Modelado de Aplicaciones con Procesos Concurrentes y Distribuidos

Daniel A. Giulianelli, Rocío A. Rodríguez, Pablo M. Vera

Departamento de Ingeniería e Investigaciones Tecnológicas
Universidad Nacional de La Matanza
Florencio Varela 1903 - (1754) San Justo - Buenos Aires
Argentina
{dgiulian, rrodri, pablovera}@unlam.edu.ar

Resumen

En este paper se muestra la posibilidad de modelar una aplicación para redes de alta velocidad a través de UML (Lenguaje de Modelado Unificado). Tomando como punto de partida la consideración que toda aplicación de este tipo tendrá procesos remotos que deben ejecutarse en forma concurrente y en tiempo real.

Para modelar una aplicación como la descripta en el párrafo anterior, se evaluaron los distintos modelos de UML 2.0, viendo que ninguno de ellos se adaptaba por completo a las necesidades presentadas. Por ello la propuesta expuesta en este paper es crear un modelo basado en los ya existentes.

Palabras Claves

Ingeniería, Software, Modelado, DTE, Diagramas, Actividades, Estados, Transición, UML, Internet, Redes, Procesos, Distribuidos, Concurrentes, Ejecución, Tiempo, Real, Procesamiento, Paralelo.

Destinado para

II Workshop de Ingeniería de Software y Bases de Datos

1. Introducción

1.1 Lenguaje de Modelado Unificado (UML)

"El Lenguaje unificado de modelado, es un lenguaje para la especificación, visualización, construcción y documentación de componentes de sistemas de software, como así también modelado de negocios y de sistemas no software. UML[1] representa una colección de las mejores prácticas de la ingeniería que han sido probadas satisfactoriamente en el modelado de sistemas grandes y complejos."

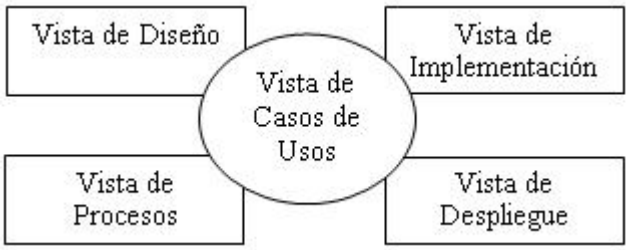
A mediados de los 70 y fines de los 80 aparecieron los lenguajes de modelado y lenguajes de programación orientados a objetos. Entre los años 1989 y 1994 surgieron muchos métodos orientados a objetos. Entre estos se destacaban los de Booch, Jacobson y Rumbaugh entre otros como Shlaer-Mellor, Coad-Yourdon. Cada uno de esos métodos tenía puntos fuertes y débiles pero todos iban evolucionado y adquiriendo cada vez más características similares.

En octubre de 1994 cuando Rumbaugh comenzó a trabajar junto a Booch en la firma Rational, se centraron en la unificación de sus dos modelos, un año más tarde se unió al proyecto Jacobson con los aportes de su propio modelo. En el año 1996 se invitó a participar a la comunidad de ingeniería de software por lo que se formo el denominado "consorcio UML" formado por varias organizaciones que pusieron sus recursos y esfuerzos para lograr una especificación completa de UML. Entre las organizaciones que contribuyeron a lograr la especificación 1.0 de UML se encuentran Digital Equipment Corporation, Hewlett-Packard, Ilogix, Intellicorp, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle, Rational, Texas Instruments y Unisys. Esta especificación se presentó a la OMG[2] en enero de 1997 para su estandarización. En esta época se incorporaron al grupo Andersen Consulting, Ericsson, Objectime Limited, Platinum technology, Ptech, Reich Technologies, Softeam, Sterling Software y Taskon.

UML actualmente está conformado por 13 diagramas y un lenguaje de especificación de restricciones denominado OCL (Object constrain Language).

1.1.1 Vistas de UML

UML consta de 5 vistas, que se ilustran a continuación:



[1] UML Unified Modeling Language Specification, Version 1.5
[2] La OMG (Object Management Group) es un consorcio internacional sin fines de lucro que establece y mantiene estándares para la industria de la computación.

En cada una de estas vistas se presta especial atención en distintos aspectos:

- ***Vista de Casos de Usos:*** Se utiliza para determinar los requisitos funcionales del sistema, define el comportamiento del mismo ¿Qué es lo que debe hacer? no ¿Cómo se hace?
- ***Vista de Diseño:*** Se hace hincapié "en el ¿Cómo?" atendiendo a las funcionalidades que deberá tener el sistema. En esta vista se diseñan las clases e interfases.
- ***Vista de Procesos:*** Se presta esencialmente atención a la concurrencia, hilos, rendimiento y funcionamiento.
- ***Vista de Implementación:*** Componentes, Archivos necesarios (por ejemplo DLL) y Gestión de Versiones (Evolución de una versión a otra).
- ***Vista de Despliegue:*** Tecnología de Hardware, Distribución, Entrega e Instalación.

1.1.2 Diagramas de UML 2.0

UML 2.0 presentado en Octubre del 2004 consta de 13 diagramas:

Diagramas Estructurales:

- Diagrama de Clases: Características de las clases y relaciones.
- Diagrama de Componentes: Estructura y conexión entre componentes
- Diagrama de Despliegue: Despliegue de componentes en los nodos
- Diagrama de Estructuras Compuestas: Descomposición en tiempo de ejecución de una clase.
- Diagrama de Objetos: Ejemplo de configuraciones de instancias
- Diagrama de Paquetes: Estructura jerárquica en tiempo de compilación

Diagramas de Comportamientos:

- ***Diagrama de Actividades*:** Comportamiento procedural y paralelo.
- Diagrama de Casos de Uso: Interacción de los usuarios con el sistema.
- ***Diagrama de Máquina de Estados*:** Como un evento cambia un objeto a lo largo de su tiempo de vida[3].
- Diagramas de Interacción:
    - Diagrama de Secuencia: Interacción entre objetos, haciendo énfasis en la secuencia.

[3] **Este tipo de diagrama puede tener distinto nivel de complejidad, desde una máquina de estados completa (autómata finito) hasta el modelo más sencillo "Diagrama de Transición de estados".**

- Diagrama de Comunicaciones: Interacción entre objetos haciendo énfasis en las vinculaciones entre los mismos.
- Diagrama de Vista General de Interacción: Mezcla del diagrama de secuencia y el de actividades.
- Diagrama Temporal: Interacción entre los objetos haciendo énfasis en el tiempo.

1.2 Diagrama de Transición de Estados (DTE)

Un diagrama de transición de estados se puede utilizar para describir el comportamiento de instancias de un elemento del modelo como por ejemplo un objeto o una interacción.
Específicamente describe las posibles secuencias de estados y acciones a través de las cuales las instancias del elemento pueden atravesar durante su tiempo de vida como resultado de reacciones ante eventos discretos (por ejemplo señales o invocaciones de operaciones).

1.2.1 Componentes de un DTE:

- Estados (dentro de los estados se destacan como estados especiales el estado de Inicio y el estado de Fin).
- Transiciones:
  - Condiciones.
  - Acciones.
  - Eventos.

1.2.1.1 Componentes de un estado

- Los estados pueden tener un nombre o ser anónimo
- Acciones: Pueden ser de entrada, salidas ó internas.
- Transiciones Internas: Son transiciones en las que no se cambian de estado.
- Subestados: Son estados animados dentro de otros. Es decir que los estados complejos pueden ser descompuestos por medio de subestados.
- Eventos diferidos: Se agregan a una cola y se los realiza en otro estado.

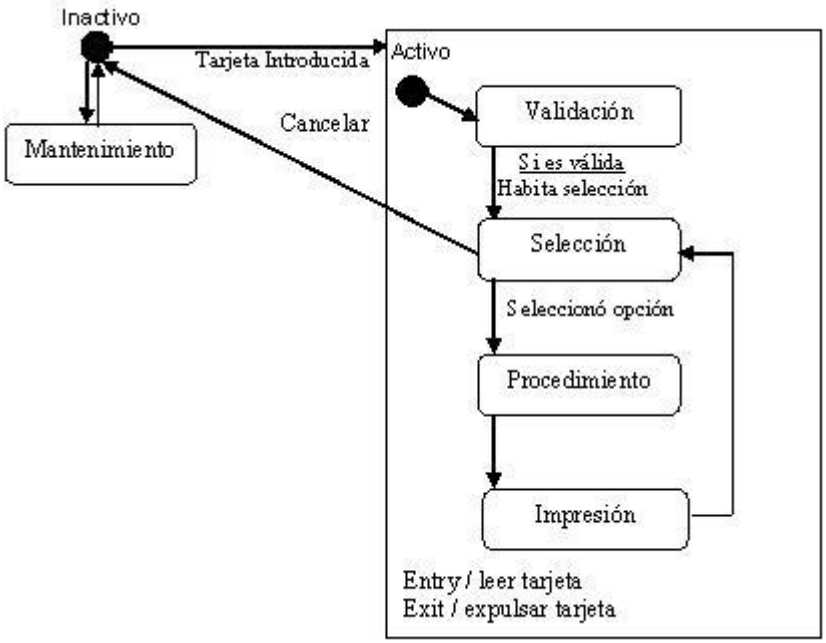
1.2.1.1.1 Acciones, Transiciones Internas y Eventos Diferidos

En el siguiente ejemplo genérico se hace alusión a un estado denominado “RASTREANDO”. En este estado se rastrea un objetivo y se definen en el las: acciones, acciones de entrada y acciones de salida, Transiciones Internas, Eventos Diferidos, que ocurren en dicho estado.

- Las **acciones de entrada (entry)** tienen lugar inmediatamente cuando se pasa al estado mencionado.
- Las **acciones de salida (exit)** son las últimas acciones que se ejecutan antes de abandonar el estado en el que estamos, se ejecutan solo si el paso de estado es efectivo o sea si se cumple la condición de guarda necesaria para pasar al estado siguiente.
- Las **acciones internas (do)** especifican acciones que se realizan en dicho estado.
- Los **eventos diferidos (defer)** son eventos que se producen pero que no causan una acción en el estado actual sino que serán atendidos en estados posteriores.
- Las **transiciones internas** son cambios que se producen internamente que no llegan a generar un cambio de estado en sí mismas sino que denotan una parte del mismo estado.

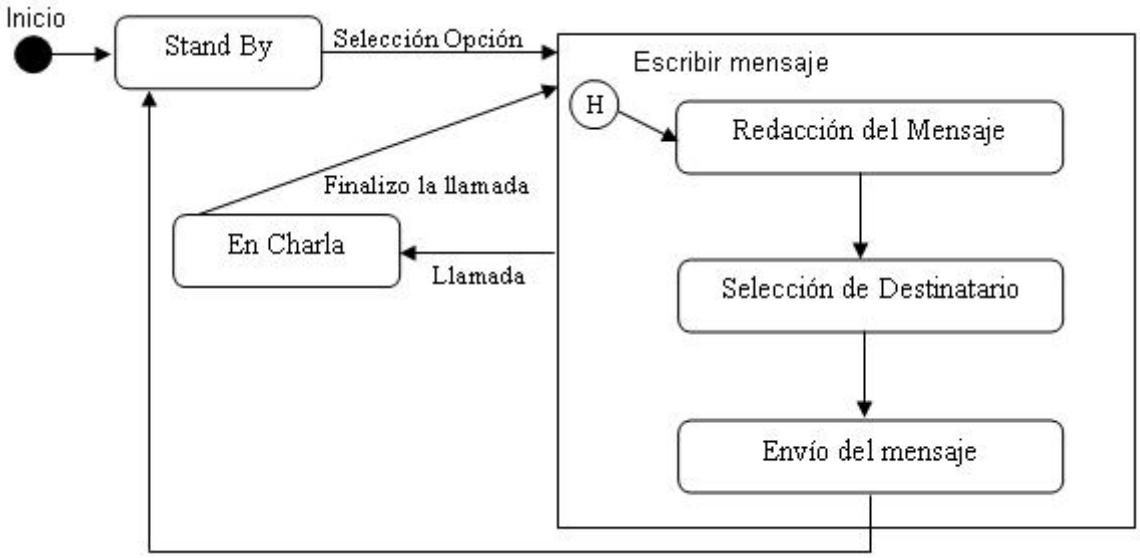
1.2.1.1.2 Subestados

Ejemplo 1: Cajero Automático (los subestados detallados forman parte del estado "Activo").



Ejemplo 2: Teléfono celular con posibilidad de enviar mensajes.

ESTADOS DE HISTORIA: Si mientras se está escribiendo un mensaje de texto, entrara una llamada, una vez finalizada la misma, se recupera el estado del mensaje que se estaba escribiendo.



1.3 Diagrama de Actividades

1.3.1 Características del diagrama de actividades

Un diagrama de actividades es un tipo especial de diagrama de estados en el cual casi todos los estados son estados acción (identifican una acción que se ejecuta al estar en él) y casi todas las transiciones evolucionan al término de dicha acción (ejecutada en el estado anterior).

Las flechas dirigidas entre estados de acción representan transiciones con evento implícito que, en el caso de decisiones, pueden tener una condición o guarda asociada (que al igual que en los diagramas de estado evalúa a *true* o a *false*).

En un diagrama de actividades también pueden existir barras de sincronización (*synchronization bar)*, a las que se encuentran asociadas varios caminos salientes. Cada camino saliente se dirige a una actividad, realizándose dichas actividades en paralelo.

Dado que el diagrama de actividades permite expresar el orden en que se realizan las cosas, resulta adecuado para el modelado de organizaciones (business modeling) y el de programas concurrentes (permite representar gráficamente los hilos de ejecución).

Su principal aportación al modelado del comportamiento es que soportan el comportamiento paralelo, lo que resulta adecuado para el modelado de flujo de trabajo (workflow) y programación multihilo (multi-thread).

1.3.2. Componentes de un Diagrama de Actividades:

- Estados de actividad
- Estados de acción
- Transiciones
- Bifurcaciones
- División y Unión
- Calles

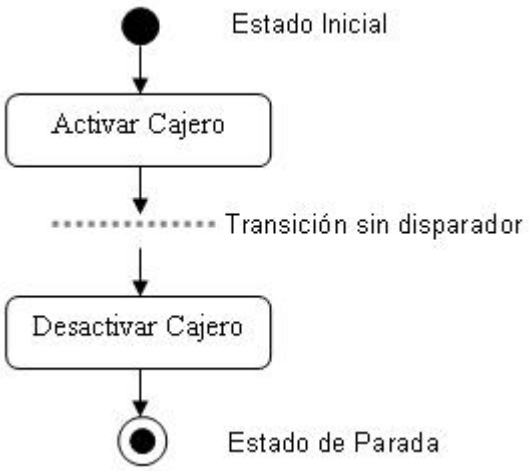
1.3.2.1. Estados de actividad y estados de acción

La representación de ambos es un rectángulo con las puntas redondeadas, en cuyo interior se representa bien una actividad o bien una acción. La forma de expresar tanto una actividad como una acción, no queda impuesta por UML, se podría utilizar lenguaje natural, una especificación formal de expresiones, un metalenguaje, etc. La idea central es la siguiente: "Un estado que represente una acción es atómico, lo que significa que su ejecución se puede considerar instantánea y no puede ser interrumpida"

En cambio un estado de actividad, sí puede descomponerse en más sub-actividades representadas a través de otros diagramas de actividades. Además estos estados sí pueden ser interrumpidos y tardan un cierto tiempo en completarse. En los estados de actividad podemos encontrar otros elementos adicionales como son: acciones de entrada (entry) y de salida (exit) del estado en cuestión, así como definición de submáquinas.

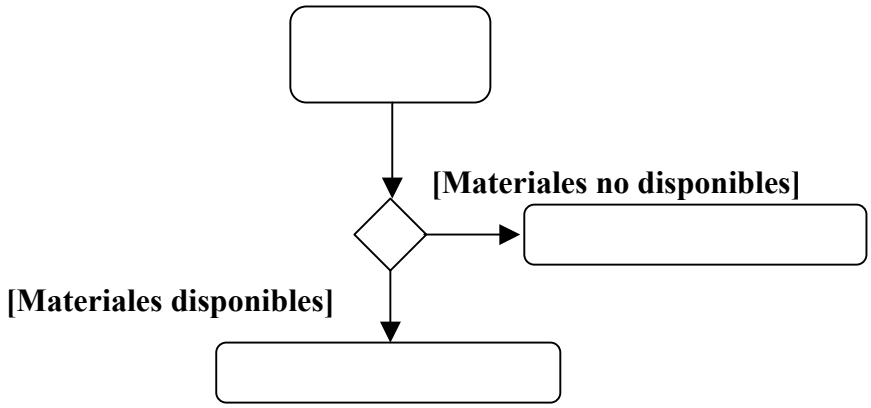
1.3.3 Transiciones

Las transiciones reflejan el paso de un estado a otro, bien sea de actividad o de acción. Esta transición se produce como resultado de la finalización del estado del que parte el arco dirigido que marca la transición. Como todo flujo de control debe empezar y terminar en algún momento, podemos indicar esto utilizando dos disparadores de inicio y fin.



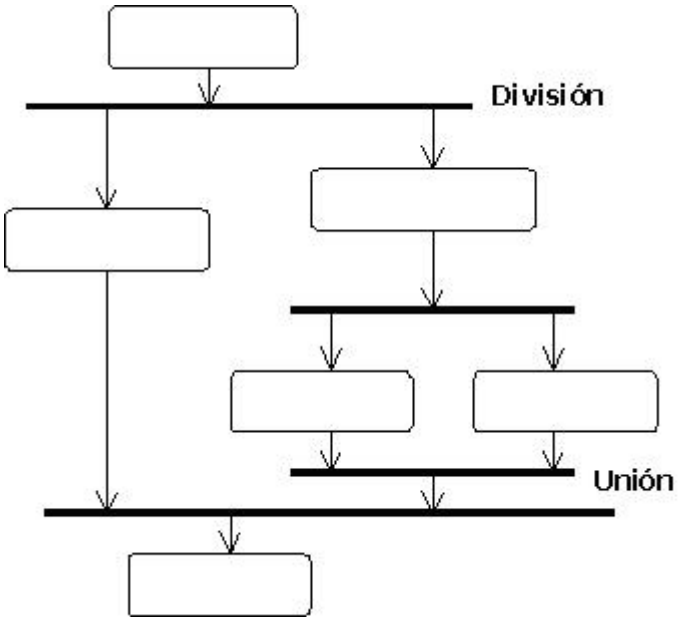
1.3.4 Bifurcaciones

Un flujo de control no tiene porqué ser siempre secuencial, puede presentar caminos alternativos. Para poder representar dichos caminos alternativos o bifurcación se utilizará como símbolo el rombo. Dicha bifurcación tendrá una transición de entrada y dos o más de salida. En cada transición de salida se colocará una expresión booleana que será evaluada al llegar a la bifurcación, las condiciones de guarda de la bifurcación han de ser excluyentes y contemplar todos los casos ya que de otro modo la ejecución del flujo de control quedaría interrumpida. Para poder cubrir todas las posibilidades se puede utilizar la palabra ELSE, para indicar una transición obligada a un determinado estado cuando el resto de guardas han fallado.



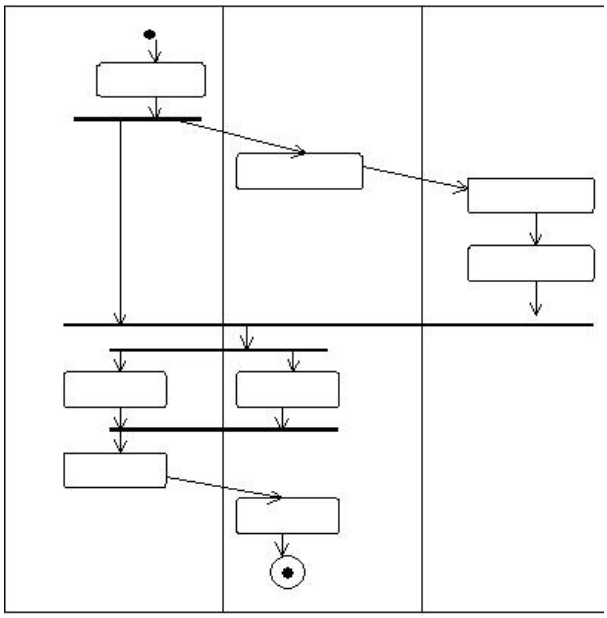
1.3.5 División y unión

No sólo existe el flujo secuencial y la bifurcación, también hay algunos casos en los que se requieren tareas concurrentes. UML representa gráficamente el proceso de división, que representa la concurrencia, y el momento de la unión de nuevo al flujo de control secuencial, a través de una línea de sincronismo (que se representa mediante una línea horizontal ancha).



1.3.6 Calles

Cuando se modelan flujos de trabajo de organizaciones, es especialmente útil dividir los estados de actividades en grupos, cada grupo tiene un nombre concreto y se denominan calles. Cada calle representa a la parte de la organización responsable de las actividades que aparecen en esa calle.



2. Propuesta - (DEA)

Tomando las particularidades del Diagrama de Transición de Estados (DTE) y las correspondientes al Diagrama de Actividades e incorporando la propuesta del grupo de trabajo, se obtiene lo que hemos llamado **Diagrama de Estados Activos** (DEA).

2.1. Metodología de modelado

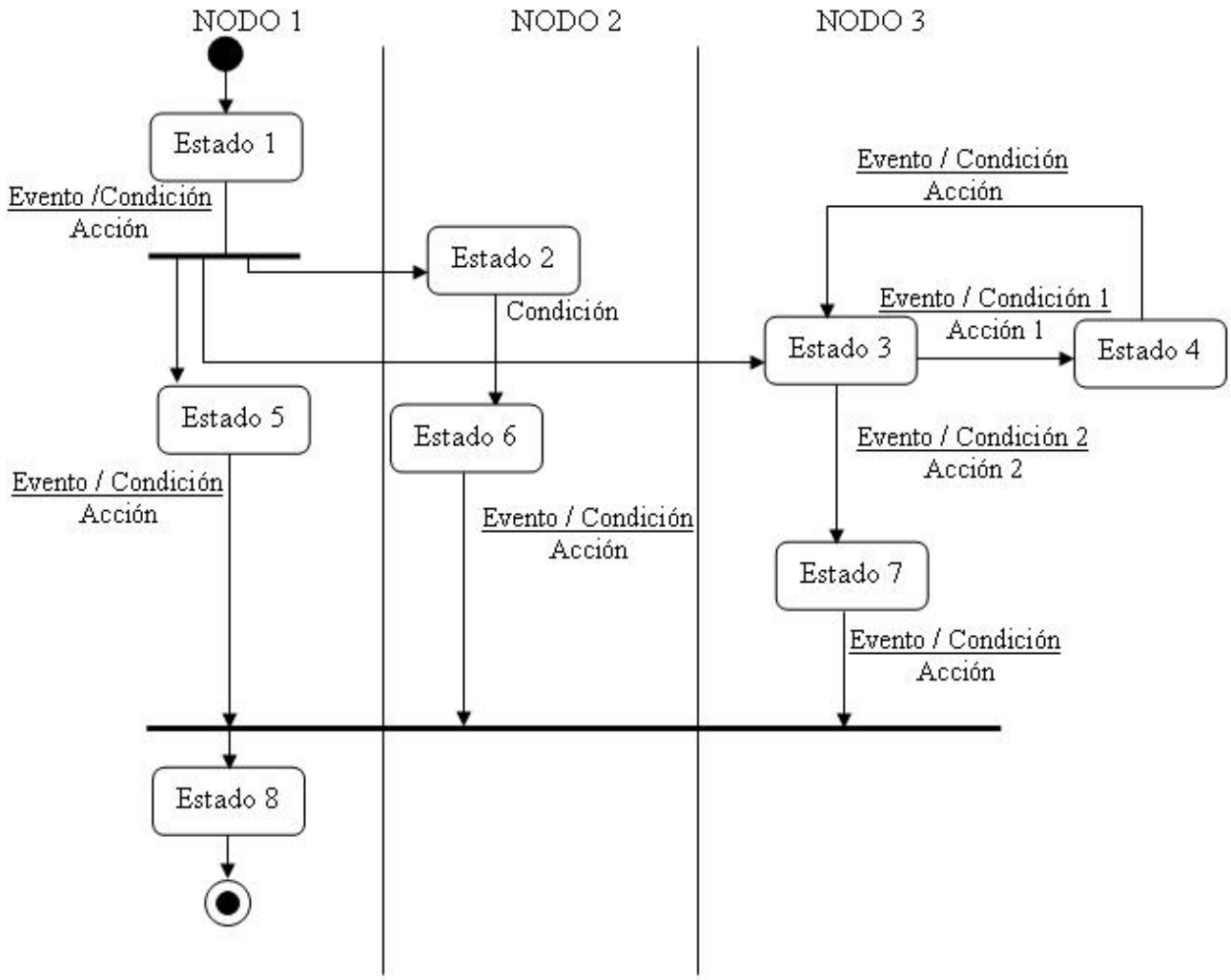
- **Para representar un proceso distribuido**: Se utilizan las calles que caracterizan al clásico Diagrama de actividades, para determinar en que nodo se realizará la ejecución de ciertos procesos.

- **Para representar la concurrencia de procesos**: Se utilizan las líneas de sincronismo del Diagrama de actividades.

- **Para detallar la información de los estados**: Además del nombre del estado se puede detallar como en el clásico DTE acciones de entrada y salidas, transiciones internas y eventos diferidos.

- **Para el cambio de estados**: Se utiliza las transiciones del DTE indicando la o las condiciones así como el evento y en el caso que las halla las acciones que permiten el paso de estado.

A modo de ejemplo se presenta el modelo propuesto:



En donde los nodos intervinientes podrían ser: NODO 1: Mainframe de Argentina - NODO 2: Mainframe de Brasil - NODO 3: Mainframe de Cuba

En el NODO 1 se inicia el proceso, luego del Estado 1 el procesamiento se **distribuye** en dos nodos más, los cuales realizan procesos en forma **independiente**. El NODO 1, queda esperando el procesamiento de los otros nodos para con los resultados enviados por los mismos realizar alguna acción y finalizar el procesamiento de la aplicación.

2.2. Particularidades de la propuesta

Del análisis del modelo presentado surgen las siguientes conclusiones:

- **Es evidente que el paso de un estado a otro está asociado a una serie de eventos y/o acciones que provocan dicho cambio de estado.**

- **Por otra parte estando dentro en un estado podrá ser posible pasar a distintos estados destino según se cumpla alguna de varias condiciones planteadas.**
- **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.**

A continuación se presentan dos ejemplos de acciones que dependen del cumplimiento de cierta condición:

1. Condición: Si la base de datos que se está utilizando en el NODO 3 superó cierto tamaño.
   Acción: Hacer un backup de la misma en el NODO 1.

2. Condición: Si es domingo por la noche hacer el backup semanal de la base de datos del NODO 3.
   Acción 1: Hacer un backup de la misma en el NODO 1
   Acción 2: Hacer un backup de la misma en el NODO 2

Las acciones y condiciones asociadas permiten clarificar el paso de un estado a otro, mientras que las líneas de sincronismo permiten establecer la obligatoriedad de respuesta de todos los procesos que se están llevando a cabo paralelamente. Por ejemplo en el caso presentado anteriormente hasta que no se obtiene una respuesta de que se ha creado la copia de seguridad en el NODO 1 y en el NODO 2, no se continúa en el NODO 3 accediendo a dicha base de datos. Es decir que el NODO 3 ha quedado esperando, para continuar su proceso, la respuesta de lo sucedido en el resto de los nodos (uniéndose el flujo en el NODO 3).

La propuesta que se presenta es el resultado de trabajos de modelado de aplicaciones para Redes de Alta Velocidad, en los que a nuestro criterio faltaba una herramienta que nos permitiera visualizar gráficamente estas particularidades.

3. Conclusión

UML ha ido evolucionando y en UML 2.0 se presentan nuevos diagramas de los cuales uno de ellos "Diagrama de Vista General de Interacción" surge por asociación de dos ya existentes anteriormente.

Hemos conseguido combinar los elementos del Diagrama de Transición de Estados con los del Diagrama de Actividades, que por ser esté último un caso especial del primero ambos guardan cierta relación. De esta asociación surge un nuevo modelo el Diagrama de Estados Activos (DEA) que satisface las expectativas planteadas, pudiendo modelar con el una aplicación de procesamiento distribuido y concurrente con distintos hilos de ejecución.

4. Referencias

4.1. Libros

- "Fast Track UML 2.0". Kendall Scott. Editorial Apress, Fecha de Edición 2004
- "The unified Modeling Language user Guide". Ed Addison Wesley 2000. Grady Booch, James Rumbaugh, Ivar Jacobson. Rational Software Corporation.
- "UML Distilled Third Edition". Martin Fowler. Editorial Addison Wesley, Fecha de Edición 2004.
- "Software process: A roadmap". In The future of software Engineering . Fuggetta, A. Ed. A. Finkelstein, pp 27-34. ACM Press, 2000.

4.2. Documentos

- Ebook "Applying UML and Patterns", Craig Larman
- "OMG Unified Modeling Language Specification". Versión 1.5 Marzo 2003
- "Extending UML with Aspects: Aspects Support in the Design Phase. 3er Aspect-Oriented Programming (AOP). Workshop at ECOOP ´99. Junichi Suzuki, Yoshikazu Yamamoto.
- From AOP to UML – A Bottom-Up Approach. Software Engineering Laboratory, Switzerland. June, 2001. Mohamed M. Kande. Jorg Kienzle. Alfred Strohmeir.
- Programación Orientada a Aspectos: Metodología y Evaluación. IX Congreso Argentino de Ciencias de la Computación CACIC, Octubre 2003. Fernando Asteasuain, Bernardo Contreras, Elsa Estévez, Pablo. R. Fillottrani.
- Getting Started with Aspect. Communication of the ACM (CACM). Vol. 44 Nro. 10, Octubre 2001. Gregor Kickzales, Eric Hilsdale, Jim Hugunin, Mike Kersten, Jefrey Palm, William G. Grisnold.
- ISO/IEC. International Standard: Information Technology. Software engineering – Product quality – Part 1: Quality model, ISO/IEC Standard 9126-1:2001, ISO/IEC, 2001.

4.3. Sitios Web

- http://agilemanifesto.org/2001.
- http://agilemanifesto.org/principles.htm/2001
- http://www.clikear.com/manuales/uml/modelodinamico.asp
- http://www.creangel.com/uml/actividad.php
- http://www.microsoft.com/msdn/articulos/archivo/230801/voices/modelsoftware.asp
- http://www.uml.org
- http://www-gris.det.uvigo.es/~avilas/UML/node22.html
- http://www-gris.det.uvigo.es/~avilas/UML/node46.html