Condiciones de Tiempo en las Aplicaciones Web

Lic. Ariel Gonzalez
Depto. de Computación – Universidad Nacional de Río Cuarto – Córdoba – Argentina
gonzalezg@exa.unrc.edu.ar
Tel: (054)-(0358) – 4676235

Resumen

Las aplicaciones Web tienen características especiales que hacen que los mecanismos de ingeniería[6][7] empleados sean diferentes de las aplicaciones estándares. Esto se debe a que los sitios Web han evolucionado notoriamente respecto a su complejidad, se pueden observar desde centros de información de solo lectura, hasta aplicaciones distribuidas basadas en Web. En este trabajo describimos un método para el diseño de aplicaciones Web[5], el cual es basado en el Lenguaje de Modelado Unificado (UML)[1][9]. Los Diagramas de Estado son usados para modelar posibles cambios de estados de una entidad al Navegar por el sitio y ejecutando acciones dentro del mismo. Los diagramas de Casos de Usos se pueden adaptar para reflejar, entre otras cosas, que usuarios pueden disparar una funcionalidad determinada, especificar que vista tiene cada uno de ellos, etc. Lo mas importante en esta prouiesta son las extensiones temporales que se añaden a los diagramas mencionados para reflejar así las duraciones y momentos en que una acción puede llevarse a cabo. La metodología de temporización utilizada se basa en relojes virtuales, los cuales son asociados mediante expresiones lógicas y aritméticas a los eventos de la aplicación.

Palabras Claves:
UML (Lenguaje de Modelado Unificado), Aplicaciones Web, Diseño Web.
DEERT (Diagrama de Evolución de Estados con Restricciones Temporales)

1. Introducción

El método que se presenta está basado en un marco de trabajo W2000 [2], éste describe una serie de actividades que se deben llevar a cabo en el diseño de aplicaciones Web [8].

Las aplicaciones Web son diferentes de los documentos hypermedia tradicionales por tres aspectos principales:

- Los usuarios no solo navegan sino también activan operaciones y transacciones,
- La estructura hypermedia puede evolucionar cuando la aplicación (sus operaciones) evoluciona.
- Diferentes usuarios pueden tener diferente visibilidad de la información y diferentes capacidades para ejecutar operaciones.

Estos tres enfoques, por supuesto no son los únicos pero son los mas importantes, la combinación del aspecto operacional y el de navegación conducen a las aplicaciones Web. La sofisticada evolución y dinamismo hacen que las operaciones afecten el estado de su contenido y navegación del sitio, es decir, no solo modifican el contenido de páginas individuales sino también agregan o eliminan grupos de páginas y sus enlaces. De las distintas aplicaciones Web surgen una variedad potencialmente grande de diferentes usuarios (roles) con diferencias en los requisitos de navegación y funcionales. Es por ello que se pueden observar una cierta categorización en los usuarios de la aplicación, con lo cual surge la necesidad de especificar de manera clara cuales son los requerimientos funcionales y de navegación asociados a cada uno.

Para describir cada una de las actividades en el diseño, se hará mediante la aplicación de un ejemplo real.

2. Breve Descripción del Ejemplo a utilizar

Una Plataforma Web para el desarrollo de actividades en lo que respecta a Educación a Distancia, debe ser un medio de comunicación e interacción entre los alumnos inscriptos a un curso y sus tutores durante el progreso del curso. Los medios usados dentro de la plataforma para la comunicación, entre otros son: *el pizarrón de curso*, *Cuadro de Novedades*, *Foro de Discusión*,

Salas de Conversación, *Sección de Materiales*, *Estadística de Accesos, de Participación con el e-mail, etc.* .Los usuarios (roles) involucrados en un curso son: *los usuarios generales (un Navegador), alumnos, tutores, veedores, monitores técnicos, etc.*. Además también existe la figura de un usuario *Administrador* del Sitio, el cual, y a través del mismo sitio, puede realizar diversas actividades como por ej. Dar de Alta un Curso.

Cuando se establece el desarrollo de un curso, el *Administrador* da de alta en la Plataforma dicho curso dejándolo en estado "*En Difusión"*. En este primer período el curso se encuentra en difusión y publicación, luego de una cierta cantidad de tiempo el curso debe cambiar a estado "*En Inscripción"*. Este permanecerá en dicho estado durante un cierto lapso de tiempo y los interesados en participar tiene la posibilidad de inscribirse mediante un formulario de registro. Al finalizar el período de inscripción el curso cambiará de estado pasando a "*En ejecución"* con el objetivo de que los integrantes (*tutores, alumnos inscriptos, etc.)* puedan ingresar a la Plataforma con sus respectivas contraseñas y comiencen a interactuar para llevar a cabo sus objetivos. EL *Tutor* del curso es el encargado entre otras cosas de: *Administrar al Pizarrón del Curso, Colgar los Materiales de Lectura, Crear Asuntos en el foro para discusión de temas, Abrir un canal de Chat para la conversación con los alumnos,* etc. . Los *Alumnos* pueden *Descargar Materiales* para su lectura, *Ingresar Opiniones al Foro* en un Asunto que se encuentre *Abierto, Ingresar a un canal abierto de Chat, enviar email,* entre otras actividades. Estas son algunas de las tareas que un usuario de la plataforma necesita para que Tutores y Alumnos interactúen de manera organizada.

El sistema completo tiene un alto grado de complejidad, pero nuestro objetivo es utilizar algunos requisitos de esta aplicación para presentar los aspectos importantes en el diseño de una aplicación Web utilizando UML y extendiéndolo en algunos casos con restricciones temporales.

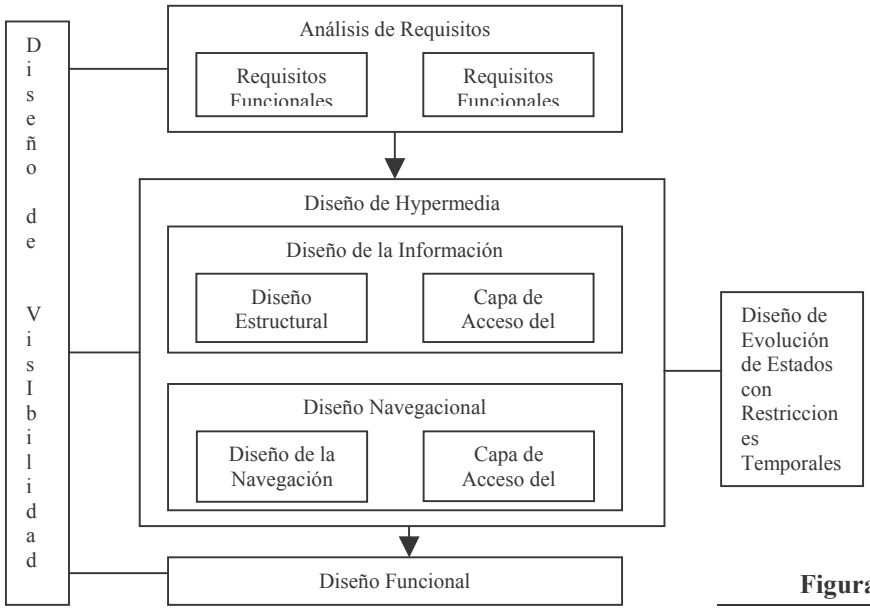

Figura 1.

3. W2000 Framework

W2000 organiza el diseño en actividades tal como los muestra la **Figura 1**. Cada actividad produce un modelo (un conjunto de diagramas relacionados entre si), el cual describe algunos aspectos de la aplicación. No define un nuevo proceso de diseño, sino que identifica cuales son las tareas en el diseño y muestra como se relacionan mediante diagramas del UML. Como se puede ver en el gráfico, hay una gran cantidad de actividades que se pueden desarrollar en paralelo, y los diseñadores pueden necesitar rehacer ciertas actividades en cualquier tiempo modificando o ampliándolas. Por detalles de estas actividades puede ver en [2].

3.1. Propuesta: Diseño de Evolución de Estados con Restricciones Temporales

Cada vez es más común encontrar sistemas en los cuales el correcto funcionamiento depende no sólo de la sucesión de acciones que realizan sino también el momento en que las mismas se levan a cabo. Estos sistemas se caracterizan por la presencia de las restricciones de tiempo sobre el comportamiento, y su correctitud no solo depende de su comportamiento funcional sino también de la habilidad para introducir las restricciones de tiempo. En el ámbito de las aplicaciones Web existen sistemas en que sus procesos funcionales deben sincronizarse para su lograr sus objetivos.

Los nuevos Diagramas de Evolución de Estados con Restricciones Temporales (**DEERT**) que se proponen son un punto clave para reflejar como y cuando deben dispararse ciertos eventos. Dichos eventos de un diagrama pueden pertenecer a un único requisito funcional o a varios.

Estos diagramas de cambio de estado con restricciones temporales deben proveer a los desarrolladores del sistema, una visión general de cómo deben sincronizarse ciertas funcionalidades de la aplicación y lograr así una mayor automatización y control del sistema. Esta actividad es indispensable cuando se trata de aplicaciones complejas. El objetivo de esta extensión es que el diseñador provea a los desarrolladores una visión gráfica y precisa de las tareas (incluyendo sus tiempos) que involucra una funcionalidad.

Antes de pasar a observar un caso de nuestro ejemplo, se exhibe un resumen del método de temporización a usar.

4. Definición Informal de un DEERT

Un DEERT es un diagrama de cambio de estados en donde la sincronización de sus eventos es controlada mediante un conjunto finito de variables reales denominadas *relojes*, cuyos valores se incrementan uniformemente con el paso del tiempo. Las restricciones temporales son expresadas ligando *condiciones de cambio de estados* a las transiciones de estados. Un cambio de estado puede producirse si su condición asociada es satisfecha por los valores corriente de los relojes. Un reloj puede ser puesto en cero al producirse un cambio de estado que así lo indique. A todo instante el valor de un reloj es igual al tiempo transcurrido desde la última vez en que fue puesto en cero.

También se deben definir condiciones (mediante el uso de los relojes) sobre los estados del diagrama para reflejar durante cuanto tiempo se puede permanecer en tal estado. Estas condiciones se denominan *Invariantes de estado*. Es decir entonces, que una transición puede tener asociado su *etiqueta*, una o más *condiciones* atómicas de cambio de estado[1], y el conjunto de relojes que son puestos en cero (opcional) cuando el cambio de estado se produce (a esto último llamaremos *afectaciones*)

Ejemplo:

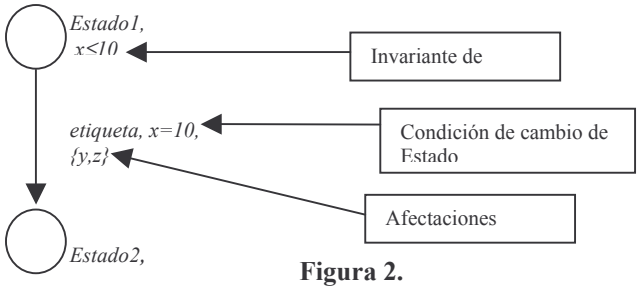

Figura 2.

Figura 2.Cuando el sistema alcanza el *Estado1*, puede permanecer en él por un período de 10 unidades de tiempo, luego, en el momento que el reloj *x* llega a un valor *10*, la condición asociada a la transición es satisfecha y por lo tanto en ese preciso instante debe producirse el cambio de

[1] Cuando la condición asociada contiene mas de una condición atómica, dicha condición (compuesta) es satisfecha cuando todas sus condiciones atómicas lo son.

estado[2], además los relojes ***y*** y ***z*** son reseteados a cero. Por razones de simplicidad cuando no se especifica la condición de una transición o un *Invariante* de estado, se asume la condición *true*. Se debe tener en cuenta que el tiempo solo pasa en los estados, es decir que los cambios de estados son instantáneos. Al inicio del diagrama todos los relojes comienzan de cero.

Continuando con el ejemplo de la Plataforma Web de Educación a Distancia, podemos especificar con este tipo de diagrama situaciones como, por ejemplo, los cambio de estados de un curso y algunas actividades en su desarrollo. Tal como se mencionó en la descripción del ejemplo cuando se planifica un curso para su desarrollo con modalidad a distancia y hacer uso de la plataforma, sus contenidistas y tutores deben tener previsto las etapas que el curso abarca y detalles de sus actividades, como por ejemplo momentos y duraciones de ciertos trabajos que involucran el uso de algunas de las tecnologías (chat, foro, correo) que la Plataforma ofrece (ver **Figura 3**).

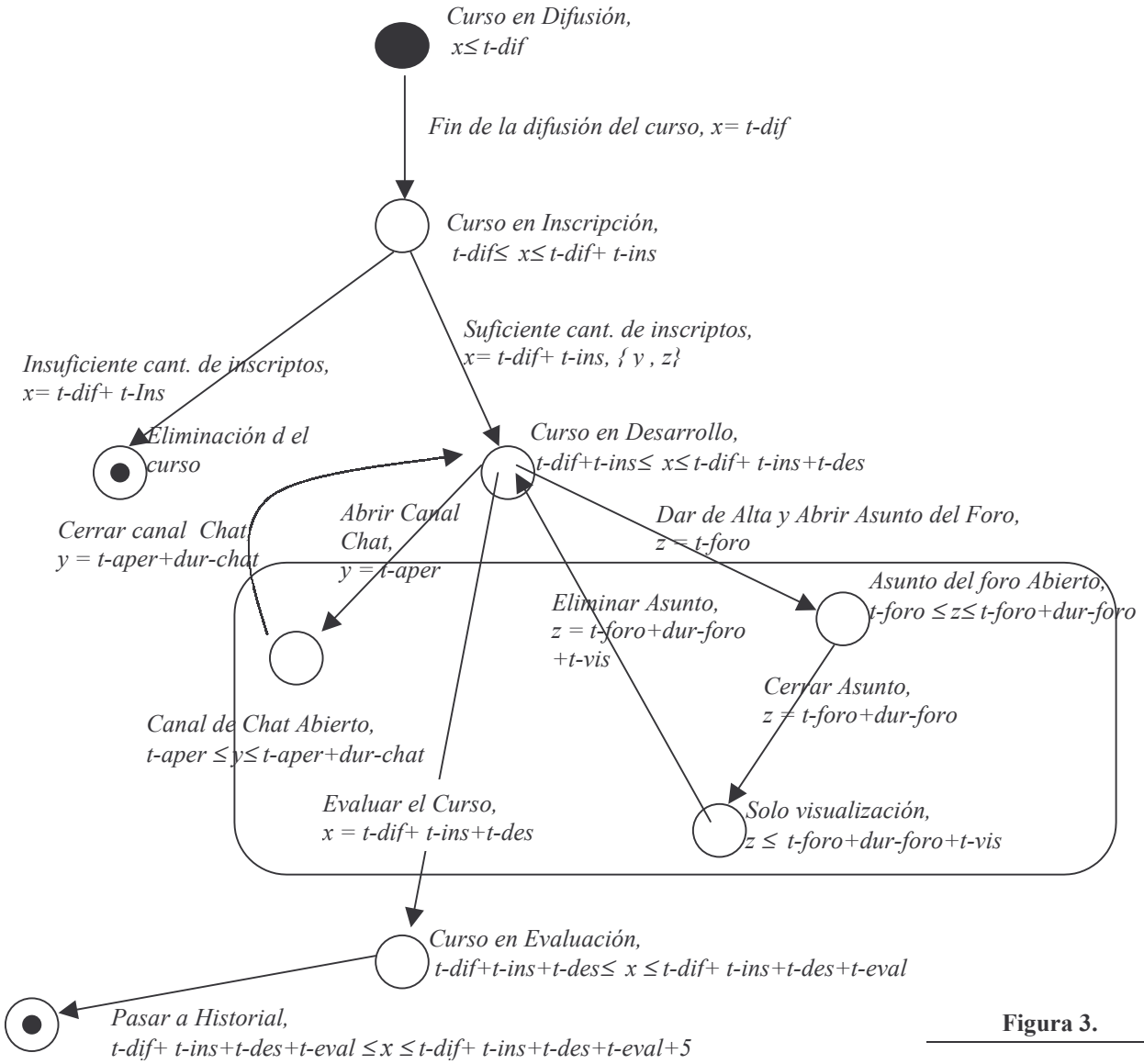

Figura 3.

4.1 Explicación del diagrama de la Figura 3

El estado inicial del diagrama corresponde a la etapa de difusión del curso, luego de un período *t-dif* el curso cambiará a estado Inscripción. Al cabo de un tiempo *t-ins* el curso dará comienzo si este ha alcanzado la cantidad suficiente de inscriptos, en caso contrario aborta y se elimina de la

[2] El cambio de estado debe producirse en el momento exacto en que el reloj x alcanza un valor 10, dado que la condición expresa dicha situación. Si dicha condición fuera de la forma 10≤*x*≤15, entonces el cambio de estado podría realizarse en cualquier momento en el intervalo de tiempo [10,15] correspondiente al valor del reloj *x*.

Plataforma. Durante su desarrollo, existen actividades predefinidas por los tutores que hacen uso, en este caso, del servicio de Chat y Foro. El Tutor puede desear que un canal de Chat se mantenga abierto para la comunicación durante un cierto período ([*t-aper .. t-aper+dur-chat*]). También puede tener predefinido que del momento *t-foro* al momento *t-foro+dur-foro* (a partir del comienzo del curso) los alumnos deberán aportar sus opiniones en un asunto determinado del foro, luego de este último día y durante *t-vis* unidades de tiempo mas el Asunto estará disponible para la visualización de sus opiniones y posteriormente será eliminado.

Notar que para reflejar estas actividades, se utilizaron dos relojes extras (*y*, *z*) los cuales fueron puestos en cero cuando el curso paso a estado *Curso en Desarrollo*, es decir, cuando dio comienzo.

Luego de un lapso de tiempo *t-des* el curso pasará a una etapa de evaluación, en la cual los alumnos solo puede ingresar a la plataforma para su visualización (en particular de sus notas), y los tutores para cargar dichas notas y hacer una autoevaluación del desarrollo del curso.

Por último, como se puede observar en el diagrama, luego de un tiempo *t-eval* y en cualquier momento del intervalo [0..5] el curso será añadido a un Historial de cursos desarrollados con la Plataforma.

La **unidad de tiempo a considerar queda a interpretación del diseñador**, en algunos casos lo mas apropiado puede ser especificar bajo la interpretación de que cada unidad de tiempo es un día, en otros casos horas, etc.

5. Conclusión y Trabajos a Futuro

Esta ponencia resume y presenta de alguna manera por un lado, que diagramas podrían ser apropiados en cada una de las actividades del diseño de una Aplicación Web, por cuestiones de resumen en algunas actividades no se exhibieron ejemplos. Para ello se trató de rescatar algunos aspectos en común que tienen estas aplicaciones y mediante adaptaciones de los diagramas que nos provee el UML especificar dichos aspectos. Se debe aclarar que en algunas actividades, para una mayor claridad, es necesario hacer uso de otros lenguajes de diseño como por ej. HDM[3] o WebML[6]. Por otro lado el propósito mas importante de este artículo es introducir un método de temporización, aprovechando el dinamismo de los diagramas de evolución de estados, para reflejar aquellas situaciones en donde es necesario clarificar no solo el orden de ciertas actividades sino también sus momentos. Esto hace posible que el equipo de desarrollo pueda automatizar, respetando las restricciones de tiempo, una gran cantidad de actividades que sobrecargaría su trabajo.

6. Referencias

[1] G.Booch, I.Jacobson and J.Rumbaugn,"The Unified Modeling Language User Guide" 1998.

[2] L. Baresi, F. Garzotto and P. Paolono, "Extending UML for Modeling Web Applications". Proceedings of the 34th Hawaii International Conference on System Sciences – 2001.

[3] F. Garzotto, P. Paoloni, D. Schwabe, "HDM – A Model-Based Approach to Hypertext Application Design" TOIS 11(1) pp.1-26 – 1993.

[4] D. Schwabe, G. Rossi and F. Lyaardet, "Improving Web Information System with Navigational Patterns", in Proc. Of Int. Conf. WW8, Elsevier, pp.589-600.

[5] D. Schwabe, G. Rossi, "An Object Oriented Approach to Web-Based Aplication Design", Theory and Practice of Object Systems, 4(4), J. Wiley – 1998

[6] María A. Nieto-Santisteban **, "**Ingeniería Web - Construyendo Web Apps.", I Jornadas de Ingeniería Web – 2001.

[7] *[PRE00]* R. Pressman, "Software Engineering: A Practitioner´s Approach. 5 th.

[8] Hans A. Schmid, Gustavo Rossi, "Modeling and Designing Business Processes in Web Applications".

[9] [UML 01] UML technical documentation. www.uml.org