INGENIERÍA DE AMBIENTES SOFTWARE: COMBINANDO EXPRESIVIDADES TEMPORALES Y OBJETUALES EN LA FASE DE MODELIZACIÓN CONCEPTUAL

Oscar Pastor[†], Carlos Heuser[††], Jaime Gómez[†††], Emilio Insfrán[†]

[†] Departament de Sistemes Informàtics i Computació
Universitat Politècnica de València
Camí de Vera s/n
46071 Valencia - España
opastor@dsic.upv.es

[††]UFRGS – Instituto de Informática
Caixa Postal 15064 – 91501-970 Porto Alegre RS - Brasil
heuser@inf.ufrgs.br

[†††] Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante
Carretera San Vicente s/n
03690 San Vicente del Raspeig - Alicante - España
jaime@dlsi.ua.es

RESUMEN

En este trabajo se presenta la fusión de dos aproximaciones metodológicas: OO-Method, basada en el uso de un modelo objetual con un soporte formal preciso, y TempER-Tr, que usa un modelo entidad-relación temporal para dar cuenta de la arquitectura estática del sistema y un nuevo lenguaje de especificación de transacciones para describir aspectos de comportamiento. La idea es sentar las bases de un nuevo método que fije una aproximación metodológica unificada para especificar los aspectos estáticos y dinámicos asociados a un sistema de información, aprovechando las propiedades más relevantes de cada uno de estos enfoques.

PALABRAS CLAVE

investigación, orientación a objetos, entidad-relación, redes de petri, modelado conceptual, análisis de requisitos

1. INTRODUCCIÓN

El ciclo de producción de software clásico consiste en un proceso dividido en etapas, en las que se parte de los aspectos más abstractos a los más concretos. Las últimas etapas de dicho proceso (codificación, mantenimiento, etc.) se han visto auxiliadas por herramientas automáticas (lenguajes de programación de alto nivel, compiladores, depuradores, etc.) que, a través de la automatización, aseguran la calidad

del producto obtenido. Lo mismo no acontece en las etapas previas de dicho ciclo, lo que es fuente de errores e ineficiencias, afectando fuertemente a la productividad y calidad del software desarrollado.

Cada vez más se están invirtiendo esfuerzos en la definición y construcción de herramientas basadas en métodos formales, y así asistimos a la emergencia de toda una disciplina, la Ingeniería de Requerimientos, en la que la especificación de requerimientos se ve soportada por distintas lógicas que capturan aspectos relevantes establecidos en la modelización conceptual.

La elección de un modelo adecuado como paradigma que soporte del ciclo de vida no solamente permite minimizar el gap semántico existente entre el modelo conceptual y el modelo de ejecución, sino que permite tratar de forma homogénea los aspectos relacionados. Es por ello que el modelo orientado a objetos, dada su expresividad, se muestra especialmente adecuado y es el elegido para abordar el proceso de modelización conceptual.

En el contexto del proyecto IDEAS (Ingeniería de Ambientes Software), financiado por la CYTED, y en particular en su tarea 2 titulada "Metodología de especificación y diseño de transacciones", los grupos de investigación del DSIC-UPV (Valencia, España) y del Instituto de Informática de la UFRGS (Porto Alegre, Brasil) se han fijado como objetivo definir un ambiente de producción de software basado en la fusión de sus dos aproximaciones metodológicas: OO-METHOD en Valencia, TempER-Tr en Porto Alegre, que, potenciando sus aspectos considerados más relevantes, generen un entorno de trabajo en la que se fije una aproximación metodológica que permite especificar los aspectos estáticos y dinámicos asociados a un sistema de información.

Aunque se pretende, como decimos, generar un ambiente en el que se complementen aspectos estáticos y dinámicos, se incide especialmente en una metodología de especificación y diseño de transacciones, porque el concepto de transacción aporta un gránulo de modelización del comportamiento de un sistema a un nivel de abstracción mayor, que se corresponde con servicios complejos que hace uso a través de un mecanismo de *event calling* de otros servicios más elementales aportados por otras clases preexistentes ya definidas.

En el contexto de IDEAS, en este trabajo se presenta el estado actual del proyecto. Para ello, después de esta introducción, se presentan en los apartados 2 y 3 los aspectos más relevantes en el momento actual de OO-Method y de TempEr-Tr respectivamente como aproximaciones metodológicas. A continuación, en el apartado 4 se efectúa un estudio comparativo entre ambos, para terminar fijando en el apartado de conclusiones las líneas de actuación a corto plazo.

2. OO-METHOD

OO-Method (Pastor 1992), (Pastor et al. 1996), (Pastor et al. 1997) es una metodología OO para la producción automática de software que ofrece un marco riguroso para la especificación de sistemas de información que incluye:

- una potente y sencilla notación gráfica para tratar con la fase de análisis.

- ***OASIS*** (Canós et al. 1995), (Pastor et al. 1995) como un lenguaje de especificación formal y OO que constituye el repositorio de alto nivel del sistema.

- la definición de un preciso modelo de ejecución que guía la fase de implementación.

Al abordar el proceso de construcción de software se definen dos fases principales:

1 *Modelado conceptual*. Se recogen las propiedades esenciales que definen el sistema sin tener en consideración detalles de implementación. Como resultado de esta fase se obtienen:

- Modelos de análisis: objetos, dinámico y funcional.

- Una especificación formal y orientada a objetos en OASIS que constituye un diccionario de datos de alto nivel y que es generada automáticamente.

2 Aplicación de un *Modelo de Ejecución.* Se fijan los detalles a ser considerados para implementar adecuadamente el Modelo Conceptual obtenido. Se especifica claramente como mapear cada una de las nociones del modelo OO recogidas en los modelos de análisis en estructuras de programación incluyendo todas aquellas propiedades dependientes de la implementación y que no forman parte directa del espacio del problema.

2.1 Modelo Conceptual

Se empieza la fase de análisis con la construcción de tres modelos: Objetos, Dinámico y Funcional. Estos modelos describen la sociedad de objetos desde tres complementarios puntos de vista y dentro de un marco riguroso OO.

MODELO DE OBJETOS

El Modelo de Objetos es descrito gráficamente por un Diagrama de Configuración de Clases (DCC) que muestra la estructura y comportamiento de todas las clases identificadas en el dominio del problema así como sus relaciones.

Una clase se representa gráficamente como una caja dividida en secciones donde se recoge información sobre sus atributos y servicios. Los servicios se declaran especificando su nombre y argumentos, distinguiendo entre los eventos de creación, borrado y los eventos compartidos como puede verse en (Pastor et al. 1996).

Para el tratamiento de la complejidad se pueden definir relaciones estructurales en términos de agregación (parte-de) y herencia (es-un).

En la Figura 1 se presenta la relación de agregación entre clases, incluyendo información sobre cardinalidades (mínimas y máximas) que determinan cuantos objetos componentes pueden estar relacionados con un objeto compuesto e inversamente, cuantos objetos compuestos pueden estar relacionados con un objeto en particular.

Figura 1. Relación de agregación entre clases

Adicionalmente, una agregación debe ser indicada como inclusiva/referencial o estática/dinámica, cuya explicación detallada puede encontrase en (Pastor et al. 1996).

Una asociación es un caso especial de agregación para tratar con colecciones de objetos usando una cláusula de agrupación GROUP BY. Una propiedad relevante de la asociación es la existencia implícita de eventos de inserción y borrado de objetos, de acuerdo a la especificación de la cláusula Group By.

La relación de herencia puede ser expresada mediante los operadores de especialización y generalización. Gráficamente se representan como flechas que conectan la subclase con la superclase y son etiquetadas con sus propiedades relevantes. En el caso de especialización puede ser etiquetada por una condición de especialización o con los eventos correspondientes de activación/cancelación de la relación de *rol* (especialización temporal).

Para finalizar el Modelo de Objetos, señalamos que una clase está completamente definida (desde esta perspectiva) por un *conjunto de fórmulas* que cubren el resto de propiedades, como ser:

- restricciones de integridad que establecen condiciones que los objetos deben satisfacer.

- derivaciones, para la definir atributos cuyos valores son calculados en función de otros.

MODELO DINÁMICO

El Modelo Dinámico especifica los aspectos relacionados con el control, vidas posibles, secuencia de eventos e interacción entre objetos. Se representa por tipos de diagramas: Diagramas de Transición de Estados (DTE) y Diagrama de Interacción de Objetos (DIO).

Diagramas de Transición de Estados

Los DTEs describen el comportamiento de objetos estableciendo *vidas posibles*. Una *vida válida*, es una secuencia correcta de estados que caracterizan un comportamiento correcto para todos los objetos de una clase.

En este contexto, los estados denotan situaciones en las que pueden encontrarse los objetos como consecuencia de la ocurrencia de eventos relevantes. Las transiciones representan los cambios de estado válidos. Estas pueden ser restringidas introduciendo condiciones. La sintaxis es la siguiente:

evento | *acción* | *transacción* ***[if*** *precondición]* ***[when*** *condición-de-control]*

donde *precondición* es una condición definida sobre atributos del objeto y que debe ser satisfecha de forma a que el servicio pueda ocurrir. Una *condición-de-control* es una condición que evita el indeterminismo para la transición de un estado a otro.

Diagrama de Interacción entre Objetos

La interacción entre objetos se modela gráficamente mediante un Diagrama de Interacción entre Objetos (DIO). En el DIO podemos especificar dos interacciones básicas:

- *disparos:* son servicios de una clase que se activan de forma automática cuando se satisface una condición en un objeto dicha clase.

- *interacciones globales:* son *transacciones* compuestas por servicios de clases diferentes.

Los *disparos* se definen como flechas que van desde la cabecera de una clase al servicio activado y se etiquetan con la condición de disparo. Las *interacciones globales* se introducen conectando los servicios que componen la interacción y nominándola mediante un identificador de interacción global (idIG).

MODELO FUNCIONAL

El Modelo Funcional captura la semántica asociada a los cambios de estado de un objeto como consecuencia de la ocurrencia de un evento. El valor de cada atributo se puede modificar dependiendo de la acción ocurrida, de los argumentos del evento y/o del estado actual del objeto.

La especificación del cambio de estado viene determinado por la categorización atributos de entre un conjunto predefinido de tres *categorías*. Estas categorías determinan qué información se necesita para determinar cómo cambia el valor del atributo ante la ocurrencia de determinados eventos. Las tres categorías de atributos son:

- ***Cardinales*****:** sus eventos relevantes incrementan ó decrementan el valor del atributo.

- ***Independientes del estado*****:** el valor del atributo sólo depende de la última acción.

- ***Situación*****:** una acción *portadora* asigna al atributo un valor de un *dominio discreto*.

La información que se obtiene al finalizar la construcción de los tres modelos de análisis constituye el modelo conceptual y posee su representación textual en OASIS. Esta representación textual constituye un repositorio completo y formal del alto nivel del sistema, y puede obtenerse en cualquier momento a partir de los modelos construidos.

2.2. El Modelo de Ejecución

El *modelo de ejecución* es un patrón genérico de comportamiento, aplicable a cualquier modelo conceptual construido siguiendo la estrategia propuesta por OO-Method. La idea consiste en dar una visión abstracta del modelo que determinará

directamente el patrón de programación a seguir cuando un desarrollador (o una herramienta CASE) aborde la fase de implementación y se definan las características del producto software final en términos del control de acceso a usuarios, activación de servicios, interfaz de usuarios, etc.

El modelo de ejecución tiene tres fases esenciales:

1. **Control de acceso**: en primer lugar, el objeto[1] que desee conectarse al sistema deberá identificarse como miembro de la sociedad de objetos.

2. **Vista del sistema**: una vez se ha conectado un usuario (u otro objeto del sistema), tendrá una visión clara de la sociedad de objetos en términos de qué clases de objetos puede ver, qué servicios que puede activar y qué atributos que puede consultar.

3. **Activación de servicios**: por último, el objeto deberá ser capaz de activar cualquier servicio disponible y de realizar las observaciones pertinentes.

Cualquier ***petición de servicio*** se caracterizará por la siguiente secuencia de acciones:

- Identificación del objeto.
- Introducción de argumentos del servicio a activar.
- Corrección de la transición entre estados.
- Satisfacción de precondiciones.
- Realización de las evaluaciones (modificación del estado del objeto).
- Comprobación de las restricciones de integridad en el nuevo estado.
- Comprobación de las relaciones de disparo.

Estos pasos, claramente definidos en (Pastor et al. 1997), guían el proceso de implementación asegurando la equivalencia funcional entre la descripción del sistema recogida en el modelo conceptual y su reificación en un entorno de programación.

3. MODELO INTEGRADO TempER-Tr

La integración del modelo entidad-relación (Chen 1976) con redes de Petri de alto nivel [Jen91] ha mostrado ser una vertiente prometedora. Existe una aproximación que se fundamenta básicamente en términos de la teoría de redes de Petri no sólo en aspectos de comportamiento, sino también en las propiedades estáticas, se trata de la aproximación ER-Tr (entidad-relación-transacción) (Heuser et al. 1991). El modelo TempER-tr presenta una propuesta de evolución referida al modelo ER-Tr en dos puntos:

- las propiedades estáticas pasan a ser modeladas por un modelo temporal.

- un lenguaje de descripción de transacciones que estaba centrado en la especificación de entidades individuales pasa a manipular conjuntos de entidades de forma más satisfactoria, utilizando algunos conceptos del lenguaje SQL.

Un diagrama TempER-Tr está compuesto de dos grupos de elementos: un grupo representando las propiedades estáticas del sistema (Modelo Entidad-Relación Temporal) y un grupo que representa las propiedades dinámicas del sistema (Modelo de Transacciones).

3.1 Modelo Entidad-Relación Temporal

El modelo TempER, como cualquier otro modelo ER, presenta como elementos básicos entidades y relaciones. Por definición, cada entidad modelada presenta un OID generado internamente por el sistema. Este OID identifica a una entidad con relación a las demás entidades en el contexto del universo de discurso. Además, permite que se represente en un mismo diagrama elementos temporizados y elementos no-temporizados. Los elementos que tienen una *T* representan elementos temporizados.

Un modelo de datos TempER está compuesto de dos partes: un diagrama que muestra los conjuntos de entidades y conjuntos de relaciones, y un diccionario de datos que describe los atributos de entidades y relaciones.

Las características básicas de este modelo son:

- La composición de los conjuntos-entidad y los conjuntos-relación tiene forma de tabla, donde la primera columna se refiere a intervalos de tiempo en que las entidades y/o relaciones ocupan una dimensión temporal. Para representar el tiempo se utilizan elementos temporales: uniones finitas de intervalos de tiempo (Gadia et al. 1993).

- Para expresar las cardinalidades, el modelo TempER adopta el principio de participación de entidades en conjuntos-relación. La lectura de las restricciones de cardinalidad depende de la clasificación temporal del conjunto-relación y por tanto, la lectura debe ser echa en función de los intervalos de tiempo.

- Los conjuntos-entidad temporizados pueden mezclar atributos temporizados y no temporizados.

3.2 Modelo de Transacciones

El modelo de transacciones se compone de los siguientes elementos:

- **Agentes externos**: representados por rectángulos sombreados que introducen datos al sistema relativos a los eventos que ocurren en el ambiente y/o reciben una respuesta del sistema proveniente de la ocurrencia de alguna transacción interna.

- **Conjunto-transacción**: elementos que definen las transacciones del sistema.

- **Flujos externos**: flechas que conectan un agente externo a un conjunto-transacción. Representando entrada/salida de datos al sistema.

- **Flujos internos**: flechas que conectan un conjunto-transacción a elementos del modelo TempER. Pueden ser de varios tipos: flujos de inclusión, flujos de

exclusión, flujos de alteración, flujos de verificación de presencia y flujos de verificación de ausencia.

Los flujos de inclusión, de exclusión y de alteración expresan que las respectivas transacciones modifican el contenido de los conjuntos-entidad /conjuntos-relación a los que están conectados. Los flujos de verificación de presencia y de ausencia verifican el contenido de éstos, sin alterarlos. Los flujos de alteración únicamente modifican los valores de atributos, ni incluyen ni excluyen entidades o relaciones, y ni amplían ni reducen la dimensión temporal de cualquier objeto.

3.3 Lenguaje de anotación del modelo TempER-Tr

Las anotaciones que acompañan los flujos internos y externos componen lo que se denomina el lenguaje de anotación del modelo TempER-Tr.

Anotaciones de flujos internos de una transacción

Las anotaciones junto a los flujos internos especifican tablas que pueden ser de exactamente un elemento -tablas unarias- o de un número variable de elementos -tablas multielemento- (prefijadas con un "*"). Estas tablas son denominadas tablas de flujo interno. Cada tupla de estas tablas contiene datos sobre entidades, si el flujo está conectado a conjuntos-entidad, o sobre relaciones, si el flujo está conectado a conjuntos-relación. En estas tablas existen columnas para la identificación de entidades (columnas de OID's) y columnas representando atributos.

Existen tres alternativas de composición para las columnas de atributos:

- si la anotación no tiene lista de atributos, la tabla de flujo interno necesariamente tendrá todos los atributos que se encuentran definidos en el esquema del diccionario de datos relativo al conjunto-entidad o al conjunto-relación conectado.

- si en la anotación está indicado qué atributos deben participar en la tabla, sólo aparecen éstos.

- si en vez de una lista de atributos está presente el literal *none*, la tabla de flujo interno no contendrá ninguna columna referente a atributos.

En cuanto a las columnas de OID's pueden darse dos situaciones:

- si la tabla de flujo conectada al conjunto-entidad sólo contiene una columna de OID's esta columna lleva el nombre especial de OID.

- si la tabla de flujo conectada al conjunto-relación contiene una columna de OID para cada conjunto-entidad asociado, cada una de esas columnas es identificada por el nombre del conjunto-entidad asociado, escrito en letras mayúsculas.

Para cada tipo de flujo interno existen una serie de reglas que deben ser obedecidas en la respectiva anotación y que están explicadas en (Heuser et al. 1997).

Anotaciones de flujos externos de una transacción

Las anotaciones junto a los flujos externos definen datos que entran o que salen del sistema. Cada variable presente en estas anotaciones, o se refiere a un dato atómico, o a datos multivaluados.

Anotaciones de fórmulas de una transacción

Cada conjunto-transacción puede poseer un conjunto de fórmulas cuya finalidad es restringir el conjunto de entidades y relaciones que participan de una determinada transacción. Una fórmula es una relación entre dos o más términos que puede ser verdadera o falsa.

Los términos envueltos en estas fórmulas pueden ser:

- escalares: elementos que contienen valores atómicos.
- columnas derivadas: columnas resultantes de una operación de proyección sobre una tabla, también se puede tratar de un vector de valores anotados en un flujo externo de entrada.
- tablas derivadas: es el resultado de operaciones de proyección, de selección y de unión sobre tablas existentes de flujos de transacción.
- elemento temporal: define un conjunto de intervalos de tiempo, existen varias formas de representación.
- asignación temporal: es el valor de los atributos temporizados, es una expresión que denota una serie de valores que estos atributos pueden asumir a lo largo del tiempo.

El modelo TempER-Tr adopta la semántica del comando *Select* del lenguaje SQL como uno de los principales medios para producir columnas y tablas derivadas. Sin embargo, el modelo TermpER-Tr omite la palabra *Select*, utilizando sólo las cláusulas *from* y *where*.

3.4 Comportamiento del modelo TempER-Tr

El comportamiento del modelo TempER-Tr es semejante a una red de Petri de alto nivel de tipo condición/evento, es decir, existen reglas que condicionan la habilitación de una transacción y reglas que definen el efecto de una transacción.

Los eventos que accionan las transacciones (agentes externos), están siempre habilitados por definición. Cada vez que ocurren, una nueva tupla de datos es colocada en un flujo externo que conecta los respectivos agentes externos con los conjuntos-transacción.

Para entender el comportamiento de una transacción primero es preciso establecer lo que serán objetos de entrada y objetos de salida de esta transacción.

Los objetos de entrada de una transacción son aquellos especificados en flujos internos cuyas flechas apuntan para el círculo que representa el conjunto-transacción,

mientras que los objetos de salida son aquellos especificados en los flujos que se originan (salen) del círculo que representa el conjunto-transacción.

Una transacción estará habilitada si ocurre que:

- Las tuplas de valores de entrada están presentes en todos los flujos externos de entrada conectados a la transacción.
- Todos los objetos de entrada están presentes en los respectivos conjuntos de entidades y relaciones.
- Todos los objetos de salida están ausentes de los respectivos conjuntos de entidades y relaciones.
- El estado que la base de datos alcanza como resultado de la ocurrencia de una transacción no viola las restricciones de integridad y otras que han sido definidas en el modelo de datos temporal.

La ocurrencia de efectuar una transacción provoca los siguientes efectos:

- Las tuplas con los valores de entrada desaparecen de los flujos externos de entrada.
- Las tuplas con los valores de salida aparecen de los flujos externos de salida.
- Los objetos referentes a los flujos de exclusión desaparecen de los respectivos conjuntos de entidades y relaciones.
- Los objetos referentes a los flujos de inclusión aparecen en los respectivos conjuntos de entidades y relaciones.
- Los atributos especificados en los flujos de alteración son modificados, o sea, sus valores anteriores desaparecen y los nuevos valores aparecen.

Todas estas alteraciones ocurren a la vez, esto es, una transacción es una transición atómica de un estado del sistema hacia otro y ningún estado intermedio es observado.

Dos transacciones que presenten algún objeto de entrada o algún objeto de salida en común se dice que están en conflicto. El modelo TempER-Tr no resuelve el conflicto, pues no establece qué transacción será efectuada. Esto se configurará en función del no-determinismo, que se deja para el nivel de modelado conceptual del sistema. Dos transacciones que no poseen objetos de entrada y objetos de salida en común se dicen concurrentes y son independientes entre sí. El modelo TempER-Tr determina cualquier secuencia de efectos de transacciones concurrentes.

4. OO-METHOD VERSUS TempER-Tr

La comparación entre las aproximaciones OO-Method y TempER-Tr nos sugiere una serie de consideraciones muy interesantes, que constituyen los objetivos del proyecto IDEAS presentado en la introducción. En particular

1 El uso del paradigma objetual en OO-Method podría ser aplicado a TempER para estructurar la fase de modelización conceptual, ya que la utilización de una noción

única de objeto facilitaría desde el punto de vista metodológico la elaboración de un modelo de objetos, y la especificación de sus servicios siguiendo la notación TempER para especificar transacciones, entendiendo que a cada servicio le correspondería una transacción

2□ En línea argumental con el punto anterior, la arquitectura estática del sistema de información analizado se representa en OO-Method con un modelo de clases, en el que todo son clases, y las relaciones entre clases son de agregación (con su cardinalidad) y de herencia. Por el contrario, en TempER se utiliza un enriquecimiento de modelo entidad relación clásico, en los componentes son clases y relaciones. Una posible vía de fusión sería considerar a las relaciones como clases, manteniendo la especificación de cardinalidad entre dichas clases.

Con los dos puntos anteriores, estamos proponiendo un TempER-Tr OO (orientado a objetos), que incorpore las ventajas del modelo OO a nivel de modelización conceptual.

3□ La comparación entre el lenguaje de descripción de transacciones de TempER-Tr y la descripción de comportamiento efectuada en un ambiente OO-Method es muy ilustrativa. Vamos a resumir a continuación los puntos más relevantes:

- Las tablas multielemento de flujo interno que utiliza TempER-Tr tienen una semántica precisa en el marco objetual de OO-Method: se corresponden con servicios proporcionados por una clase asociación, que agrupa a una colección de objetos como instancia de una clase agregada por asociación. Ello proporciona claridad conceptual a la manipulación de ese tipo de situaciones.

- Los flujos de alteración o modificación en TempER-Tr exigen en su especificación, como es lógico, que se precise qué atributos van a ser modificados y cómo. Para ello introduce como ayuda al especificador, dos prefijos especiales new# y old#, que representan el valor de un atributo antes y después de la ejecución de la transacción. En el modelo objetual OASIS subyacente a OO-Method, la especificación de las fórmulas de evaluación de como fórmulas del tipo:

  F [evento] F’

  (donde F es una fórmula que debe satisfacerse en el estado origen en el que ocurre el evento *evento*, y F’ es una fórmula que debe satisfacerse en el estado final), en el marco de una lógica dinámica formalmente caracterizada, le proporciona una semántica precisa a ese tipo de situaciones. La sintaxis asociada a las evaluaciones permite usarlas de forma sencilla en la especificación de un sistema.

- La especificación de los flujos de comprobación de presencia o de ausencia en TempER-Tr se corresponde también en OO-Method con otro tipo de fórmulas dinámicas usadas en la especificación de una clase, que son las precondiciones, representadas como:

  not F [evento] false

y con el significado de si el evento *evento* es activado y F no se satisface, no hay estado final alcanzable posible. Como ejemplo sencillo de precondiciones implícitas en OO-Method, tenemos la de no-existencia para el evento de creación de una clase, y la de existencia del objeto para activar su evento de destrucción. En TempER-Tr éstos serían ejemplos de flujos de verificación de ausencia y de presencia, respectivamente.

4□ La incorporación de la dimensión temporal que aparece en TempER-Tr proporciona un marco expresivo potente y sencillo, inexistente en OO-Method. Este punto merece ser analizado con más detalle.

La dimensión temporal de una entidad como tal no existe como posibilidad expresiva en la declaración de una clase OASIS. Existen, no obstante, posibles representaciones:

- basadas en declarar una clase para especificar las propiedades de la población actual, entendiendo actual como existente en el tiempo del sistema, y guardar complementariamente como instancias de clase componente (vistas como un atributo multivaluado de la clase de la que hablamos) los intervalos temporales de existencia con sus propiedades particulares, a efectos de histórico.

- también se podría, de manera alternativa, introducirse atributos para representar el tiempo y considerar como distintas instancias de una única clase a las distintas modificaciones sufridas por un componente a lo largo de su historia.

- una tercera posibilidad consistiría en usar dos clases, una para denotar las propiedades en el estado actual y otra relacionada con la primera para almacenar la información de tipo histórica.

Desde ese punto de vista, los flujos de inclusión conectados a entidades temporalizadas, por ejemplo, tienen la siguiente representación en OASIS:

- si es la primera vez que existe la entidad, es un evento de creación en la clase correspondiente

- si ya existía y lo que ocurre es que se amplía su existencia con un nuevo intervalo temporal, en OASIS se tienen dos posibles representaciones:

    - mantener una única clase, cada intervalo de existencia deberá ser una instancia de una clase agregada simple, en la que se almacena la fecha de inicio para la nueva situación y la fecha de fin para la que se agota.

    - si se tiene una representación basada en clase única, se debería introducir una nueva instancia, cuyo *oid* lo constituiría el oid del objeto implicado más la etiqueta temporal del momento de activación del evento.

    - Si se utilizan dos clases, una para el estado actual de la población activa de la clase, y otra a efectos de histórico, en este segundo caso:

        1□ se insertará una instancia en la clase que representa el estado actual

2□ y se borrará de dicha clase, insertándose en la clase histórica la instancia que ha pasado a tener la categoría de histórica.

Con los flujos de exclusión, la situación es similar, aunque ahora en términos de eliminación de instancias. Dependiendo de la representación seleccionada para OASIS, tendríamos una forma de representar ese suceso de borrado.

En cualquier caso, se trata de una representación forzada por la carencia de esa expresividad temporal, cuya adición a OO-Method va a seguir siendo estudiada en el contexto de IDEAS.

5□ En cuanto al comportamiento de ambos modelos, aparecen muchos aspectos semejantes.

La ocurrencia de una transacción en TempEr-Tr requiere fundamentalmente:

- que se conozcan todos los datos de entrada especificados para la transacción,
- que los objetos implicados estén igualmente presentes,
- que los objetos definidos como objetos de salida desaparezcan del conjunto de entidades y relaciones que constituyen el banco de datos
- y que el estado final alcanzado sea válido, es decir, no viole restricción de integridad alguna.

El efecto de la transacción queda determinado por su especificación, de acuerdo con los flujos de inclusión, exclusión, alteración, verificación de presencia y de ausencia correspondientes.

En OO-Method se define un modelo abstracto de ejecución que fija los siguientes pasos en la ejecución de un evento o transacción:

- identificar el objeto servidor
- introducir los parámetros de entrada
- comprobar la precondición especificada para el servicio en cuestión
- realizar las evaluaciones, que cambiarán el estado de los objetos implicados
- comprobar que las restricciones de integridad se satisfacen en el estado final alcanzado
- comprobar si se cumple alguna condición de disparo que origine la activación automática de algún servicio

5. CONCLUSIONES

El trabajo realizado ha puesto de manifiesto que las aproximaciones metodológicas presentadas, OO-Method, basado en el uso de un modelo objetual con un soporte formal preciso y soportado por herramientas CASE que incluyen generación

automática de código, y TempER-Tr, que usa un modelo entidad-relación temporal para dar cuenta de la arquitectura estática del sistema, y un nuevo lenguaje de especificación de transacciones para describir aspectos de comportamiento, pueden ser la base de un nuevo método que fusione las propiedades más útiles de cada aproximación.

En particular, TempER-Tr se puede beneficiar del uso de un modelo OO como marco formal de referencia, lo que estructuraría el espacio del problema al incorporar las conocidas ventajas asociadas al uso del paradigma OO desde el punto de vista de la modelización conceptual del sistema. Ese TempER-TR OO debería de definir un nuevo modelo de objetos al estilo del modelo de objetos de OO-Method, pero manteniendo y profundizando en esa especificación de aspectos temporales no incorporados actualmente al modelo de objetos de OO-Method.

Desde ese punto de vista, como cada transacción en ese TempER-Tr OO se correspondería con un servicio ofertado por una clase del sistema, la especificación de dichas transacciones sería conceptualmente equivalente a la definición de un evento o transacción en OASIS, con lo que la fusión de los dos métodos sería una realidad.

Las estructuras de especificación basadas en distintos tipos de flujos que introduce TempER-Tr equivalen en OASIS a un conjunto bien caracterizado de fórmulas de una lógica dinámica que se usa como marco formal. El estudio pormenorizado de esa equivalencia debe ser realizado como tarea inmediata de investigación, para determinar qué estrategia de especificación es a la vez más correcta y más fácil de utilizar.

Todo este trabajo es el que da soporte a la tarea 2 del proyecto IDEAS (Ingeniería de Ambientes Software), financiado por la CYTED, y que en el marco de la red RITOS se ha propuesto como objetivo el de generar un avanzado entorno de producción de software basado en las aproximaciones metodológicas desarrolladas por los grupos del DSIC de la UPV en Valencia (OO-Method), y del Instituto de Informática de la UFGRS en Porto Alegre (Brasil) (TempER-Tr). Ese entorno se encuentra en fase de desarrollo de acuerdo con todas los resultados obtenidos hasta ahora, presentados de forma (necesariamente) breve en este trabajo.

REFERENCIAS

(Booch et al. 1997) Booch G., Rumbaugh J, Jacobson I 1997. "Unified Modeling Language. Vers. 1.0" Rational Software Corporation.

(Canos et al. 1995) Canós J.H., Penadés M.C., Ramos I. "A Knowdledge-Based Arquitecture for Object Societies". In Proc. DEXA95. (London. 1995): 18-25

(Chen 1976) Chen P. 1976. "The Entity-Relationship Model – Toward a Unified View of Data". ACM Transactions on Database Systems, New York, 1(1) (Mar): 9-28.

(Gadia et al. 1993) Gadia S., Nair S. 1993. "Temporal Databases: A prelude do parametric data". In: Tansel, A. et al (eds.) Temporal Databases – Theory Design and Implementation. Redwood City: The Benjamin/Cummings Publishing Co. Inc: 28-66.

(Heuser et al. 1991) Heuser C., Peres E. "ER-TR Diagrams: An Approach to specifying Database Transactions". 10th International Conference on the Entity Relationship Approach. San Mateo, California. 1991.

(Heuser et al. 1997) Heuser C., Antunes D. "Conceptual Modeling of Transactions combined with Temporal ER Diagram". 1997. Report Interno. UFRGS – Instituto de Informática Porto Alegre RS, Brasil. 1997.

(Jensen et al. 1991) Jensen K., Rozenberg G. 1991. High-Level Petri Nets – Theory and Application. Berlin-Heildelberg, Springer-Verlag .

(Pastor 1992) Pastor O. “OO-METHOD: An Object Oriented Methodology for Software Production”. Proc. DEXA 92, Springer-Verlag, pp. 121-127. ISBN: 3-211-82400-6. 1992

(Pastor et al. 1995) Pastor O., Ramos I. “OASIS 2.1.1: A Class-Definition Language to Model Information Systems Using an Object-Oriented Approach”. February 94 (1ª ed.), Mars 95 (2ª ed.), October 95 (3ª ed.). 1995.

(Pastor et al. 1996) Pastor O., Pelechano V., Bonet B., Ramos I. “An OO Methodological Approach for Making Automated Prototyping Feasible”. Proc. DEXA96. Springer-Verlag. September 1996.

(Pastor et al. 1997) Pastor O., Insfrán E., Pelechano V., Romero J., Merseguer J. “OO-METHOD: An OO Software Production Environment Combining Conventional and Formal Methods”. CAiSE97. June 1997.

[1] En nuestra visión de las aplicaciones los usuarios son también objetos.