Técnicas de reuso dentro de la Ingeniería del Dominio

Laura Felice; Daniel Riesco*

INTIA. Facultad de Ciencias Exactas. UNCPBA
lfelice@exa.unicen.edu.ar

* Departamento de Informática.
Facultad de Ciencias Físico Matemáticas y Naturales. UNSL
driesco@unsl.edu.ar

1. Introducción

La mayoría de los métodos de desarrollo, incluyendo los métodos orientados a objetos, no incluyen actividades y técnicas de reuso bien definidas. A menos que las actividades del reuso sean parte integral y natural del proceso de desarrollo es probable que el reuso no se practique en los proyectos de software.

El reuso debe ser planificado con el comienzo del planning en los estados muy tempranos del proceso de desarrollo. Dejar el reuso como una actividad del final del proceso es un error común. Una clave para maximizar los beneficios del reuso es pensar en 'grande' Pensar ´grande´, en términos de 'grandes' componentes, es decir no simplemente reusar una función sino reusar subsistemas completos y arquitecturas enteras, así como también significa pensar en términos del reuso en altos niveles de abstracciones de software, por ejemplo reuso de diseño en vez de reuso de código. Generalmente, reusar código es lo primero en que se piensa y lo que se practica.

Es importante considerar dos aspectos de reuso para ser incorporados dentro del proceso de desarrollo a fin de soportar la práctica del reuso de software:

- actividades para crear o adquirir componentes reusables,
- actividades para usar componentes reusables como bloques construidos en la creación de nuevos sistemas

Estas actividades tienen que ser soportadas por técnicas practicadas habitualmente en los métodos de desarrollo. El análisis de las componentes comunes es una de esas técnicas, éste es considerado parte del *análisis del Dominio*. El análisis del dominio es la identificación y explotación de componentes comunes a través del sistema relacionado y es esencial para la práctica del reuso de software.

Teniendo en cuenta estas cuestiones, y que en el proceso de desarrollo propuesto en RAISE [7] no se hace referencia explícita a la *reusabilidad* ni tampoco se da ninguna técnica para aplicarlo, en este proyecto de investigación se propone realizar un análisis de dominios a fin de definir un modelo conceptual de dominios reusables para ser incorporado al método de desarrollo.

Relacionado con el método RAISE es importante destacar el trabajo de Beltaifa y Moore [1] donde se propone una infraestructura para soportar reuso mejorando tanto la facilidad como la eficiencia en el reuso de componentes de software. La principal diferencia con el enfoque de nuestro proyecto es la definición de un proceso integrado definido en todos los estados de desarrollo.

Este artículo está organizado de la siguiente manera: en la sección 2 se describe el Análisis del dominio, en la 3 se aborda con el método en cuestión, y en la sección 4 se describe en forma simplificada el modelo conceptual propuesto para los dominios reusables. Por último en la sección 5 se presentan conclusiones y el alcance del proyecto.

2. Análisis del dominio

La definición de dominio asocia los items de información del mundo real como cuerpos de información con relaciones profundas y comprensivas entre ellos con respecto a alguna clase de problemas.

Esta definición general cubre dos perspectivas: dominio como una colección de problemas y dominio como una colección de aplicaciones futuras o existentes. Desde el primer punto de vista, los problemas son el objeto primario de estudio y los sistemas son el objeto secundario. Desde el segundo punto de vista, los sistemas son el objeto primario de estudio. Esta distinción afecta profundamente acerca de cómo razonamos con respecto al análisis del dominio. Para los que adoptan el primer punto de vista, el resultado del análisis es una teoría de los problemas en el dominio. La teoría debe tener algún poder predictivo y debe pasar por una validación. Para aquellos que adoptan el segundo punto de vista, el resultado del análisis del dominio es una taxonomía de componentes del sistema que hacen explícita las características en común entre los sistemas estudiados.

3. Raise y el paradigma del Tríptico

El método Raise se refiere al paradigma del desarrollo de software como el paradigma del Tríptico. El paradigma del Tríptico [2] es un paradigma de desarrollo riguroso de software, que descompone a la Ingeniería de Software en tres grandes fases: Ingeniería de Dominio, Ingeniería de Requisitos y Diseño de Software.

1. *Ingeniería de Dominio:* Antes de desarrollar software es claro que se debe determinar qué se quiere hacer, es decir determinar los requisitos. Estos requisitos involucran componentes, acciones y comportamientos del dominio del cliente o dominio de aplicación. Además, generalmente se expresan en función de términos que residen en el dominio. La Ingeniería del dominio analiza el dominio de aplicación y construye modelos de tal dominio. Los modelos pueden ser descriptos formal e informalmente.
   Las características en común que comparten una o mas clases de problemas tienen que ser especificables. El dominio tendrá al menos un área de aplicación que lo usa y posee un conjunto de reglas de decisión que ayudan al analista a decidir si un área de aplicación debería ser incluida en el dominio.
2. *Ingeniería de Requisitos:* los requisitos establecen qué se espera del software. La Ingeniería de Requisitos analiza las expectativas del cliente y construye modelos de éstas. Los requisitos se describen formal e informalmente. Los modelos no describen cómo funciona el software sino solamente qué debe ser entregado. Las descripciones de requisitos también son validadas por los stake-holders.
3. *Diseño de Software:* El objetivo es desarrollar, para el cliente, un sistema o paquete de software que satisfaga los requisitos. Comprende varios pasos de desarrollo, comenzando desde niveles abstractos hasta niveles más concretos asociados con código ejecutable:
   a. Arquitectura de Software: La arquitectura del sistema implementa los requisitos de interfaz y del dominio, y es lo que el usuario puede observar mientras usa el software. Esas interfaces observables externamente reflejan decisiones de diseño que determinan las estructuras de datos, funciones, eventos y procesos.
   b. Organización del Programa: Es una especificación en la cual, además del comportamiento de la interfaz visiblemente observable, se especifica la estructura

interna del software. Determina aspectos que pueden haberse dejado abstractos aún por la arquitectura de software.

Pasos posteriores del diseño de software concluyen con:

c. Pasos de refinamiento: Transforma estructuras de datos y operaciones abstractas en otras más concretas.
d. Código

Nuestra propuesta consiste en agregar a la Ingeniería del dominio de este paradigma un modelo de reuso basado en el segundo punto de vista nombrado anteriormente, donde el resultado del análisis del dominio es una taxonomía de componentes del sistema que hacen explícita las características en común entre los sistemas estudiados.

La figura 1 muestra una Infraestructura actual inmersa en un Dominio utilizando un conjunto de artefactos reusables organizados según el modelo RC (Componentes reusables). Este modelo describe componentes reusables agrupadas como clases de objetos, y un proceso transformacional con reuso desde especificaciones RSL [6] a código [5]. El concepto y ejemplos de diversas infraestructuras pueden ser encontrados en forma detallada en [3].

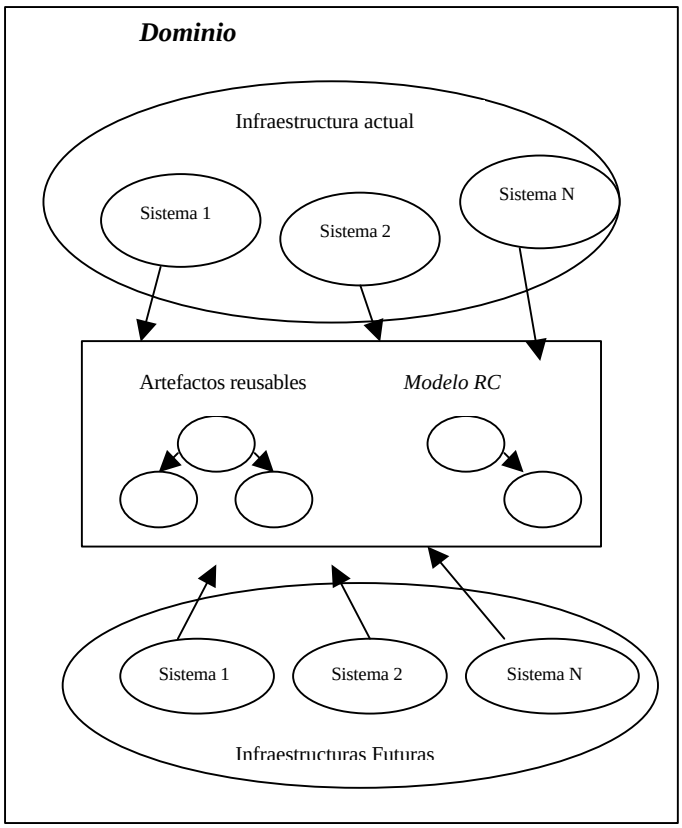

Figura 1. Dominio reusable

El modelo *component-based* es muy común en la práctica del reuso [4]. Trabajar con dominios permite agrupar artefactos reusables en el dominio como un aggregate que representa una realización de aplicaciones en el dominio. Llamaremos a ese aggregate un *dominio reusable.*

El dominio reusable representado en la figura 1 está limitado a sistemas dentro de ese dominio pertenecientes a cierta Infraestructura. La idea es estudiar una variedad de dominios relacionados e identificar sistemas reusables a través de esa colección de dominios.

4. Modelo conceptual del dominio reusable MCDr

Dependiendo del punto de vista adoptado (donde los sistemas o infraestructuras son el objeto primario), el modelo de un dominio es, tanto:

- un sistema formal de (i) items, (ii) relaciones entre términos (iii) reglas para componer términos en expresiones; (iv) reglas para razonar usando aquellos términos; (v) reglas para mapping desde items en un dominio de problemas a expresiones en el modelo a items en el dominio del problema, o
- una definición de las entidades, operaciones, eventos y relaciones entre las cosas en común o regularidades en un dominio, junto con una clasificación de estos.

Definimos al modelo conceptual del dominio reusable (simplificado en este artículo) como una tupla MCDr = $<\Sigma c, Ax, I>$, donde:

- Σc es la descripción de la componente. Esta descripción consiste de una *signatura* de componente (describiendo el vocabulario específico y relevante para la descripción de la componente): $\Sigma_c=(S_c, \Omega c, <=_c)$ donde

  Sc es un conjunto finito de sorts de datos,

  Ωc es una familia de conjuntos de operaciones de datos

  $<=_c \subseteq Sc \times Sc$ es una relación binaria sobre el conjunto de sorts de datos, tal que (Sc , <=c) es un orden parcial.

- Ax es un conjunto de reglas o axiomas que determinan la manera en que las componentes están conectadas, sus dependencias y la forma de interactuar con otras, así como las propiedades internas de la componente.

- I es una interface que maneja las comunicaciones entre un dominio reusable, que debe consistir de:
  - una taxonomía incluyendo: palabras claves, terminología, patterns, algoritmos.
  - mecanismos de comunicación y control: dependiendo del dominio, representa la comunicación dentro de un sub-dominio reusable y el control, en especial para dominios complejos las dependencias que existen entre las componentes. Esto se define por medio de invariantes, pre y post condiciones.

5. Conclusiones

El objetivo del proyecto de investigación en el cual se encuentra inmerso este trabajo, es proveer al método RAISE de una técnica de reusabilidad para todos los estados del desarrollo. Particularmente en este artículo, se presenta el análisis de dominios a fin de definir un modelo conceptual de dominios reusables para ser incorporado al método. Trabajos relacionados con este proyecto se refieren a la definición de un modelo RC para la descripción de componentes reusables y un proceso transformacional con reuso desde especificaciones RSL a código. Se ha trabajado en la clasificación y selección de componentes en infraestructuras concretas.

Referencias

[1] Beltaifa, R. ; Moore, R: ´A Software Reuse Infrastructure for an Efficient Reuse Practice´. Technical report 230. UNU/IIST. Macau. 2001. www.iist.unu.edu.

[2] Bjorner, Dines. "Software Engineering: A New Approach". Lecture Notes. Technical University of Denmark. 2000. www.it.dtu.dk/~db.

[3] Bjorner, Dines. "Informatics Models of Infrastructure Domains". CSIT´01 Yerevan, Armenia, 17-21 Setiembre 2001

[4] Brown, A; Wallnau, D " Engineering of Component-Based Systems". Component Based Software Engineering. IEEE Computer Society. Ed: Alan Brown. Carnegie Mellon University.

[5] Felice, L; Riesco, D "A rigorous model for raise specifications reusability" Capítulo 6 ´Practicing Software Engineering´ editado por Dr. Joan Peckham and Dr. Scott J. Lloyd. pp 63-81. Idea Group Publishing, 2003

[6] George, C; P. Haff, K. Havelund, A. Haxthaunsen, R. Milne, C. Nielsen, S. Prehn, K. Ritter. "The RAISE Specification Language"; Prentice Hall. 1992

[7] George, C., Haxthausen, A., Hughes, S., Milne, R., Prehn, S., Pedersen, J; "The RAISE Development Method"; Prentice Hall. 1995