

Método de Transformación de Prototipo a Bases de Diagramas UML y Casos de Aplicación*

Silvana Roncagliolo

Beatriz Marin

silvana@ucv.cl

beatriz.marin.c@ucv.cl

Escuela de Ingeniería Informática
Universidad Católica de Valparaíso
Av. Brasil 2241 – Casilla 4059
Fono 32 273761 – Fax 32 273859
Valparaíso, Chile

Resumen

Este trabajo presenta un método para obtener las bases de los diagramas UML a partir de un prototipo creado en la fase de captura de requerimientos. Estos diagramas son inicialmente los de casos de uso y de clases, los que se generan en base al trabajo con el prototipo, identificando atributos, clases, casos de uso, actores, etc. También se muestran dos casos en los que se aplica la metodología propuesta, los cuales ejemplifican la utilización de ésta al tener un prototipo para un sistema determinado.

Palabras claves: prototipo, UML, casos de uso, aplicación.

1. Introducción.

Una de las etapas más importantes en el desarrollo de un software es la de captura de requerimientos, ya que es la base para el resto del proceso de desarrollo. Por esta razón es primordial que la captura de requerimientos se haga en forma correcta, pero debido a los problemas de comunicación y escaso entendimiento entre desarrolladores y clientes, ésta es difícil la mayoría de las veces. Por este motivo es que los desarrolladores crean prototipos que apoyan la etapa de captura de requerimientos, los cuales generalmente son desechados para luego iniciar el desarrollo propiamente tal del sistema.

El uso de los lenguajes orientados a objeto generó la aparición una serie de métodos de desarrollo orientados a objeto similares, aunque cada uno con sus propios conceptos, notación, terminologías, etc. [Fow99]. Debido a estos múltiples métodos de desarrollo existían grandes inconvenientes para el aprendizaje, aplicación, construcción y uso de herramientas orientadas a objetos, por lo que Jim Rumbaugh, Grady Booch e Ivar Jacobson se unieron en Rational

* Este trabajo ha sido financiado por la Dirección de Investigación de la Universidad Católica de Valparaíso, Proyecto de Investigación Interno N° 209.726/2000.

Software Corporation en busca de la unificación de estas metodologías, así surge UML (Lenguaje Unificado de Modelado), que fue aceptado como estándar en 1997 por la OMG (Object Management Group).

Basados en estas ideas, este artículo presenta una metodología que propone una manera de obtener más utilidad del prototipo, traspasando la información inmersa en éstos a las bases de los diagramas UML; en consecuencia el trabajo inicial de realizar un prototipo no es desechado sino que es reutilizado en el desarrollo mismo del sistema. Esta metodología es el resultado de dos años de investigación, en donde primeramente fue enfocada al estudio e internalización conceptual de metodologías orientadas al objeto, de manera de fundamentar la importancia de UML; y luego fue enfocada al desarrollo mismo de la metodología y a la aplicación de ésta en dos casos de prueba.

2. Aspectos generales de UML.

El Lenguaje Unificado de Modelado (UML) es un lenguaje visual, que se usa para entender, diseñar, configurar, mantener y controlar la información sobre un determinado sistema. UML esta pensado para ser útil en el desarrollo iterativo de software [Rum00].

UML se basa en diversos modelos, los cuales sirven para abstraer la realidad a un nivel que sea comprensible sin perder detalles. Los modelos ayudan a captar los requerimientos y el dominio de un problema; además éstos se pueden utilizar para generar partes aprovechables para el desarrollo de un sistema, por ejemplo la interfaz de usuario, la base de datos, etc. [Cra99].

La descripción de los sistemas en UML se realiza a través de vistas, las cuales a la vez están integradas por diagramas [Rum00]. Esta estrategia de utilizar vistas surge del hecho de que un solo diagrama no puede expresar toda la información que se requiere para describir un sistema, por lo que se utilizan conjuntos separados de diagramas, las vistas, para representar proyecciones del sistema relacionadas con aspectos particulares, funcionales y no funcionales.

La vista lógica se compone de la vista estática, la vista de los casos de uso, la vista de interacción, la vista de la máquina de estados y la vista de actividades [Iva00]. La vista de los casos de uso es la principal, pues constituye el hilo conductor de todo el proceso de desarrollo; esto porque es la única que no describe aspectos de construcción del sistema, sino que de su comportamiento. La vista de los casos de uso es utilizada por todos los participantes en el proceso de desarrollo: los clientes, pues a través de ella se definen y expresan los requerimientos del sistema; los equipos de diseño, desarrollo y pruebas, pues esta vista muestra la funcionalidad del sistema tal como es percibido por los actores externos.

Por otra parte, la vista física se compone de la vista de implementación y la vista de despliegue.

Los casos de uso se entienden como un servicio que el actor puede obtener del sistema y el conjunto de casos de uso de un actor es denominado vista [Fow99]. Además se pueden definir otros dos tipos de casos de uso:

- Casos de uso con relación Extend: Se interpreta como una descomposición por generalización de un caso de uso.
- Casos de uso con relación Usa: Se interpreta como un caso de uso que es necesario que se presente como una pre_ejecución o post_ejecución a un caso de uso que entrega un servicio a un actor para que éste se realice satisfactoriamente.

Los actores se entienden como cualquier entidad que tenga interacción con el sistema y se pueden clasificar en [Ucv00]:

- Actores Primarios: Son los actores para los cuales fue creado el sistema. Por ejemplo: usuarios y otros sistemas.
- Actores Secundarios: Son los actores que existirán sólo si existen los actores primarios. Por ejemplo: Base de datos.

3. Propuesta de Metodología.

Al iniciar un desarrollo de un sistema informático los desarrolladores generalmente realizan prototipos para corroborar con el cliente la funcionalidad que el sistema debe entregar o para mostrarle al cliente la interfaz que tendrá el sistema, es decir, los prototipos son desarrollados principalmente para visualizar la funcionalidad del sistema, o sea, los usuarios del sistema, los servicios que estos tienen, menús, etc.

Luego de verificar la especificación de requerimientos con el cliente, generalmente los prototipos son dejados de lado, sin sacarles mayor provecho. Es por esto que se propone una metodología que ayude a los desarrolladores a generar los diagramas básicos de UML a partir del prototipo creado para el sistema, disminuyendo así el tiempo de desarrollo del sistema informático.

La relación entre los componentes del prototipo y los diagramas de casos de uso se esquematiza en la Tabla 1.

Prototipo	UML
Perfiles de usuario (implícitos y explícitos).	Actores primarios tipo usuario.
Servicios que un usuario obtiene de un sistema.	Casos de uso para cada actor.
Submenús dentro de los servicios de un usuario.	Caso de uso con relación extend.
Pre-condiciones o post-condiciones de un servicio de un usuario.	Caso de uso con relación usa.

Tabla 1: Relación entre componentes de un prototipo y diagramas de casos de uso.

Al realizar un sistema informático se debe partir por la identificación de los casos de uso presentes en el sistema, para luego hacer los diagramas. Si existe un prototipo de la aplicación, éste sirve como apoyo a la identificación de los actores y sus casos de uso.

Al realizar los diagramas a partir de un prototipo, estos deben ser revisados constantemente con la descripción de los casos de uso [Ucv00]. Este análisis se hace para verificar, en esta primera etapa, si la secuencia del curso normal de los eventos se representa correctamente con estos diagramas; si esto no ocurre se puede modificar este diagrama para obtener finalmente uno de mayor precisión.

Además, el prototipo ayuda a la generación de los diagramas de clases, los cuales son muy importantes no sólo para la visualización de las clases, interfaces, colaboraciones y relaciones entre ellas, sino que también sirven para la construcción del código de las aplicaciones.

Los elementos de los prototipos de una aplicación difieren según el lenguaje de programación utilizado, aunque se pueden identificar elementos comunes como lo son las cajas de texto, etiquetas, grillas, menús, cajas de

selección desplegables, cajas de opción, cajas de selección, áreas de texto, botones, etc. Los principales elementos de los prototipos se presentan en la Figura 1.

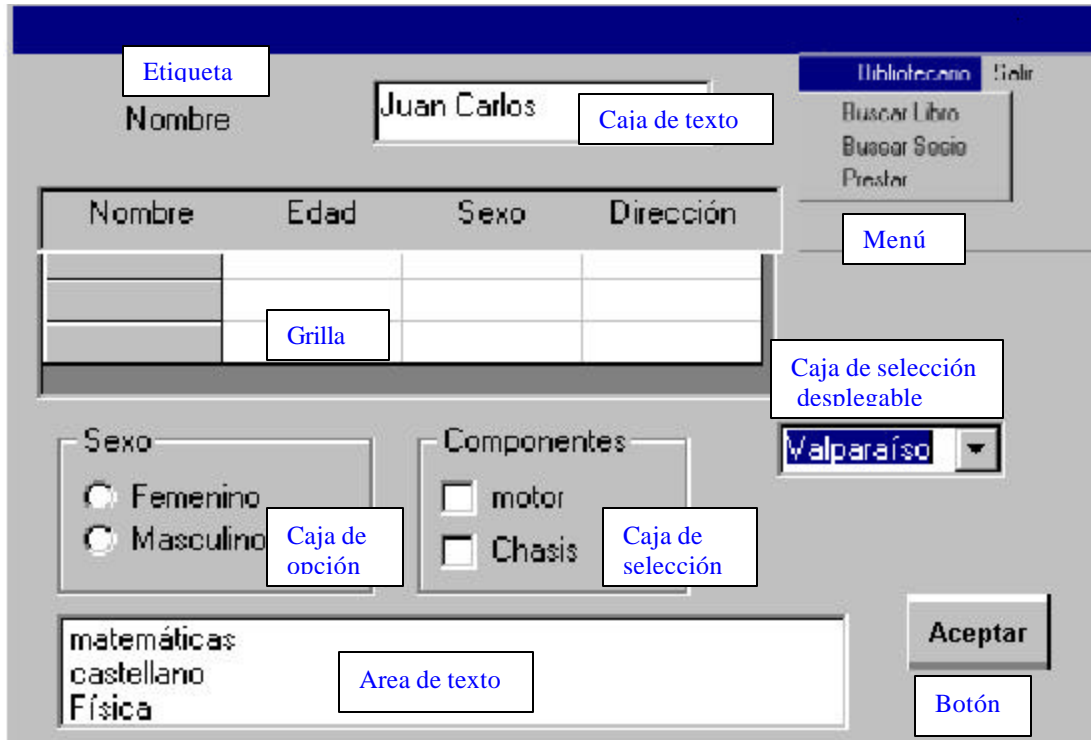


Figura 1: Principales elementos presentes en los prototipos.

Al disponer de un prototipo es posible asociar los elementos de éste con ciertos componentes UML, como son los atributos, clases, etc. Los aspectos principales de la metodología propuesta, que asocia los elementos del prototipo con los componentes de los diagramas UML, son descritos en la Tabla 2.

Luego de la aplicación de la metodología propuesta se debe realizar un chequeo de estos diagramas con la descripción del problema para lograr diagramas de mayor calidad. Finalmente a partir de estos diagramas se generan el resto de los diagramas UML que permiten desarrollar el software.

4. Aplicación de la metodología.

Para la aplicación de la metodología propuesta se han utilizado dos prototipos estándar creados cada uno por un equipo desarrollador, en ambos casos no se tienen antecedentes de requerimientos ni la descripción del problema, esto es realizado en forma intencional para evaluar la metodología propuesta. Debido a esto, es probable que los diagramas generados no representen fielmente la funcionalidad del sistema, pero si se espera que sean una buena aproximación y que luego de refinarlos con la descripción del problema representen toda la funcionalidad del sistema.

Prototipo	UML
Relación entre etiquetas y cajas de texto.	La etiqueta se considera como nombre de un atributo candidato.
Relación entre etiquetas y grillas.	Cada etiqueta se considera como nombre de un atributo candidato.
Relación entre etiquetas y cajas de opción.	Las etiquetas adyacentes a las cajas de opción son valores excluyentes de un atributo y la etiqueta que agrupa éstos es considerada nombre de un atributo candidato.
Relación entre etiquetas y cajas de selección.	Las etiquetas adyacentes a las cajas de selección son valores no excluyentes de un atributo y la etiqueta que agrupa estos es considerada nombre de un atributo candidato.
Relación entre etiquetas y cajas de selección desplegables.	La etiqueta es considerada nombre de un atributo candidato y el valor seleccionado en la caja desplegable es considerado el valor de dicho atributo.
Relación entre etiquetas y áreas de texto.	La etiqueta que encabeza el área de texto es considerada como nombre de un atributo candidato y el área de texto los posibles valores que puede tomar ese atributo.
Relación entre etiquetas y menús.	La etiqueta que encabeza el menú representa un actor y las etiquetas englobadas por éste representan las vistas de un actor.
Relación entre etiquetas y botones.	La etiqueta escrita en el botón representa una acción, pero si por medio de ésta se puede acceder a una vista de un actor, entonces se considera como un atributo candidato.

Tabla 2: Propuesta de asociación entre elementos del prototipo y componentes de diagramas UML.

El primer prototipo analizado fue para la empresa Rhona, del cual inicialmente no se tenían antecedentes y al aplicar la metodología se logró apreciar que se trataba de un sistema que manejaba las existencias de la empresa y podía realizar cotizaciones, ver estados de crédito, etc. Es decir, con la utilización de la metodología se pudieron identificar las clases, atributos, casos de uso, actores y las relaciones entre éstos, obteniendo resultados satisfactorios de la aplicación de la metodología [Ron01].

Luego se analizó otro prototipo, esta vez se trató de uno para la empresa Socima, del cual fue posible distinguir que se trataba de un sistema de inventarios, donde se agrupaban los productos en familias de productos, se mostraba cada producto con sus especificaciones, etc. Al igual que el caso anterior, la utilización de la metodología fue de gran ayuda, pues en un principio no se tenían antecedentes del sistema a realizar y luego de aplicar la metodología se obtuvo el diagrama de Casos de Uso y el diagrama de Clases [Ucv01].

Si bien dos casos de aplicación son pocos para probar una metodología, en ambos se obtuvieron resultados que cumplieron con las expectativas de ésta, es decir, se logró crear el diagrama de Casos de Uso y el diagrama de Clases sólo con el prototipo inicial de un sistema, reforzando la idea de que no sea desechable y siendo de gran ayuda para empezar con el análisis.

5. Conclusiones y trabajo futuro.

La metodología propuesta entrega una herramienta para que a partir de un prototipo inicial aceptado por el cliente se generen las bases de los diagramas UML. Esta generación de diagramas se centra más al de los Casos de Uso y al diagrama de Clases, ya que los diagramas de Casos de Uso describen el comportamiento del sistema y los diagramas de Clases describen la esencia del sistema. Además a partir de estos diagramas se pueden obtener los demás diagramas UML, como lo son diagramas de estados, diagramas de secuencia, diagramas de componentes, etc.; con los cuales se puede tener una vista clara y completa del sistema.

Esta metodología tiene como principal objetivo ser una herramienta más a disposición de los desarrolladores para sacarle el máximo provecho posible a un prototipo existente; de manera que en la eventualidad de que los desarrolladores no conozcan completamente los requerimientos funcionales de un sistema, puedan conocerlos a través de los diagramas UML generados con esta metodología a partir de este prototipo.

Luego de probar esta metodología con dos aplicaciones existentes, es posible darse cuenta de la facilidad de uso y rapidez que presenta para generar los diagramas de Casos de Uso y el diagrama Conceptual de Clases de dichas aplicaciones, lo cual representa un apoyo efectivo a los desarrolladores. Es importante también destacar que los diagramas generados con la metodología propuesta son coherentes con la descripción del sistema informático, es decir, los diagramas efectivamente representan la funcionalidad requerida por el sistema, aunque tal vez no representen toda la funcionalidad de éste, debido a que el prototipo no la especificara por completo.

Además, esta metodología es el resultado de más de dos años de trabajo e investigación acerca de metodologías orientadas al objeto, como OMT, de lo cual se concluye que la más importante hoy en día es UML, dado que es un lenguaje que unifica todos los conceptos utilizados en los desarrollos orientados al objeto, y aunque posee una gran cantidad de diagramas para representar la funcionalidad requerida en una aplicación, principalmente se basa en los diagramas de Casos de Uso y el diagrama de Clases. Es por esta razón que luego se ideó una metodología que ayudara en la generación de estos diagramas, obteniendo información de los prototipos de los sistemas informáticos, los cuales juegan un rol importante en la mayoría de los desarrollos de sistemas informáticos.

Finalmente, la forma más óptima de utilizar esta metodología es utilizando primeramente la Tabla 1, que ayuda a identificar los elementos de los casos de uso presentes en el prototipo, para luego generar el diagrama de Casos de Uso. Al probar la metodología con prototipos web, es posible identificar las relaciones entre Casos de Uso mediante links en las páginas web. Luego de generar los diagramas de Casos de Uso, se utiliza la Tabla 2, con la cual se identifican los elementos del diagrama de Clases presentes en el prototipo, con los que se genera el diagrama Conceptual de Clases.

6. Bibliografía.

[Cra99] Craig Larman, "UML y Patrones, Introducción al análisis y diseño orientado a objetos", Prentice Hall, México, 1999.

[Fow99] Martin Fowler con Kendall Scott, "UML gota a gota", Addison Wesley Longman, México, 1999.

[Iva00] Ivar Jacobson, Grady Booch, James Rumbaugh, "El proceso unificado de desarrollo de software", Pearson educación, Madrid, 2000.

[Ron01] Silvana Roncagliolo, Jorge Bozo & Beatriz Marin, “Propuesta de Método de Transformación de Prototipo a Bases de Diagramas UML”, IX Encuentro Chileno de Computación, Punta Arenas, Noviembre 2001

[Rum00] James Rumbaugh, Ivar Jacobson, Grady Booch, “El lenguaje unificado de modelado, Manual de referencia”, Pearson educación, Madrid, 2000.

[Ucv00] Escuela Ingeniería Informática - UCV , Informe Técnico, “MTP-UML”, Valparaíso-Chile, 2000.

[Ucv01] Escuela Ingeniería Informática - UCV , Informe Técnico, “ Propuesta de Método de Transformación de Prototipo a Bases de Diagramas UML y Casos de Aplicación de la Propuesta”, Valparaíso-Chile, 2001.