrónico	

Figura 2: Nomenclatura mensajes y RPC

Los nodos que interactúan pueden tener diferentes tipos de conexiones físicas entre sí, lo que en algunas ocasiones es

importante destacar, ya que según sea el tipo de enlace habrá ciertas velocidades de transferencia que variarán. Lo que puede permitir decidir derivar ciertos procesos a un determinado nodo u a otro, o manejar distintos tiempos de time out para las respuestas. Al igual que en el diagrama de despliegue de UML se indicará en el DEA través de una línea entre cada par de nodos la característica del enlace.

Particularidades del modelado de procesos concurrentes: El modelado del procesamiento concurrente puede requerir la diferenciación de los distintos hilos (threads) de ejecución del sistema. Si bien la propuesta al incorporar las líneas de sincronismo del diagrama de actividades ya muestra los procesos o hilos que se ejecutan en paralelo, se propone también para una notación más clara en casos en los que se detallan varios estados dispares dentro de cada hilo. Se propone utilizar una notación de calles con líneas punteadas dentro de la calle principal del nodo para indicar el procesamiento independiente de cada hilo (ver Figura 3).

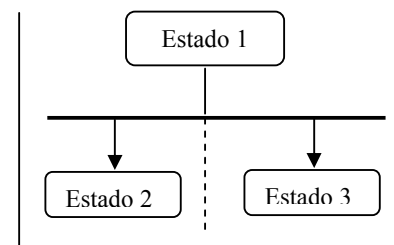


Figura 3: Subcalles para los hilos de ejecución

Cuando se utilizan hilos es muy posible que existan recursos que se deben compartir y por lo tanto es necesario administrar su acceso, ya que solamente un thread puede utilizarlo en un momento dado. Dos métodos comunes para la administración de recursos en un ambiente concurrente son los semáforos y los monitores [4].

1) Los semáforos son variables especiales que señalizan el acceso a los recursos utilizando dos primitivas (ejemplo: signal y wait) que retardan momentáneamente el acceso a un recurso en uso.

2) Los monitores son facilidades que brindan los lenguajes de programación y cumplen con la misma función que los semáforos pero de una forma más transparente ya que no es necesario manejar en forma manual un envío de primitivas signal y wait desde los procesos sino que esto se maneja en forma automática al encolar un proceso en el recurso controlado por el monitor.

Proponemos dos construcciones gráficas para indicar recursos compartidos y el método de acceso a los mismos. Dentro de estas construcciones es posible especificar el tipo de recurso al que nos estamos refiriendo (ver Figura 4).

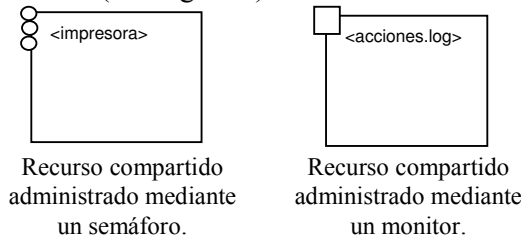


Figura 4: Nomenclatura designada para Semáforos y Monitores

Cardinalidad: Cuando se cuenta con múltiples conexiones provenientes desde distintos nodos, generalmente existe un proceso principal que monitorea las conexiones y al llegar nuevas conexiones crea distintos hilos para cada una de ellas. A menudo, las acciones que realizan cada uno de esos hilos son idénticas, por lo tanto proponemos la utilización de una nomenclatura, similar a un objeto con varias ocurrencias en UML, pero aplicado a las calles que está representando el hilo en ejecución (ver Figura 5).

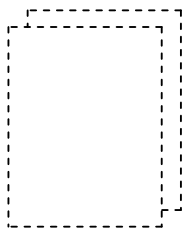


Figura 5: Las acciones que se realizan "n" veces se indican dentro de esta representación gráfica.

En la Figura 6 en la que puede observarse el modelado de una aplicación por medio del DEA. Esta aplicación consiste en un juego online multijugador. Por cada Jugador que se conecta al servidor, deben ocurrir una serie de estados ("Recibiendo conexión", "Leyendo configuración" y "Registrando jugador") que se realizarán de la misma forma para cada uno de ellos.

Metodología de Modelado

Se propone a continuación un listado de pasos que permitirán realizar un Diagrama de Estados Activos, de forma sistemática.

1) Identificar los distintos nodos o locaciones donde se va a ejecutar la aplicación para así definir con cada una de ellos las calles del diagrama.

2) Identificar el nodo desde el cual se va a iniciar el proceso o la funcionalidad que estamos modelando y ubicar la calle que lo representa en el centro del diagrama para minimizar el cruce de líneas entre las demás calles ya que es usual que éste sea el nodo principal que comanda al resto.

3) Determinar cuales conexiones entre nodos son relevantes de destacar e indicar entre dichos nodos el tipo de conexión física. Solo se deberán marcar las conexiones entre los nodos que tengan cierta relevancia como por ejemplo que puedan tener problemas de velocidad, ancho de banda, etc.

4) Comenzar el modelado desde el nodo ubicado en la calle central, usando la nomenclatura habitual del Diagrama de Transición de estados empezando con el círculo lleno que indica el comienzo del proceso.

5) Continuar el modelado utilizando de forma habitual los estados, cambios de estado y transiciones del DTE pero teniendo en cuenta que ahora podemos utilizar la nomenclatura del diagrama de actividades cuando tenemos actividades que se realizan en paralelo.

6) Cuando el flujo de control principal o uno de los flujos secundarios salen del nodo actual utilizar la nomenclatura de mensajes

o RPC propuesta para clarificar de que forma se realiza dicha comunicación.

independientemente de que nodo venga la conexión, englobar dicha funcionalidad dentro de un recuadro de cardinalidad.

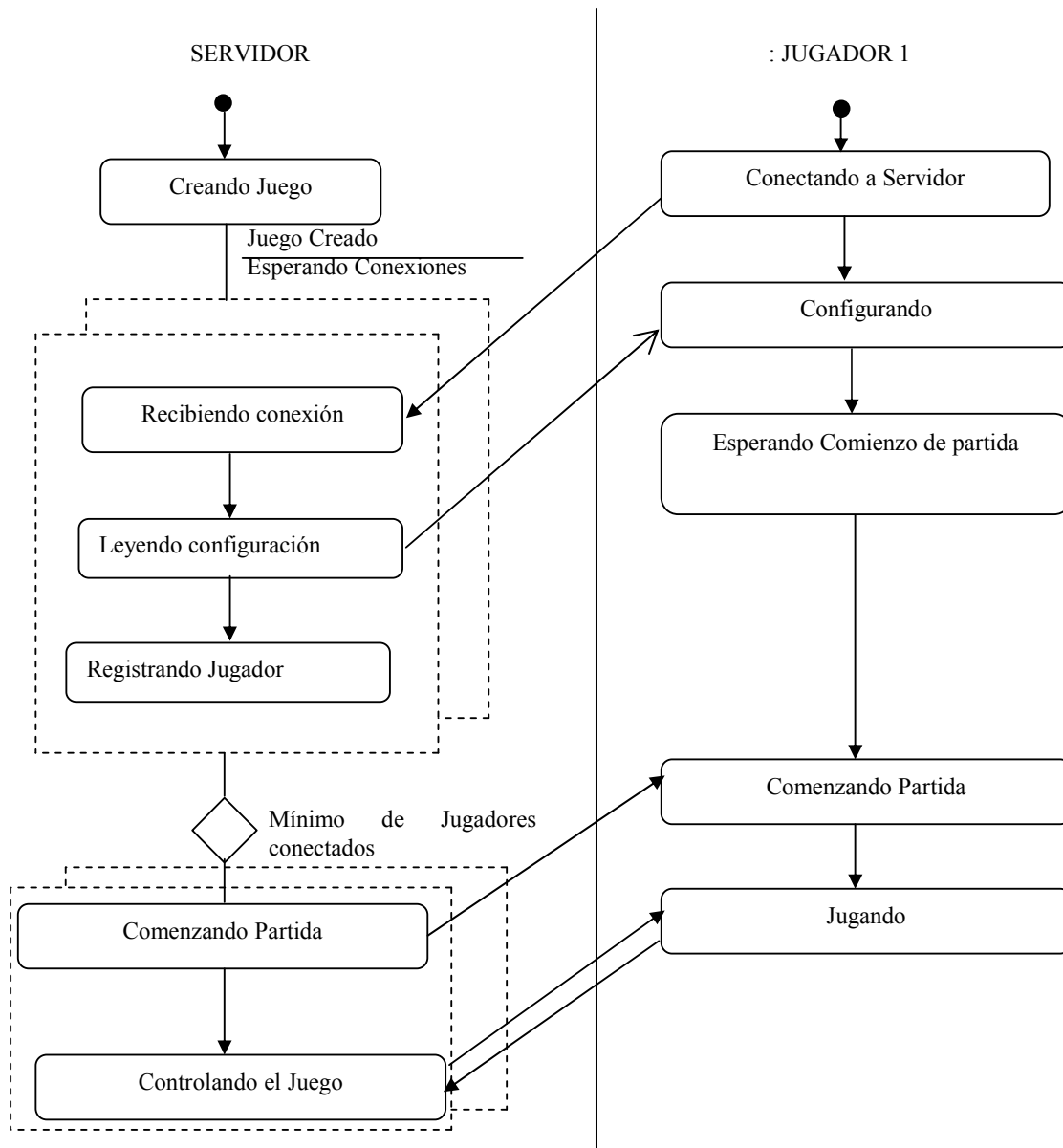


Figura 6: Modelado a través del DEA de un juego online multijugador.

7) Si uno de los estados requiere la utilización de un recurso compartido especificar como se realiza el control de acceso al mismo mediante un semáforo o un monitor.

8) Para rutinas repetidas dentro de un mismo nodo, es decir que disponen de un control de conexión y para cada una de ellas se realiza un proceso igual,

Modelado de una aplicación

Se desea modelar el control de acceso a una bóveda de alta seguridad.

Esta cuenta con diferentes mecanismos para controlar el acceso a la misma.

Como primera medida de seguridad se requiere poseer la llave que habilita un teclado mediante el cual se ingresa una clave de acceso. Existe un número de veces

que se puede ingresar en forma errónea la clave antes de ser disparada la alarma. Una vez dado por válido dicho código se realiza un escaneo de retina para autenticar a la persona.

Luego se habilita un teclado en cada una de las dos sucursales, en donde se ingresa una clave de seguridad. Una vez chequeada la validez del código se procede a realizar un escaneo de retina a quienes ingresan dichas claves. Estos escaneos son procesados mediante los patrones almacenados en el servidor de la casa central. Si son válidos se habilita el acceso a la bóveda.

Modelado por medio de UML

Se realizarán a continuación una serie de diagramas para modelar los aspectos más relevantes de la aplicación:

- 1) Diagrama de despliegue
- 2) Diagrama de clases
- 3) Diagrama de colaboraciones

Se recomienda leer una breve reseña de la construcción de los modelos que se emplearán (ver [11] y [13]).

Diagrama de Despliegue: Mediante el diagrama de despliegue se pueden observar los distintos nodos, el tipo de conexión que existe entre ellos y las operaciones que despliega cada uno de esos nodos (Ver Figura 7).

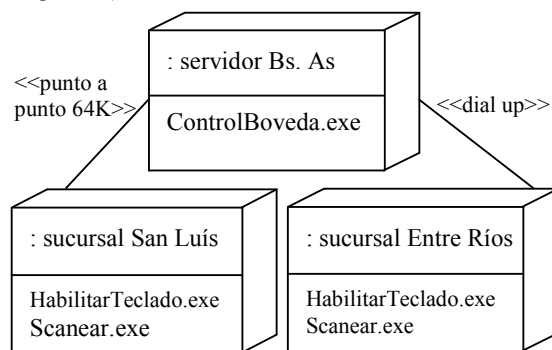


Figura 7: Diagrama de Despliegue

Como se observa en la figura anterior, este diagrama muestra la configuración en tiempo de ejecución de los nodos de procesamiento y los componentes que residen en ellos.

Diagrama de Clases: Este diagrama permite identificar las clases, las clases activas y las operaciones que contienen las mismas (ver Figura 8).

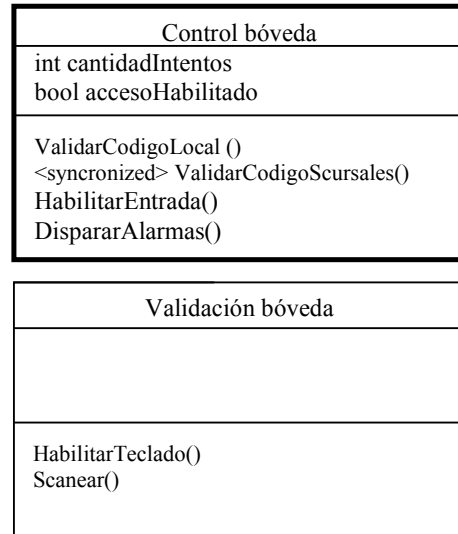


Figura 8: Diagrama de Clases

Una clase activa es una clase cuyos objetos tienen uno o más procesos o hilos de ejecución por lo tanto pueden dar lugar a actividades de control. Es importante destacar que en UML las clases activas se denotan con el contorno grueso.

Como se observa en la figura anterior, este diagrama se utiliza para modelar la vista de diseño estática de un sistema la cual soporta los servicios que el sistema debe proporcionar a los usuarios finales.

Diagrama de Colaboraciones: En este diagrama que resalta la organización estructural de los objetos que envían y reciben señales se muestran las interacciones organizadas alrededor de las instancias y los enlaces de unas con otras.

En este caso fue imprescindible utilizar estereotipos para poder vincular éste diagrama con el de despliegue. Puede observarse que a través del estereotipo Location se logra identificar al nodo en cuestión (ver Figura 9).

Las instancias de las clases activas se diferencian de las instancias de clases no

activas representándolas con el contorno grueso.

utilizando el círculo vacío (usando la nomenclatura habitual del Diagrama de Transición de estados).

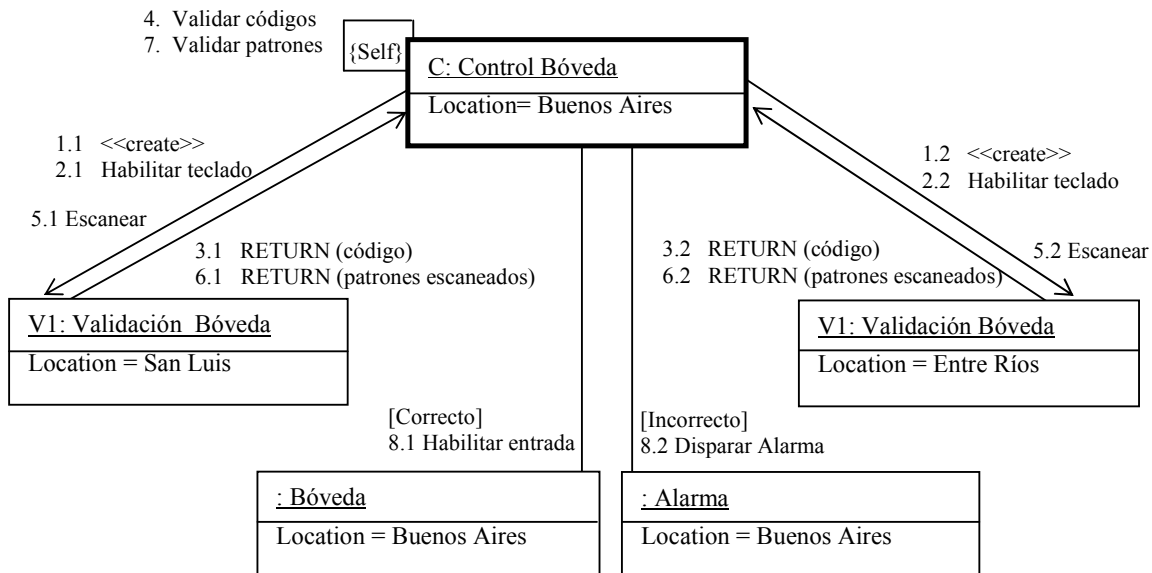


Figura 9: Diagrama de Colaboraciones

Modelado por medio del DEA

Se aplica la metodología de modelado indicada en la Sección 3, para modelar la aplicación descrita, por medio del DEA (Figura 10).

1) Se identifican los distintos nodos: Sucursal de San Luis, Sucursal de Entre Ríos y Casa Central en Buenos Aires. Entonces se consideran tres calles una para cada nodo.

2) Se ubica en el centro del diagrama la calle correspondiente al nodo Casa Central ya que desde allí se inicia el proceso o la funcionalidad que estamos modelando. Esto evitará el cruce de líneas lo que quitaría claridad al diagrama.

3) Se marcan las conexiones entre los nodos por medio de líneas y sobre cada una de ellas se indica el tipo de conexión. Entre Casa Central y la sucursal San Luis la conexión tal como lo mostramos en el diagrama de despliegue es punto a punto y de Casa Central a la sucursal Entre Ríos es Dial Up.

4) Tomando como punto de partida el nodo central, se indica el estado de inicio

5) Se comienza a modelar un diagrama de estados dentro del nodo central y recurrimos a una línea de sincronismo en el momento en que éste nodo se conecta simultáneamente con los nodos remotos.

6) Cuando se realiza una comunicación desde el nodo central hacia los otros y viceversa, usamos la nomenclatura propuesta para indicar los mensajes y llamadas a procedimientos remotos (RPC) sincrónicos ó asincrónicos.

7) Si suponemos que las claves de las sucursales, San Luis y Entre Ríos se chequean contra un archivo que se encuentra en la Casa Central, ese archivo será un recurso compartido por lo cual utilizaremos por ejemplo un semáforo.

8) En esta aplicación cada sucursal ejecuta en Casa Central el control del ingreso de la clave y el chequeo de la cantidad de intentos fracasados. Esta rutina se englobará mediante la representación gráfica sugerida para denotar cardinalidad, a fin de mostrar que ambos nodos la realizarán en forma independiente de lo sucedido con el otro nodo.

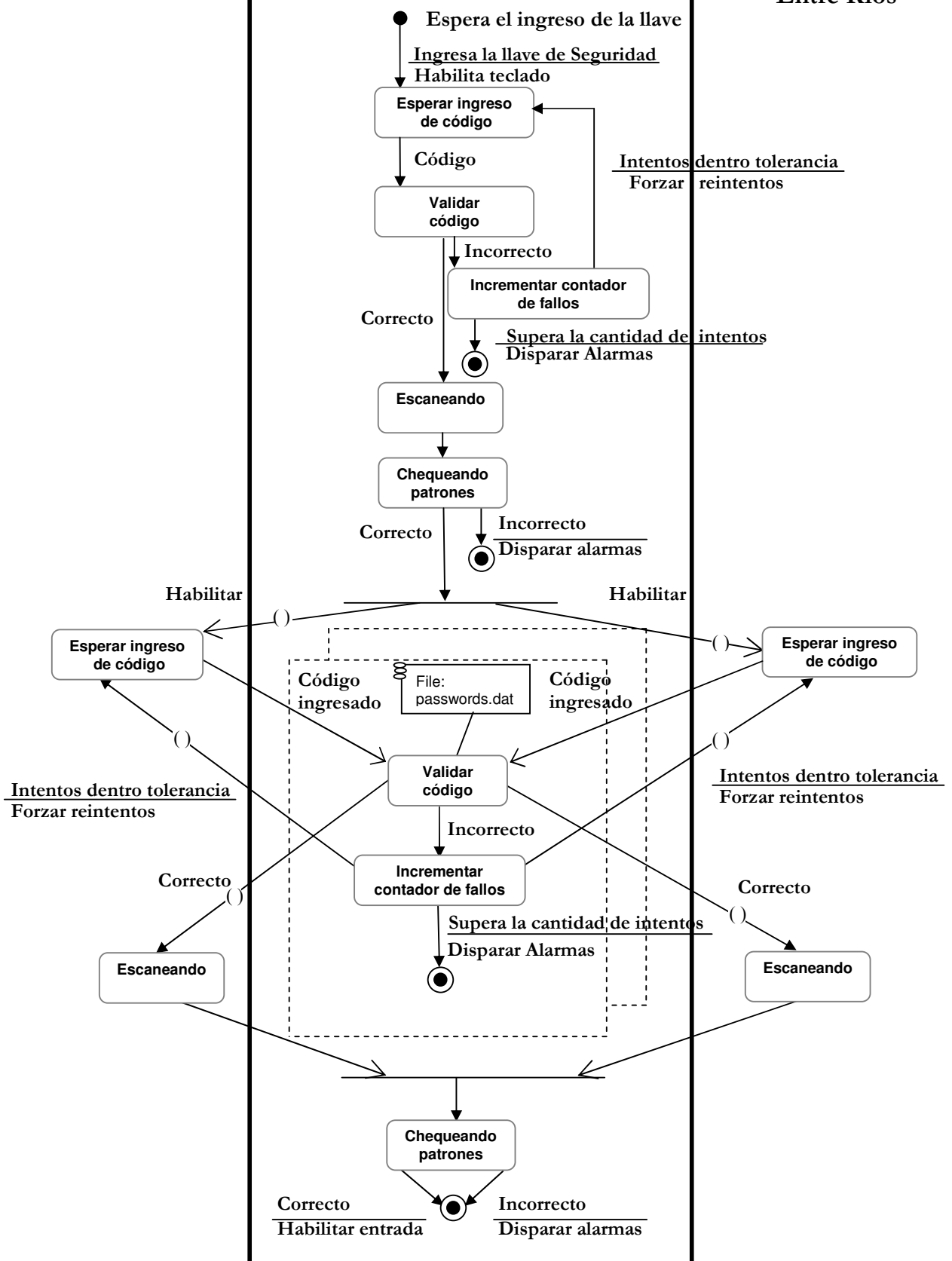


Figura 10: DEA que integra los elementos típicos anteriormente planteados para las aplicaciones con procesos concurrentes y distribuidos.

Comparación con UML

UML brinda herramientas para el modelado de procesos concurrentes mediante el uso de clases activas, dichas clases indican que sus objetos contienen diferentes hilos de ejecución.

Para poder modelar la comunicación entre diferentes procesos y threads UML propone la utilización de diagramas de objetos adornados con varios estereotipos como ser location que nos indica la ubicación física de ese hilo o proceso. También para establecer la ubicación física es posible realizar un diagrama de despliegue que indique que procesos ejecuta cada nodo y cuales son las conexiones físicas entre ellos. Si se quiere modelar como se comunican procesos ubicados en distintos nodos es posible que un diagrama de colaboración donde cada objeto tenga que estar estereotipado con la propiedad location se vuelva muy difícil de seguir. Por ello la propuesta del DEA es utilizar las calles para ver que es lo que ocurre dentro de cada nodo y como se comunican unos con otros.

Para la comunicación entre procesos UML propone la diferenciación entre mensajes sincrónicos y asincrónicos. Esta metodología no especifica una semántica particular para llamadas a procedimientos remotos RPC. Estos últimos indicarían que en un momento dado se invoca un procedimiento ubicado en otro nodo sin que en dicho nodo necesariamente se este ejecutando un proceso que estuviera siendo monitoreado a la espera de mensajes. El párrafo precedente justifica porque resulta necesario diferenciar mensajes de llamadas a procedimientos remotos (RPC).

En cuanto a la utilización de diagramas de estado UML propone su utilización para modelar el comportamiento de cada una de las clases activas por separado. Siendo necesario para ver su interacción construir un diagrama de colaboración.

Para los recursos cuyo acceso debe ser sincronizado, UML propone estereotipar los métodos que por ejemplo deban ser sincronizados dentro de una clase pero no

hace referencia a elementos físicos compartidos como ser un archivo, una unidad de disco, etc.. Elementos que muchas veces es necesario destacar para apreciar como varios hilos van a competir por dicho recurso.

Conclusiones

De la comparativa presentada en el Item 5, puede desprenderse que el DEA es una herramienta que permite modelar aplicaciones con procesos concurrentes y distribuidos, plasmando las características de estas aplicaciones en un único diagrama. Si bien el diagrama parece a simple vista ser más complejo (Figura 10), evita tener que realizar una serie de modelos (ver Figura 7, 8, 9) los cuales traen aparejados los siguientes inconvenientes:

- 1) Tener que relacionar los distintos modelos para lograr llegar a la visión que se observa en el DEA.
- 2) Tener que usar estereotipos para poder modelar características no nombradas en UML.
- 3) El conjunto de modelos realizados en UML no logra mostrar la diferencia entre mensajes y llamadas a procedimientos remotos (RPC). Así como tampoco se pueden visualizar rutinas repetitivas, es decir el concepto de cardinalidad.

Por los motivos expuestos sostenemos que el DEA ayuda al análisis, incorporando las características de los modelos de UML y agregando aquellas características específicas de este tipo de aplicaciones, no soportadas por dichos modelos.

Lo expuesto nos motiva a proponer este modelo para presentarlo a un usuario final ya que consideramos que aporta una integración de las distintas visiones conseguidas a través de varios diagramas de UML.

Consideramos por último que el Diagrama de Estados Activos (DEA) puede transformarse en una herramienta a través de la cual un usuario no experto en desarrollo de sistemas pueda con mínimas explicaciones, comprender y transmitir a

sus pares el funcionamiento de una aplicación que requiere la utilización de procesos concurrentes y distribuidos.

El DEA no solo permite modelar sistemas centralizados sino que se adapta perfectamente para permitir el modelado de sistemas descentralizados donde no exista un nodo central que comande el proceso.

Trabajos Futuros

El próximo paso a seguir con el DEA es el modelado de una aplicación real distribuida que representa el sistema de seguridad de una importante cadena de supermercados a nivel Nacional. Mostrando como interactúan las distintas sucursales, los servidores regionales y el servidor central.

Seguidamente y con la colaboración de docentes de la materia Sistemas Operativos de la Universidad se va a analizar la posibilidad de utilizar el DEA como base de modelado de un Sistema Operativo completo, cuyo desarrollo se esta llevando a cabo por alumnos de la Universidad Nacional de La Matanza.

Referencias

Libros

- [1] Booch G, Rumbaugh J y Jacobson I. El lenguaje unificado de modelado. Addison Wesley, pp: 392-394, 2003.
- [2] Fowler M. UML Distilled. Addison Wesley, Third Edition, Pearson Education , pp: 1-16, 2004.
- [3] Kendall S. Fast Track UML 2.0, Apress, California 2004.
- [4] Stalling W. Operating Systems Internals and Design Principles. Third Edition, Prentice Hall., pp: 208-230, 1998.

E-Book

- [5] OMG Unified Modeling Language Specification. Versión 1.5 Marzo 2003.

Páginas Web

- [6] <http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/multiple-html/>
- [7] <http://www.creangel.com/uml/actividad.php>
- [8] <http://ingenieria.unlam.edu.ar/contenidos/investigaciones/wisbd022.pdf>
- [9] <http://www.investigamos.com.ar>
- [10] <http://www.investigamos.com.ar/CONGRESOS.htm>

- [11] <http://www.microsoft.com/spanish/msdn/articulos/archivo/230801/voices/modelsoftware.asp>

- [12] <http://www.uml.org/#UML2.0>

- [13] <http://www-gris.det.uvigo.es/~avilas/UML/node22.html>

- [14] <http://www-gris.det.uvigo.es/~avilas/UML/node46.html>