Evolución de Plantillas Genéricas para la descripción de Casos de Uso a Plantillas Genéricas para Análisis y Diseño

Ing. Marcela Daniele **AC. Daniel Romero**

Dpto. de Computación. Facultad: Ciencias Exactas, Físico-Qcas y Naturales. UNRC

{mdaniele,dromero}@exa.unrc.edu.ar

Resumen

Nuestra propuesta se basa en definir plantillas genéricas para la descripción de Casos de Uso, y a partir de ellas, definir plantillas para las etapas de Análisis y Diseño.

Introducción

Las metodologías de desarrollo de software y las técnicas de modelado fueron creadas para simplificar la complejidad del desarrollo de sistemas, y por lo tanto hay que utilizarlas en cualquier proceso de producción de software.

Tom DeMarco [14] propone la Ingeniería de Software basada en modelos, donde compara la construcción de un sistema de software con la construcción de cualquier otro tipo de sistemas ingenieriles, y por lo tanto propone la realización de modelos del sistema antes de la construcción del sistema mismo. De esta forma, el modelo de un sistema provee un medio de comunicación entre todos los participantes en el proyecto, cliente, usuarios y desarrolladores. Los métodos de desarrollo de software que se usan en la actualidad adoptaron la filosofía propuesta por este autor y, salvando las particularidades de cada uno, todos coinciden que la construcción de sistemas depende de modelos previamente definidos.

El Proceso Unificado [4] es un proceso de desarrollo de software que define *quién* está haciendo *qué*, *cuándo*, y *cómo* para construir o mejorar un producto de software. Las características principales que lo definen son: dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental.

En la primera etapa, denominada Captura de Requerimientos, esta metodología propone como artefacto crear una lista de los casos de uso y los actores del sistema. Un caso de uso es una funcionalidad del sistema que da valor agregado a algún actor o usuario del mismo. Luego cada

caso de uso requiere de una detallada descripción donde se indica: una precondición, una poscondición, un flujo de eventos principal y el flujo de eventos alternativo.

En las siguientes etapas, denominadas Análisis y Diseño, el Proceso Unificado tiene el propósito de obtener modelos del sistema más precisos y lograr una arquitectura estable, que permita realizar una correcta implementación del sistema, traduciendo estos modelos a un lenguaje de programación. La metodología propone que cada modelo construido en una etapa, debe basarse en los modelos construidos en la etapa anterior, y de esta manera asegura que el sistema obtenido responde a una construcción evolutiva e incremental.

En la etapa de Análisis se detectan las clases de análisis a partir de las descripciones de cada caso de uso y se construye un Diagrama de Clases UML [3][16] de análisis para cada uno. Además, se definen los escenarios más evidentes de cada caso de uso y se modelan Diagramas de Colaboración UML [3][16] lo cual permite identificar y definir las responsabilidades de las clases.

En la etapa de Diseño, cada clase obtenida en el análisis es transformada a una o más clases de diseño, y para cada clase de diseño se definen de manera completa y precisa sus atributos y métodos. Esta es la base o arquitectura estable requerida para avanzar hacia la implementación del sistema.

Nuestra Propuesta

Desde el año 1999, estudiamos el Proceso Unificado como metodología base para desarrollar sistemas. En el año 2004, desarrollamos el proyecto innovador "Definición y uso de Plantillas Genéricas para la descripción de Casos de Uso" [19]. El mencionado proyecto cubre la primera etapa, Captura de Requerimientos, donde propusimos las *plantillas genéricas para la descripción de casos de uso* en los problemas de: **inserción**, **eliminación**, **modificación** y **búsqueda** de elemento. En la actualidad, nos dedicamos a estudiar y analizar como evolucionar estas plantillas genéricas para cubrir las siguientes etapas del proceso de desarrollo. Definimos plantillas genéricas para la etapa de Análisis y Diseño, y de esta manera completamos las tres etapas fundamentales, Captura de Requerimientos, Análisis y Diseño, para lograr el modelo completo, con diferentes vistas, de una solución posible del problema a modelar.

Basados en que la definición de soluciones genéricas aporta una sustancial mejora en el accionar profesional cuando se deben plantear soluciones a problemas que requieren de igual tratamiento que otros ya presentados, analizados y resueltos. Nuestro principal propósito es que el modelo

construido represente una solución para el problema planteado y que pueda ser reutilizado en otros contextos.

Al igual que con las plantillas para la descripción de casos de uso, las plantillas para las etapas de análisis y diseño, permiten modelar problemas tales como: agregar, borrar, modificar o buscar un elemento. Y sin interesar cual fuere el elemento específico sino que basta con tener una plantilla que de la solución a cualquier problema de dicha naturaleza y simplemente reemplazar los atributos formales por los valores reales que se definen y plantean para cada caso de uso.

La figura 1 muestra una plantilla para la descripción de un caso de uso de Inserción de elemento y la figura 2 muestra un ejemplo de su instanciación.

Plantilla 1: Inserción de <<**elemento**>>

Parámetros:

- Elemento: ítem a ser insertado. Un ítem está compuesto por los atributos que lo definen: atributos clave y atributos.
- Atributos clave: las propiedades que identifican al elemento unívocamente.
- Atributos: propiedades que componen el elemento.
- Reglas de negocio ($\mathbf{r_1,..,r_n}$):

Nombre: Inserción de **<<elemento>>**.
Pre-condición: existe un **<<elemento>>** a ser ingresado.
Post-condición: **<<elemento>>** queda registrado en el sistema, o **<<elemento>>** ya estaba registrado en el sistema.
Descripción: realiza la inserción de un **<<elemento>>**, controlando la existencia del elemento en el sistema y el cumplimiento de las reglas del negocio ($\mathbf{r_1,..,r_n}$) asociadas al **<<elemento>>**.

ACTOR	SISTEMA
1. Ingresa **<<atributos clave>>** del **<<elemento>>**. 3. Ingresa el resto de los **<<atributos >>** del **<<elemento>>.**	**2.** Verifica existencia por **<<atributos clave>>.** 4. Verifica corrección de **<<atributos>>** ingresados. 5. Verifica reglas de negocio **<<$\mathbf{r_1,..,r_n}$>>** asociadas al caso de uso. 6. Realiza el alta del **<<elemento>>**

2.1. El sistema informa de la existencia del **<<elemento>>** identificado con **<<atributos clave>>.**
4.1. El sistema informa que al menos uno de los **<<atributos>>** ingresado no es correcto.
5.1. El sistema informa las reglas $\mathbf{r_i}$ que no se verifican (con $1 \leq i \leq n$).
Nota: En cualquier momento el usuario puede cancelar la ejecución del caso de uso.

Figura 1: Plantilla para la Inserción de un Elemento

CASO DE USO: Inserción de Cliente

Instancia: Plantilla 1 (Inserción de <<elemento>>)
Elemento: Cliente.
Atributos del Cliente.

ATRIBUTOS	CLAVE	VERIFICACION	ACCION
CUIT	SI	Debe ser NO NULO. El formato del cuit debe ser: 2 caracteres, un guión, 8 caracteres, un guión, 1 caracter.	
NOMBRE		Debe ser NO NULO.	
CONDICION DE IVA		El cliente puede ser: Responsable Inscripto, Responsable No Inscripto, Monotributo, Exento o Consumidor Final.	
MUTUAL		La mutual debe seleccionarse de las previamente cargadas en el sistema.	Include(Buscar Mutual)
CUENTA		Debe ser NO NULO. Se debe crear la cuenta del cliente.	Include(Inserción Cuenta)

Regla r1: Cada cliente debe tener una cuenta.

Figura 2: Instanciación de plantilla 1

Bibliografía

[1] "ANALISIS Y DISEÑO ORIENTADO A OBJETOS. Con Aplicaciones.". Grady Booch. Addison Wesley.

[2] "EL LENGUAJE DE PROGRAMACION JAVA". Ken Arnold – James Gosling. Addison Wesley/Domo.

[3] "EL LENGUAJE UNIFICADO DE MODELADO". Grady Booch, James Rumbaugh, Ivar Jacobson. Addison Wesley. 1999.

[4] "EL PROCESO UNIFICADO DE DESARROLLO DE SOFTWARE". Ivar Jacobson, Grady Booch, James Rumbaugh. Addison Wesley. 1999.

[5] "INGENIERIA DE SOFTWARE: Un Enfoque Práctico", Roger S. Pressman. Mc Graw Hill. 1997.

[6] "Pattern-Oriented Software Architecture, Volume 1: A System of Patterns", Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal. March 1997

[7] "Refactoring: Improving the Design of Existing Code" by Martin Fowler (Author), Kent Beck (Author), John Brant (Author), William Opdyke (Author), Don Roberts (Author).

[8] "SOFTWARE ENGINEERING", Ian Sommerville.Addisson Wesley.

[9] D.R., SANLOZ holonic – "Evaluación de Proyectos", 1998, 2000 http://www.geocities.com/Eureka/Office/4595/evalproy.html.

[10] IEEE: Standard Collection: Software Engineering, IEEE Standard 610.12-1990, IEEE 1993.

[11] Lilia Verónica Toranzost. Organización de estados iberoamericanos para la educación, la ciencia y la cultura – o.e.i. Evaluación de proyectos- Marco metodológico. Junio 2001. www.campus-oei.org/calidad/marco.PDF.

[12] Naur, P. y B. Randall(eds.) Software Engineering: A Report on a conference Sponsored by the NATO Science Committee, NATO, 1969.

[13] Sánchez, M. (2002). La investigación sobre el desarrollo y la enseñanza de las habilidades de pensamiento. *Revista Electrónica de Investigación Educativa 4,* (1). http://redie.ens.uabc.mx/vol4no1/contents-amestoy.html

[14] Tom DeMarco. Libro: Structured Analysis and System Specification, 1979.

[15] Zelkovitz, M.V., Shaw, A.C. y Gannon, J.D.: Principles of Software Engineering and Design. Prentice Hall, 1979.

[16] Object Management Group, 2000, OMG Unified Modeling Language Specification, Version 1.3, ad/00-03-01.

[17] "Design Patterns: Elements of Reusable Object-Oriented Software". Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Addison-Wesley. 1995.

[18] "Patterns in Java. Volume 1. A Catalog of Reusable Design Patterns Illustred with UML". Mark Grand. John Wiley & Sons Inc. 1998.

[19] Definición y uso de Plantillas Genéricas para la descripción de Casos de Uso. Evaluado y Aprobado como Proyecto de Innovación e Investigación para el Mejoramiento de la Enseñanza (PIIMEG 2004) por la Secretaría de Ciencia y Técnica y la Secretaría Académica de la Universidad Nacional de Río Cuarto. Resolución Rectoral Nº 302/04. Año 2004. Autores: Ing. Marcela Daniele, An. Nicolás Florio. An. Daniel Romero.