# Extendiendo las plantillas genéricas para la definición de casos de uso con un framework genérico distribuido

Marcela Daniele Paola Martellotto Daniel Romero

Departamento de Computación
Facultad de Ciencias Exactas, Fco-Qcas y Naturales
Universidad Nacional de Río Cuarto
{marcela,paola,dromero}@dc.exa.unrc.edu.ar

## Resumen

En el día a día, el ingeniero de software se enfrenta con problemas que pertenecen a diferentes contextos de aplicación pero presentan un comportamiento similar. Situaciones en las que el ingeniero debe esforzarse por plantear soluciones genéricas, que sean instanciadas y den solución a problemas específicos similares.

Los patrones de software representan soluciones reusables en problemas recurrentes, definidos como una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular.

Las especificación de las plantillas genéricas para la definición de casos de uso en problemas de inserción, eliminación, modificación y búsqueda de elemento representan un patrón de solución a ese tipo de problemas. Su principal propósito es que los alumnos aprendan a construir modelos que representen una solución para el problema planteado, y resulten en soluciones genéricas reusables.

Este trabajo propone la construcción de un framework genérico distribuido, que extiende la especificación de las plantillas genéricas para la definición de casos de uso, utilizando tecnología de servicios web y aprovechando sus ventajas provistas en sistemas distribuidos. Este framework facilita la inclusión de temas vinculados a patrones de diseño y tecnologías distribuidas, ayudando a los estudiantes, futuros ingenieros de software, la comprensión de dichos temas.

**Palabras Claves:** Plantillas Genéricas, Casos de Uso, Framework, Patrones de Diseño, Servicios Web.

## 1. Introducción

Cuando se afronta el desarrollo de un Sistema de Software, es fundamental no descuidar la idea básica de la Ingeniería de Software, que consiste

en observar al sistema como un producto complejo, y a su proceso de construcción como un trabajo ingenieril. Es decir, un proceso basado en metodologías, técnicas, teorías y herramientas, que requiere de una planificación adecuada, análisis, diseño, implementación, prueba y mantenimiento [11].

El modelo de un sistema provee un medio de comunicación entre todos los participantes en el proyecto, cliente, usuarios e ingenieros de software. En la propuesta de Tom DeMarco [7], define que la Ingeniería de Software está basada en modelos, y compara la construcción de un sistema de software con la construcción de cualquier otro tipo de sistemas ingenieriles. Los métodos de desarrollo de software que se usan en la actualidad adoptaron esta filosofía y, salvando las particularidades que cada uno presenta, la mayoría coinciden en que el desarrollo de sistemas de software se basa en la construcción de modelos, y en la evolución de dichos modelos, hasta completar todas las etapas en el ciclo de vida del desarrollo de un sistema de software.

Desde el año 1999, en un curso Análisis y Diseño de Sistemas, que los autores de este trabajo dictan en el tercer año de las carreras Analista en Computación y Licenciatura en Ciencias de la Computación, de la Universidad Nacional de Río Cuarto, se utiliza la metodología de desarrollo de software orientada a objetos y basada en modelos, denominada Proceso Unificado [1].

El curso de Análisis y Diseño de Sistemas, persigue como principal objetivo, lograr que los alumnos aprendan a construir modelos que representen soluciones óptimas, correctas y de calidad al problema planteado. Y no conformes con esto, se pretende que el alumno pueda razonar en la generalización de estas soluciones obtenidas, a problemas de la misma naturaleza pero aplicados en diferentes situaciones o contextos.

En [2] se propusieron las plantillas genéricas para la descripción de casos de uso en problemas de inserción, eliminación, modificación y búsqueda de elementos. En los años 2003 y 2004, se implementó la propuesta en el curso de Análisis y Diseño de Sistemas mencionado anteriormente, en donde se obtuvieron excelentes resultados, tanto en el curso como en el impacto en el accionar de los alumnos en asignaturas posteriores relacionadas [6], favoreciendo además al futuro accionar profesional. En el año 2005, se evolucionaron estas plantillas a las siguientes etapas del ciclo de desarrollo de software, obteniendo las plantillas genéricas para especificar los casos de uso en las etapas de análisis y diseño [3, 5]. En esta oportunidad, los resultados obtenidos fueron también muy alentadores [6].

En [4] se presentó un framework, utilizando patrones de diseño, que permite a los alumnos implementar los casos de uso genéricos descriptos mediante las plantillas, y teniendo presente los requisitos no funcionales deseados en todo sistema de software. En este trabajo se propone mejorar aquella propuesta [4], aportando a la construcción de un framework genérico distribuido que extiende la especificación de las plantillas genéricas para la definición de casos de uso, utilizando la tecnología de los servicios web y aprovechando las ventajas que la misma provee en sistemas distribuidos. Esto contribuye a facilitar la introducción y enseñanza de este tipo

de tecnologías distribuidas, en el curso de Análisis y Diseño de Sistemas.

Este trabajo está organizado de la siguiente manera. En la sección 2 se describe la propuesta y sus objetivos, en la sección 3 se expresan brevemente las principales características de los sistemas distribuidos y las arquitecturas más conocidas, además se justifica la selección de los servicios web para representar la arquitectura del framework genérico distribuido propuesto en este trabajo. En la sección 4 se exponen los patrones de diseño utilizados para la construcción del framework. En la sección 5 se describe el framework y se muestra gráficamente la arquitectura definida para su construcción. Por último, en la sección 6 se presentan las conclusiones y el trabajo futuro.

## 2. Descripción de la Propuesta

Los Patrones de Diseño [9] constituyen una de las innovaciones de mayor impacto sobre el desarrollo orientado a objetos, y su uso es cada vez más requerido por los diseñadores de software. Por esta razón, el uso de patrones de diseño se considera una habilidad básica que deben adquirir los estudiantes en Ciencias de la Computación.

Los beneficios de la utilización de patrones en la ingeniería de software son reconocidos por toda la comunidad informática. Los patrones ayudan a construir soluciones a problemas conocidos, sobre la experiencia colectiva de ingenieros de software más experimentados, y permiten promover buenas prácticas del diseño del software [10].

Las plantillas genéricas para la definición de casos de uso [2, 3] conlleva una importante mejora en el accionar de los alumnos cuando se requiere plantear soluciones a problemas que requieren de igual tratamiento que otros ya estudiados y resueltos. Por otro lado, a través de estas plantillas los alumnos despiertan la necesidad de definir sus propias plantillas para dar solución a los nuevos problemas que se le presenten de manera reiterada. El propósito es que los alumnos aprendan a construir modelos que representen una solución para el problema planteado y sirvan como soluciones genéricas a ser utilizadas en nuevos contextos.

Un framework es una estructura de soporte sobre la cual otro proyecto de software puede ser organizado y desarrollado. Es el esqueleto sobre el cual varios objetos son integrados para dar alguna solución específica.

El diseño conceptual del framework propuesto en este trabajo, es el resultado de la evolución de las plantillas genéricas para la definición de casos de uso, junto con la combinación de diferentes patrones de diseño.

En la estructura del framework se visualizan los patrones *Mediator*, *Data Transfer Object* (DTO) y *Registry*. El framework trabaja mediante una arquitectura cliente-servidor, utilizando servicios web para establecer la comunicación.

Un servicio web es una aplicación que puede ser descrita, publicada, localizada, e invocada a través de una red [14]. Los servicios web proveen esencialmente un medio estándar de comunicación entre diferentes aplicaciones de softwa-

re. Su uso en la Web se ha extendido rápidamente debido a la creciente necesidad de comunicación e interoperabilidad entre aplicaciones.

El framework propuesto aprovecha los beneficios de los servicios web para obtener una solución genérica distribuida en los problemas de inserción, eliminación, modificación y búsqueda.

Para la construcción del framework propuesto se tienen en cuenta los requisitos no funcionales deseables en la mayoría de los sistemas de software, entre los principales tenemos: (i) el desarrollo de sistemas distribuidos, (ii) el aprovechamiento de los recursos de hardware, (iii) la mejora en la velocidad de acceso y disponibilidad de los objetos del dominio y (iv) el reuso de componentes. Además, los objetivos planteados en torno a la propuesta pueden resumirse en: Fomentar el uso de una metodología de desarrollo de software a partir de proyectos reales.

- Enseñar al alumno a razonar y construir una solución genérica y reusable.
- Enseñar al alumno a instanciar problemas reales sobre el modelo genérico propuesto.
- Crear en el alumno la habilidad para seleccionar y utilizar convenientemente, tecnologías distribuidas en sus futuros proyectos.
- Disminuir la dificultad para comprender los diversos modelos de solución planteados para resolver el mismo tipo de problemas.
- Internalizar en el accionar del alumno, futuro Ingeniero de Software, buenas prácticas de diseño y favorecer su futuro accionar profesional.

# 3. Sistemas distribuidos y Servicios Web

Los sistemas distribuidos proveen los mecanismos necesarios para un manejo transparente de la comunicación, permitiendo que el programador se concentre en la lógica de la aplicación. Algunas arquitecturas propuestas para sistemas distribuidos son:

(i) Invocación remota de métodos (RMI). Es independiente del sistema operativo y sólo disponible para el lenguaje JAVA.

(ii) DCOM de Microsoft. Disponible para los sistemas operativos de la familia de Microsoft, y tiene soporte sobre varios lenguajes: Visual Basic, C++, C.

(iii) CORBA de la OMG. Independiente de plataforma y para varios lenguajes: Java, C, C++, Ada. Este tipo de sistemas requieren un alto grado de acoplamiento y una comunicación sincrónica.

(iv) Servicios Web. Son una arquitectura de computación distribuida en evolución que posibilitan que aplicaciones de diferentes plataformas tecnológicas puedan utilizar “servicios” de otras aplicación.

El framework propuesto en este trabajo, está basado sobre servicios Web, por considerar la mas independiente de todas las propuestas.

Un Servicio Web se beneficia de la especificación de XML para definir tanto su descripción, como los mensajes que recibe y produce al igual que en los servicios de registro del servicio.

Los servicios web presentan un bajo grado de acoplamiento y comunicación de tipo asincrónica. La principal ventaja de los servicios web es que resuelven el problema de la interoperabilidad de las tecnologías de objetos distribuidos, porque están basados en estándares abiertos aceptados por la industria en general, propuestos y administrados por la World Wide Web Consortium (W3C) [14]. Para representar un servicio web es necesario:

**WSDL** (Web Service Description Language), es el lenguaje utilizado para la descripción del servicio. Este formato describe la interfaz del componente (sus métodos y atributos) basado en XML [12].

**SOAP** (Simple Object Access Protocol), es el protocolo para la representación de los mensajes que permite que las aplicaciones interactúen con servicios web [13].

**HTTP** (Hiper Text Transport Protocol), es el protocolo de transporte de los mensajes por Internet. **UDDI** (Universal Description, Discovery and Integration), es un directorio de servicios web distribuido, basado en Web que permite que se listen, busquen y localicen.

# 4. Descripción de Patrones de Diseño usados para la construcción del Framework

Tres patrones fueron utilizados en este framework, el patrón *mediator* [9] para el manejo de las interfaces de usuario, el patrón *data transfer object* [8] para reducir el número de llamadas al servidor y el patrón *registry* [8] para acceder a los objetos persistentes.

## 4.1. Patrón *mediator*

El objetivo de este patrón es centralizar la lógica de cambio de estados de diferentes objetos, en un objeto administrador, permitiendo de esta manera un mejor reuso de los demás objetos. Da como resultado una implementación más cohesiva, y minimiza el acoplamiento entre los objetos. En nuestro framework se utiliza para coordinar los objetos de la interfaz gráfica. En la Figura 1 se presenta un esquema correspondiente a este patrón.

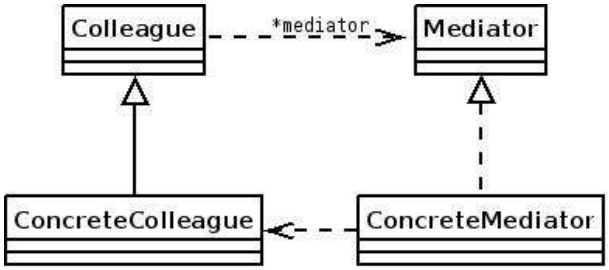


Figura 1: Esquema patrón *mediator*

## 4.2. Patrón *data transfer object*

Cuando se trabaja con interfaces remotas, cada invocación es costosa, y por lo tanto es necesario reducir el número de invocaciones, para ello, es necesario transferir mayor cantidad de información en cada una de ellas. En este contexto, el patrón DTO permite encapsular la información en la comunicación entre el servidor y el cliente. Usualmente es necesario tener un objeto assembler (armador) del lado del servidor para hacer la correspondencia entre la información a

transferir y los datos del dominio [1]. En la Figura 2 se presenta el esquema correspondiente a este patrón.

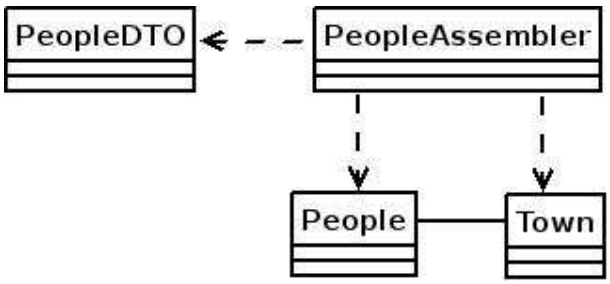


Figura 2: Esquema patrón *Data transfer object*

### 4.3. Patrón *registry*

Un registry es un objeto que permite acceder a objetos del dominio, valiéndose únicamente de un identificador de los mismos [1].

## 5. Descripción del Framework

Un framework es una estructura soporte definida en el cual otro proyecto de software puede ser desarrollado. El principal objetivo es resolver un problema complejo.

En este framework, los problemas abordados son: (i) utilizar correctamente los patrones de diseño en un contexto real, (ii) incorporar nuevas tecnologías en la práctica de los alumnos, y (iii) obtener una estructura que de soporte a un sistema distribuido para los problemas de inserción, eliminación, modificación y búsqueda de elemento. En la Figura 3 se puede ver la arquitectura del framework propuesto.

El framewrok está compuesto por un cliente y un servidor. En el cliente se tienen las clases correspondientes a la interfaz de usuario, allí se aplicó el patrón *mediator*. En el servidor se encuentra el patrón DTO para transformar los elementos del dominio (People y Town), y el patrón *registry* que da un acceso fácil al dominio para el *web service*. La comunicación con el servidor se hace por medio de la interfaz provista por el servicio web.

## 6. Conclusiones y trabajo futuro

El crecimiento tecnológico y la diversidad del mismo, hace muy difícil incorporar todos los nuevos conceptos a las currículas de las carreras, es por ello que es necesario dotar al alumno, fututo Ingeniero de Software, con las habilidades necesarias para adaptarse a estos cambios. También es importante buscar propuestas que permitan incluir nuevas tecnologías sin descuidar la enseñanza básica.

En este trabajo se presenta una propuesta que puede ser utilizada en cursos de Ingeniería de Software, para facilitar la comprensión y resolución de problemas que se presentan habitualmente y de forma reiterada en el desarrollo de sistemas de software. En particular, se presentó un framework genérico distribuido que implementa de manera distribuida los problemas de inserción, eliminación, modificación y búsqueda de elementos.

Esta propuesta fomenta en el alumno, la idea de que la utilización de patrones de diseño ayuda

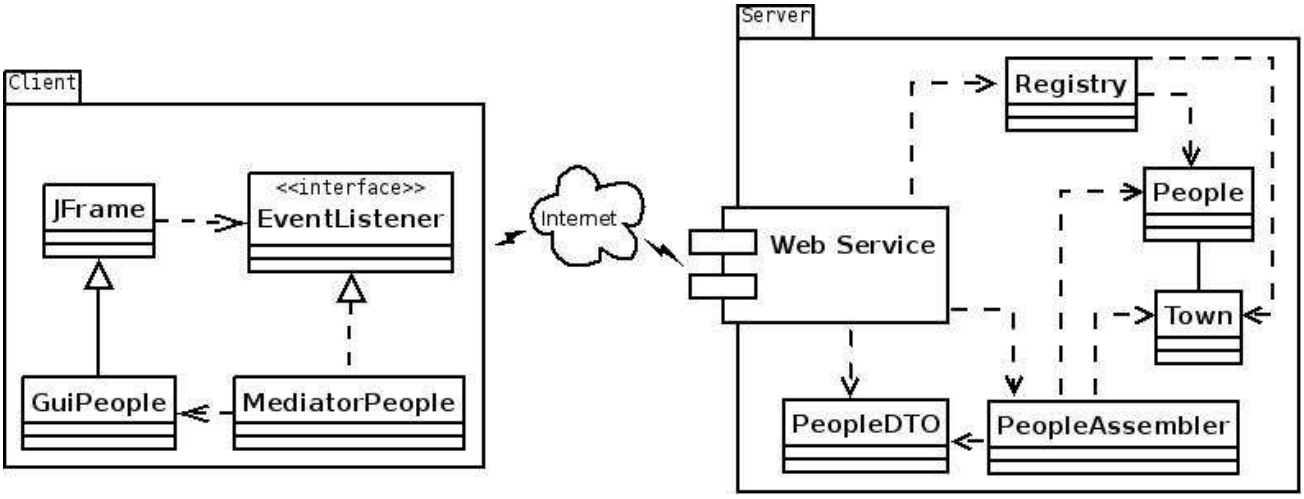


Figura 3: Arquitectura del Framework

a la comunicación con los demás ingenieros, a la estandarización, la flexibilidad y la robustez en la arquitectura del software. El alumno aprende a usar patrones de diseño de una forma adecuada para la resolución de problemas y es motivado para construir nuevas propuestas que den solución a nuevos problemas. Además, le ayuda a adoptar esta filosofía como una buena práctica en sus trabajos futuros.

Por otra parte, la inclusión de las tecnologías distribuidas a través del uso de servicios web para afrontar la construcción de este framework, posibilita incluir en los contenidos del curso Análisis y Diseño de Sistemas un tema tan importante y trascendente como este, que de otra manera no se contaría con el tiempo suficiente para abordarlo como tema específico. Además, aporta importantes beneficios tanto a cursos posteriores como al futuro accionar profesional de los alumnos, en su rol de ingenieros de software.

Como trabajo futuro, se encuentra en desarrollo una herramienta que utiliza este framework y permitirá agregar un modelo de datos, un conjunto de restricciones y un conjunto de preferencias de usuario dadas como entrada, en formato XML, y operará realizando inserción, eliminación, modificación y búsqueda de información de los datos del modelo, absolutamente de forma genérica. El desarrollo de esta herramienta resulta en el trabajo de tesis de un alumno de la carrera Licenciatura en Ciencias de la Computación de la UNRC.

# Referencias

[1] Grady Booch, Ivar Jacobson, and James Rumbaugh. *Proceso Unificado de Desarrollo de Software*. Addison Wesley, 2000.

[2] M. Daniele, N. Florio, and D. Romero. Definición y uso de plantillas genéricas para la descripción de casos de uso. Technical report, 2004. Proyecto de Innovación e Investigación para el Mejoramiento de la Enseñanza (PIIMEG 2004). Finaciado por la Secretaría de Ciencia y Técnica y la Secretaría Académica de la Universidad Nacional de Río Cuarto. RR N$^{o}$ 302/04.

[3] M. Daniele and D. Romero. Evolución de plantillas genéricas para la descripción de casos de uso a plantillas para el análisis y diseño. Technical report, 2005. Proyecto de Innovación e Investigación para el Mejoramiento de la Enseñanza (PIIMEG 2005). Finaciado por la Secretaría de Ciencia y Técnica y la Secretaría Académica de la Universidad Nacional de Río Cuarto. RR Nº 109-110/05.

[4] M. Daniele and D. Romero. Construcción de un framework para la enseñanza de patrones de diseño. In *Proc. of World Congress on Computer Science, Engineering and Technology Education. New Engineering to a New World WCCSETE 2006. Itanhaém, Santos, Brasil*, 2006.

[5] M. Daniele and D. Romero. La enseñanza de construcción de soluciones genéricas en un curso de análisis y diseño de sistemas. In *I Congreso de Tecnología en Educación y Educación en Tecnología (TE ET 06),La Plata, Argentina*, 2006.

[6] M. Daniele and D. Romero. Proyectos e informes finales del PIIMEG (2004, 2005, 2006), "Definición y Uso de Plantillas Genéricas para la descripción de Casos de Uso" y "Evolución de Plantillas Genéricas para la descripción de Casos de Uso a Plantillas Genéricas para Análisis y Diseño". Technical report, 2006. https://dc.exa.unrc.edu.ar/materias/sistemas/archivos/PIIMEG/.

[7] Tom DeMarco. *Structured Analysis and System Specification*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1979.

[8] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[9] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Professional, 1995.

[10] Michael Kircher and Prashant Jain. *Pattern-Oriented Software Architecture: Patterns for Resource Management*. John Wiley & Sons, 2004.

[11] R S Pressman. *Software engineering: a practitioner's approach (5th ed.)*. McGraw-Hill, Inc., New York, NY, USA, 2001.

[12] World Wide Web Consortium (W3C). Extensible Markup Language (XML) 1.0, second edition, 1999. Available at: `http://www.w3.org/XML`.

[13] World Wide Web Consortium (W3C). Simple Object Access Protocol (SOAP) 1.1, 2000. Available at: `http://www.w3.org/TR/soap/`.

[14] World Wide Web Consortium (W3C). Web Services Activity, 2002. Available at: `http://www.w3.org/2002/ws/`.

# Bibliografía consultada

- Booch Grady, Rumbaugh James and Jacobson Ivar. "The Unified Modeling Language". Addison Wesley. 1999..
- Sun Developers Network. "Java Remote Method Invocation (Java RMI)". http://java.sun.com/products/jdk/rmi/
- Carlo Ghezzi, Mehdi Jazayeri, Dino Mandrioli. Fundamentals of Software Engineering. Prentice Hall, 1991.
- Ian Sommerville - Software Engineering. Addison-Wesley, 5ta Edición. 1996.
- Bertrand Meyer. Object Oriented Software Construction. Prentice Hall. 1997.
- G. Gordon Schulmeyer, James I. McManus - Handbook of Software Quality Assurance. Prentice Hall, 3ta Edición. 1999.
- Craig Larman, - "Appliyng UML and Patterns"- Prentice Hall. Third Edition, 2005.
- Jackson M. Software Requirement and Specifications. Addison Wesley, 1995.

- Mark Grand. Patterns in Java. Volume 1 y 2. - A Catalog of Reusable Design Patterns Illustrated with UML- Wiley Computer Publishing, 1998.
- 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.
- Naur, P. y B. Randall(eds.). Software Engineering: A Report on a conference Sponsored by the NATO Science Committee, NATO, 1969.
- 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
- Zelkovitz, M.V., Shaw, A.C. y Gannon, J.D.: Principles of Software Engineering and Design. Prentice Hall, 1979.
- Object Management Group, 2000, OMG Unified Modeling Language Specification, Version 1.3, ad/00-03-01.
- Paul Jorgensen. Software Testing: A Craftsman's Approach. CRC Press, 1995.