Una Aproximación a la Estimación Automática del Tamaño Funcional

Marcela Daniele **Paola Martellotto** **Daniel Romero**
Departamento de Computación,
Universidad Nacional de Río Cuarto
Enlace Ruta 8 y 36 Km. 601,
C.P. X5804BYA, Río Cuarto, Córdoba, Argentina.
{mdaniele, pmartellotto, dromero}@exa.unrc.edu.ar

Abstract

La estimación del tamaño del software es una actividad en crecimiento dentro de la ingeniería del software. Contar con herramientas automáticas que permitan realizarla es uno de los desafíos más pujantes en este sentido. En el desarrollo de los sistemas es común reutilizar la lógica de las soluciones que se consideran correctas en diferentes contextos.

En este trabajo presentamos cómo se puede obtener una estimación del tamaño de un sistema de software utilizando técnicas de estimación funcional, analizando la lógica que es compartida entre las diferentes descripciones de casos de usos que resuelven problemas similares. Presentamos una aproximación hacia la automatización de la estimación funcional y los pasos que consideramos necesarios para la construcción de una herramienta que dé soporte a esta automatización.

1. Introducción

La medición del tamaño del software es actualmente un medio esencial para realizar las estimaciones oportunas de diversos indicadores necesarios para el desarrollo de productos de software. De este modo la precisión de los instrumentos de medición juega un rol importante porque se construye una herramienta de soporte para los directores de proyectos de desarrollo de software[4]. Existe una gran cantidad de trabajos de investigación e instituciones dedicadas a la temática de la medición del software, entre ellos [7, 8, 10, 3]. Estos trabajos presentan procedimientos de medición, y métodos de desarrollo de procesos de medición, pero entre ellos no hemos encontrado que esté incorporada la lógica de problemas resueltos previamente.

En [4] se describe el procedimiento para la realización de la estimación del tamaño funcional de un sistema orientado a objetos. En [5] presentan cómo de manera genérica se puede realizar el modelo de requisitos, análisis y diseño para un componente de un sistema de software.

En este trabajo, presentamos como es posible incorporar la lógica de problemas comunes, dentro de un procedimiento de medición y, de esta manera, reducir el esfuerzo y la información necesaria para realizar la estimación de un sistema de software.

Básicamente, nuestra aproximación aplica un procedimiento de estimación funcional a la descripción genérica de problemas comunes, dejando de forma paramétrica los pasos y valores que son específicos del dominio del problema. Esto nos permite obtener un procedimiento de medición que sea dependiente sólo del modelo de dominio. Además, describimos cómo es posible construir una herramienta de estimación automática del tamaño funcional de un sistema de software, mediante la utilización de herramientas de modelado.

Este trabajo está organizado de la siguiente manera. En la Sección 2, se describen las plantillas genéricas para la descripción de casos de uso. En la Sección 3, se delinean los pasos para un procedimiento de medición de tamaño funcional. En la Sección 4, presentamos nuestra propuesta de unificación del procedimiento de medición con las plantillas genéricas. En la Sección 5, se describe cómo se puede automatizar nuestra propuesta. Un caso de estudio es descripto en la Sección 6. Finalmente, en la Sección 7 se dan unas consideraciones a tener presentes en este trabajo, y en la Sección 8 tenemos las conclusiones.

2. Plantillas de Casos de Uso

En ingeniería de software, la resolución de problemas requiere de un trabajo ingenieril. Existen problemas que, si bien pertenecen a diferentes contextos de aplicación, responden a un comportamiento similar en todos los casos. En estas situaciones el ingeniero de software trata de encontrar una solución estándar o genérica, que pueda ser aplicada

en la resolución de problemas que requieren de igual tratamiento que otros ya estudiados y resueltos.

En esta línea, en [5] definen las denominadas *Plantillas Genéricas para la definición de Casos de Uso*, la idea principal es dar de una manera genérica la descripción de *casos de usos del sistema*, *el análisis* y *el diseño*, de los problemas de *inserción*, *eliminación*, *modificación* y *búsqueda de elementos*. La metodología que utilizaron es la definida por el *Proceso Unificado* (*PU*)[9]. El *PU* define quién está haciendo qué, cuándo hacerlo, y cómo construir o mejorar un producto de software. Es una guía para todos los participantes del proyecto: clientes, usuarios, desarrolladores, directivos, que ordena las actividades y dirige las tareas de cada uno y del equipo como un todo. Especifica los artefactos que deben desarrollarse en cada una de las etapas del desarrollo y ofrece criterios para el control y la medición, además de reducir riesgos y hacer el proyecto más predecible.

El *PU* recomienda utilizar *UML* (Unified Modeling Language)[11] como medio de expresión de los diferentes modelos que se crean durante las etapas del desarrollo. UML es un lenguaje estándar de modelado que permite visualizar, especificar, construir y documentar los artefactos de un producto de software.

Lo que se hizo en [5] fue describir la lógica de la solución para los problemas de *inserción*, *eliminación*, *modificación* y *búsqueda de elementos*, independientemente de cuál sea la información que se esté manipulando; de esta manera pueden decir que "verificar corrección de los ≪atributos≫ingresados" es una acción que se debe hacer independientemente de que se ingresen los datos de una persona, un diagnóstico o un servicio (ver más adelante la Figura 2). En la Figura 1 podemos ver el diagrama de *Casos de uso* correspondiente a las funcionalidades mencionadas, y las relaciones entre ellas. Como consecuencia del proceso de desarrollo y de la lógica descripta para estas funcionalidades en [5] podemos encontrar la evolución que va desde la descripción de cada uno de los *casos de uso* hasta llegar a la etapa de diseño. En la Figura 3 vemos el diagrama de secuencia, en la etapa de diseño, correspondiente al caso de uso *inserción de un elemento*.

En resumen, en [5] podemos encontrar una aproximación de cómo se puede modelar un problema desconociendo el dominio de aplicación del mismo, y que esta solución sirva como esqueleto a completar en las implementaciones reales.

3. Procedimiento de medición de tamaño funcional

En [4], se presenta un procedimiento de medición de tamaño funcional para sistemas orientados a objetos, el cual es una instanciación del método de medición de tamaño funcional COSMIC-FFP[3]. Este método es aprobado como

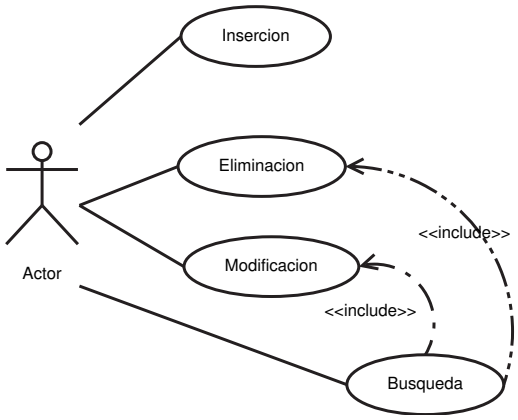

Figura 1. Diagrama de Casos de Uso

estándar por la ISO/IEC[8] y considerado como un método de segunda generación. El procedimiento [4] utiliza el *Modelo de Procesos de Medición* propuesto en [10] que consta de cuatro etapas: Diseño del procedimiento, Aplicación de dicho procedimiento, Análisis de los resultados y Exploración de los resultados. A continuación, describiremos brevemente el diseño y la aplicación del *Procedimiento de medición de tamaño funcional* dado en [4].

3.1. Diseño del procedimiento de medición

Para la definición del diseño del procedimiento de medición, lo primero que los autores definen es una clasificación de los casos de uso del sistema, luego distinguen entre diferentes tipos de mensajes en un diagrama de secuencia y por último definen una serie de reglas que permitirán la aplicación del procedimiento (ver 3.2).

La clasificación de los casos de uso se hace entre *primarios* y *secundarios*. Los *primarios* representan las funcionalidades más importantes, y los *secundarios* son los casos de uso que surgen al descomponer la complejidad de los casos de uso primarios mediante relaciones de "*extends*" e "*include*".

Para la descripción de los casos de usos, los autores utilizan *diagramas de secuencias*, y se distinguen cuatro tipos de mensajes que se envían entre los objetos y cada uno tiene un *estereotipo*. Estos mensajes son:

- Mensaje de Señal: (≪signal≫) representan una interacción entre el *actor* y el *sistema*.
- Mensaje de Servicio: (≪service≫) representan modificaciones del estado de la clase receptora.
- Mensaje de Consulta: (≪query≫) representan la consulta sobre el estado de un objeto o población de la clase.

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

Parámetros:

Elemento: ítem a ser insertado. Un ítem está compuesto por atributos clave y atributos.
Atributos clave: las propiedades que identifican al elemento unívocamente.
Atributos: propiedades que componen el elemento.
Reglas de negocio ($r_1...r_n$): indicar las reglas que deben ser verificadas en el caso de uso.

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.
Descripción: realiza la inserción de un <<elemento>>, controlando la existencia del elemento en el sistema y el cumplimiento de las reglas del negocio ($r_1...r_n$) asociadas al <<elemento>>.
Actor: nombre de cada uno de los actores que interactúan con el caso de uso.

FLUJO DE EVENTOS PRINCIPAL	
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 <<$r_1...r_n$>> asociadas al caso de uso. 6. Realiza el alta del <<elemento>>
FLUJO DE EVENTOS ALTERNATIVO	
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[1]. 5.1. El sistema informa las reglas r_i que no se verifican (con $1 \leq i \leq n$).	

Figura 2. Plantilla de Descripción del Caso de Uso Inserción de Elemento

- Mensaje de Conexión: (≪connect≫) representan mensajes propios de la relación estructural entre objetos.

Se definen tres tipos de reglas de procedimiento:

- Reglas de representación: Establecen una correspondencia entre las primitivas del *modelo de requisitos* de OO-Method[7] y los conceptos del metamodelo de COSMIC-FFP[3]. Estas reglas se agrupan en *usuarios*, *frontera*, *procesos funcionales*, *grupo de datos*, *atributos de datos* y *movimientos de datos* (entrada, lectura, escritura y salida).

- Reglas de duplicidad: se definen para evitar la duplicidad en la medición de los movimientos.

- Reglas de medición: Para obtener un valor cuantitativo se asigna un valor numérico de 1*Cfsu* (COSMIC Functional Size Unit) a cada movimiento de dato, y se definen las reglas que permiten obtener el tamaño funcional de cada proceso funcional.

De esta manera, describimos las generalidades del diseño del procedimiento de medición del tamaño funcional dado en [4].

3.2. Aplicación del procedimiento de medición

Para la aplicación del procedimiento hay que seguir tres pasos, éstos son: documentación del software, construcción del modelo de software y aplicación de las reglas de asignación numérica.

- *Documentación del software*: Identificar los casos de uso del sistema y describirlos mediante diagramas de secuencia. De esta manera, se construye el modelo de requisitos del sistema.

- *Construcción del modelo de software*: Identificar los elementos relevantes del modelo de requisitos de acuerdo a las reglas de procedimiento.

- *Aplicación de las reglas de asignación numérica*: Cuantificar mediante las reglas de la función de medición, para obtener el tamaño funcional del sistema.

De esta manera presentamos los conceptos necesarios para estimar el tamaño funcional de los sistemas orientados a objetos en una etapa temprana del proceso de desarrollo.

4. Propuesta

En la Sección 2, delineamos una manera de describir de forma genérica los problemas de inserción, eliminación,

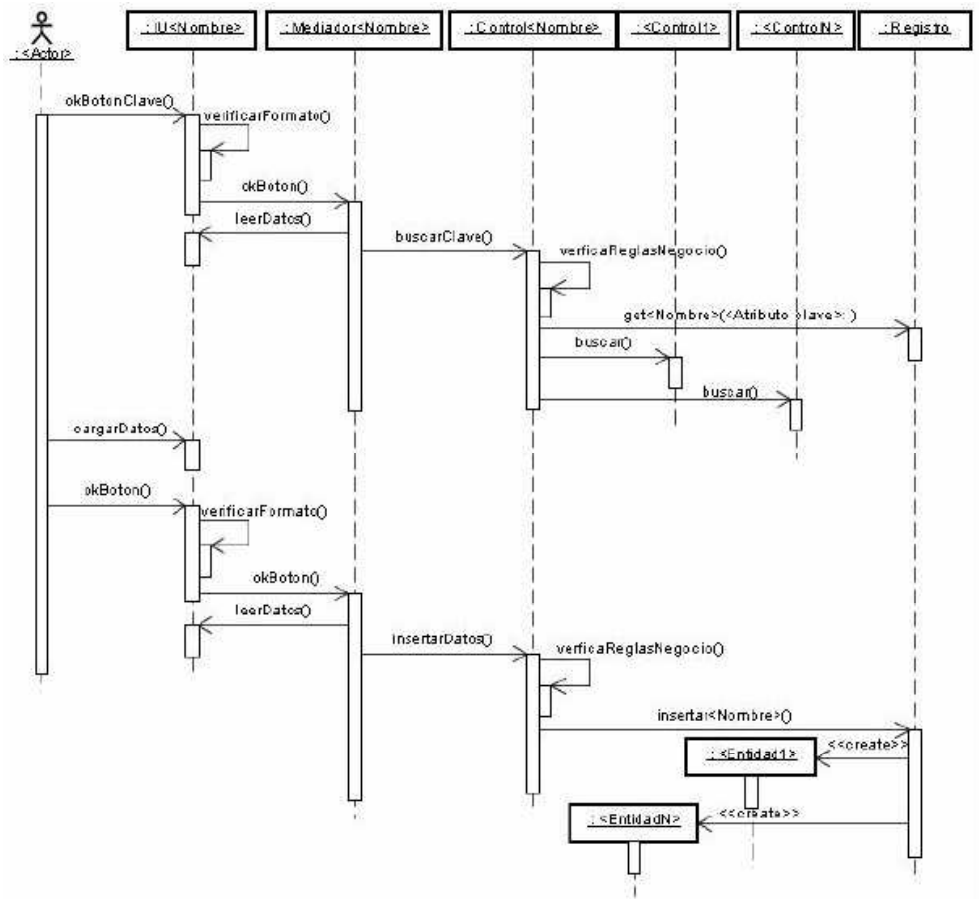

Figura 3. Diagrama de Secuencia para el Caso de Uso Inserción de Elemento

modificación y búsqueda de elementos. Esto nos permitió dar la solución independientemente del dominio de aplicación.

En la Sección 3 describimos los detalles de un procedimiento de medición de tamaño funcional para sistemas orientado a objetos, y vimos que la aplicación del procedimiento está dividida en tres etapas consecutivas.

En este trabajo, nuestra propuesta se centra en aplicar el *proceso de medición* [4] a las *plantillas genéricas* de [5], y de esta manera, obtener un proceso de medición genérico. Note que no es posible obtener una estimación final debido a la falta de información de la estructura del dominio, pero si es posible utilizar la lógica descripta para conocer parte de la estimación y obtener como resultado una métrica dependiente de la estructura del dominio de aplicación, métrica que, cuando tengamos el dominio real, nos servirá para estimar el tamaño funcional de nuestro sistema.

4.1. Aplicación del proceso de medición

Para aplicar el proceso de medición, seguiremos los tres pasos descriptos en [4], ellos son: documentación del software, construcción del modelo de software y aplicación de las reglas de asignación numérica.

Paso 1: Documentación del software

Este paso implica obtener los requisitos funcionales de usuario del sistema. En las *plantillas genéricas*, esto es dado por el modelo de casos de uso del sistema. En la Figura 1 vemos el diagrama de casos de uso correspondiente a *Inserción*, *Eliminación*, *Modificación* y *Búsqueda* de elemento.

Paso 2: Construcción del modelo de software

En este paso, se identifican los elementos relevantes del modelo de requisitos para la construcción del modelo de software a medir. Lo primero, es clasificar los casos de uso *inserción*, *modificación* y *eliminación* como *primarios*, y el caso de uso *búsqueda* como *secundario* (esto es debido a que el caso de uso búsqueda es incluido por los demás casos de uso). Luego, lo que hacemos es estereotipar los mensajes de acuerdo a las reglas de representación dadas (ver Sección 3.1). En la Figura 4 se muestra el diagrama de secuencia pa-

ra el caso de uso *inserción*, con los mensajes estereotipados. Note que, desconociendo de qué tipo es el elemento a insertar, esto no cambia el tipo de los mensajes que participan en la lógica del caso de uso.

Paso 3: Aplicación de las reglas de asignación numéricass

Este último paso, implica cuantificar los movimientos de datos identificados en el Paso 2. Para ello, aplicamos las reglas de duplicidad y de medición dadas en la Sección 3.1.

A continuación, daremos el cálculo de las reglas para el caso de uso *inserción* y veremos cómo de manera similar se puede extender el proceso a los demás casos de uso.

La lógica del caso de uso *Inserción* se divide en dos partes (ver Figura 4), verificar si la clave del elemento a insertar existe o no en el sistema y, en el caso de no existir, cargar el resto de la información del elemento.

Primera parte. El actor debe ingresar la clave del elemento, esto es representado con ≪signal≫, luego el sistema busca si la clave existe, lo que representamos con ≪query≫. En este punto, el sistema deberá realizar tantas búsquedas dependiendo de como sea la estructura de la clave[1], aquí es donde tenemos nuestra primera dependencia de la estructura del dominio.

Segunda parte. El actor ingresa el resto de los datos del elemento a insertar, estereotipado con ≪signal≫, el sistema realiza la inserción del elemento, que se representa con ≪service≫(new). Aquí, nuevamente volvemos a depender de la estructura del dominio, debido a que en las situaciones en donde el elemento sea compuesto, es necesario interactuar con otra entidad.

De esta manera, para determinar el tamaño funcional del caso de uso *inserción* aplicamos las reglas de la función de medición, y obtenemos la siguiente fórmula:

$$Size_p(insercion) = 2+(1+Nkey)+(1+Nagregation)$$

donde, 2 *es la cantidad de mensajes ≪signal≫;* $1+Nkey$*, es el mensaje ≪query≫de la clave, más la cantidad de objetos que hay que visitar para verificar la clave, por defecto 0 (cero); y* $1 + Nagregation$ *corresponde al mensaje ≪service≫(new), más la cantidad de entidades que participan en la creación del objeto principal, por defecto 0 (cero).*

De forma similar al caso de uso *inserción*, se hace el análisis para el resto de los casos de uso. En Tabla 1, se muestran los resultados obtenidos luego de hacer el análisis.

[1] A modo de ejemplo, la clave de un cheque está formada por el número del cheque más el número de cuenta del propietario más el código del banco

Procesos Funcionales	Tamaño Funcional
Inserción	$4 + Nkey + Nagregation$
Modificación	$3 + Nagregation$
Eliminación	$3 + Nagregation$
Búsqueda	$4 + Nkey + Nagregation$
Total	$14 + 2Nkey + 4Nagregation$

Cuadro 1. Métrica de estimación de tamaño funcional para cada Caso de Uso

Como era de esperar, la métrica para determinar el tamaño funcional depende directamente de la estructura del dominio del problema sobre el cual se quiere realizar la estimación, abstrayendo la lógica de los casos de uso.

En la Sección siguiente describimos cómo se puede construir una herramienta para estimar de manera automática el tamaño funcional de sistemas de software[2].

5. Estimación automática del tamaño funcional de sistemas de Software

En la Sección 4, presentamos cómo, conociendo la lógica del problema, podemos obtener una métrica dependiente del contexto que nos permita estimar el tamaño funcional de un sistema de software[3]. El resultado obtenido en la Sección 4 corresponde a los casos de uso de una única entidad.

Utilizando el *Proceso Unificado* [9], uno de los primeros artefactos recomendados a construir en el desarrollo de software es el *modelo de dominio*. En el se encuentran representadas las principales entidades del sistema, y la notación más común usada es un *diagrama de clases* de UML[11].

En esta Sección, describiremos cómo tomando como entrada el *modelo de dominio* es posible obtener de forma automática el tamaño funcional de un sistema de software. Un aspecto a considerar es que, nos referimos a sistemas orientados a objetos en donde las funcionalidades a tener presente son del tipo *inserción*, *modificación*, *eliminación* y *búsqueda de elementos*.

Los pasos propuestos son:

Paso1) Marcar las entidades: Este paso implica identificar el tipo de las entidades en *principales* y *secundarias*. Las *principales* son las clases sobre las cuales vamos a tener funcionalidades del sistema, ejemplo, una clase *servicio* sobre la cual vamos a tener que dar de alta servicios. Las *secundarias* son las clases que

[2] Nos referimos a sistemas orientados a objetos en donde los requisitos funcionales sean del tipo *insercion, modificación, eliminación* y *búsqueda de elementos*.

[3] Cuando nos referimos al contexto hablamos del dominio del problema sobre el cual vamos a trabajar.

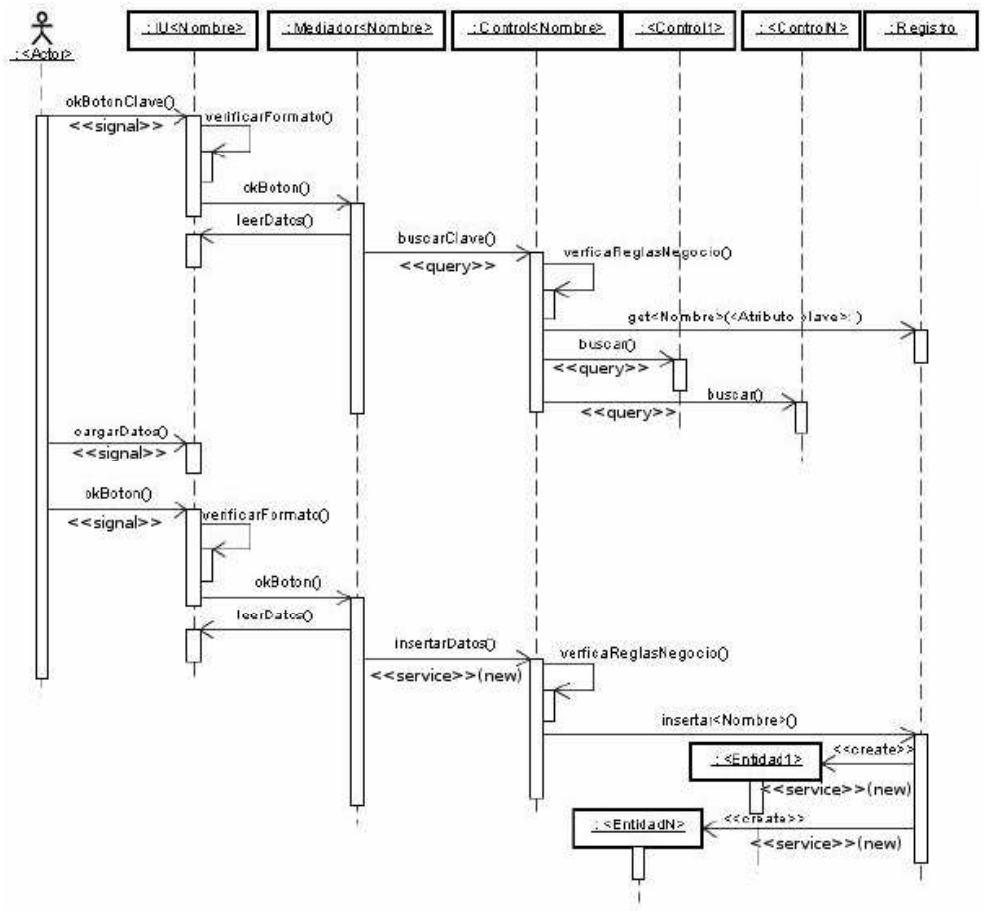

Figura 4. Ejemplo de estereotipos en el Diagrama de Sec. Inserción de Elemento

no tienen funcionalidades directas, ejemplo, una clase donde su información sea heredada de un sistema legado y no sea necesario realizar operaciones de alta de elementos.

Paso 2) Determinar las claves y relaciones: En este punto, el objetivo es detectar los valores de $Nkey$ y $Nagregation$ de las entidades principales. Esto se consigue analizando el modelo de dominio del problema.

Paso 3) Estimar el tamaño funcional: Para aplicar la métrica de la Sección 4.1, es necesario conocer las funcionalidades que cada entidad tiene. En general, se considera que las entidades tienen todas las funcionalidades (*inserción*, *eliminació*n, *modificación* y *búsqueda*), pero en algunos contexto esto no puede ser así. Según las funcionalidades, se aplica la métrica correspondiente. El resultado final, es la suma de todos los valores obtenidos.

5.1. Herramienta de estimación automática

Observando los pasos descriptos previamente, resulta sencillo imaginar la construcción de una herramienta que permita realizar la estimación de manera automática.

A partir del *modelo de dominio* y explicitando la composición de la clave de las clases, el resto es aplicar la métrica presentada.

Existe una gran cantidad de herramientas disponibles, podemos citar entre ellas [2, 1, 6, 12], las cuales pueden extenderse para agregarle esta funcionalidad.

6. Caso de estudio

En esta sección, describiremos la aplicación de nuestra propuesta de proceso de medición sobre un caso de estudio. El caso de estudio seleccionado corresponde a un sistema administrativo universitario, en donde nos centraremos en el módulo de *registrar un alumno*.

Los requisitos funcionales del usuario son presentados mediante un diagrama de casos de uso (ver Figura 5). El modelo estructural que representa el dominio específico es dado a través de un diagrama de clases de UML (ver Figura 6).

Figura 5. Caso de estudio: Diagrama de Casos de Uso

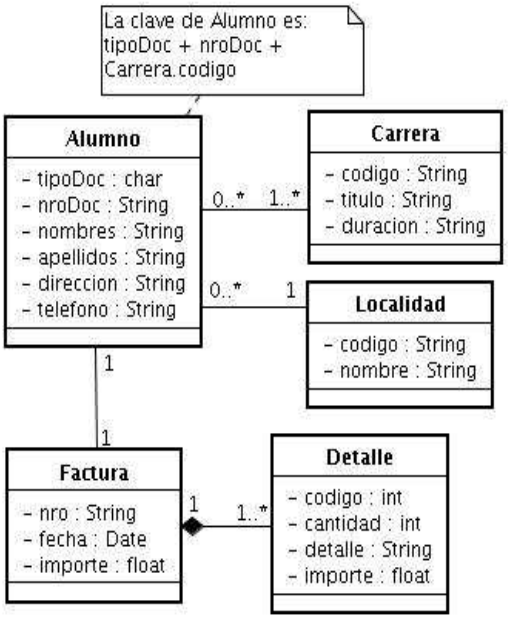

Figura 6. Caso de estudio: Modelo de dominio

6.1. Aplicación del proceso de medición al caso de estudio

Para determinar el tamaño funcional del módulo *registrar alumno*, seguiremos los tres pasos del proceso descripto en la Sección 5.

Paso1) Marcar las entidades: Se marcan como principales las entidades sobre las que existe algún tipo de funcionalidad, de esta manera tenemos:
Entidades principales: $Alumno$, $Carrera$, $Localidad$ y $Factura$. *Entidades secundarias*: $Detalle$.

Paso 2) Determinar las claves y relaciones: Del modelo de dominio se observa que: la entidad $Alumno$ posee una clave compuesta por $tipoDoc + nroDoc + Carrera : codigo$; y que la entidad $Detalle$ depende de la entidad $Factura$. En el Cuadro 2 se ven reflejados los valores de $Nkey$ y $Nagregation$ para cada entidad.

Paso 3) Estimar el tamaño funcional: Observando los requisitos funcionales del sistema, se definen las funcionalidades deseadas para cada entidad. Por ejemplo, la entidad $Alumno$ tiene todas las funcionalidades, mientras que $Localidad$ sólo la funcionalidad de *Búsqueda*. De esta manera, aplicamos la métrica correspondiente en cada caso. Los demás resultados están expresado en el Cuadro 2.

El Cuadro 2 muestra el tamaño funcional para cada entidad, así como el tamaño de todo el módulo, el mismo fue obtenido mediante la aplicación de la métrica dada en el Cuadro 1. De esta forma, el tamaño funcional del módulo *registrar alumno* es de $29 Cfsu$.

7. Consideraciones

Es importante destacar el concepto de *Aproximación a la Estimación Automática del Tamaño Funcional*, debido a que, en este trabajo, nuestro objetivo fue delinear una manera de analizar los problemas comunes y extrapolar estos a otras áreas de la ingeniería de software, como por ejemplo la estimación por medio de técnicas funcionales.

Otra consideración es que, nosotros observamos las funcionalidades del tipo *inserción*, *modificación*, *eliminación* y *búsqueda de elementos*, y en el análisis realizado suponemos que las entidades seleccionadas contarán con todas estas funcionalidades. En contexto reales, el ingeniero de software puede no requerir todas las funcionalidades sobre alguna de las entidades, en este marco, es necesario precisar cuales de ellas serán necesarias, y utilizar la métrica correspondiente. Esta situación no afecta al análisis realizado, ni a los procedimientos descriptos.

8. Conclusiones

Estimar el tamaño de sistema de software es una actividad que en muchos casos no se la considera en el nivel

Entidad	$Nkey$	$Nagregation$	Funcionalidad				Tamaño
			Inserción	*Eliminación*	*Modificación*	*Búsqueda*	
$Alumno$	1	0	Si	Si	Si	Si	16*Cfsu*
$Carrera$	0	0	No	No	No	Si	4*Cfsu*
$Localidad$	0	0	No	No	No	Si	4*Cfsu*
$Factura$	0	1	Si	No	No	No	5*Cfsu*
						Total	29*Cfsu*

Cuadro 2. Caso de estudio: Resumen de la medición del tamaño funcional

de la importancia que ella tiene. Contar con herramientas automáticas que permitan realizarla es uno de los desafíos más pujantes en este sentido. Resolver de manera genérica problemas comunes es una tarea cotidiana del ingeniero de software. En este trabajo, nos aprovecharnos de estos dos conceptos y, de esta forma, presentamos una aproximación para estimar de manera automática el tamaño funcional de un sistema de software.

En primer lugar, analizamos la lógica de los casos de uso de *inserción*, *eliminación*, *modificación* y *búsqueda* de elemento; y la aplicación parcial de un método de estimación funcional, esto nos permitió obtener una métrica para estimar el tamaño funcional de estos casos de uso. Luego, describimos el procedimiento para su aplicación en contextos reales. Para una mejor comprensión incorporamos un caso de estudio en donde se ven ejemplificado estos pasos. También presentamos los requisitos necesarios para la construcción de una herramienta que permita de manera automática estimar el tamaño funcional de un modelo de dominio.

Consideramos que, debido a la simpleza de su aplicación, esta aproximación sería de gran ayuda para la toma de decisiones que aborda un director de proyectos de software.

Actualmente, estamos trabajando en refinar los valores a diferentes contextos de aplicación, y estudiando la creación de nuevas plantillas que permitan abordar otros tipos de problemas y, de esta manera, poder contar con una librería de opciones que hagan más completa la estimación.

Referencias

[1] ArgoUML Project. ArgoUML. Available at: `http://argouml.tigris.org/`.

[2] Borlan Together Tecnologies. Together. Available at: `http://www.borland.com/us/products/together/` .

[3] Common Software Measurement International Consortiun. COSMIC-FFP Measurement Manual, 2003. Version 2.2.

[4] N. Condori-Fernandez, S. Abrahao, O. Pastor, and S. Marti. Un Procedimiento de Medición de Tamaño Funcional: Diseño y Aplicación. *Revista de Procesos y Métricas de las Tecnologías de la Información. Vol 1, Nro 3*, 2004.

[5] 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.

[6] Eclipse Foundation. Eclipse. Available at: `http://www.eclipse.org/`.

[7] E. Insfran. Requirements Engineering Approach for Object-Oriented Conceptual Modeling, 2003. PhD Thesis, Technical University of Valencia.

[8] ISO/IEC. Iso/iec 19761 software engineering - cosmic-ffp- a funcional size measurement method, 2003.

[9] I. Jacobson, G. Booch, and J. Rumbaugh. *El Proceso Unificado de Desarrollo de Software*. Addison Wesley, 1999.

[10] J. P. Jacquet and A. Abran. From Software Metrics to Software Measurement Methods: A Process Model. *3rd Int. Standard Symposium and Forum on Software Engineering Standards*, 1997.

[11] Object Management Group. Unified Modeling Language 2.0, 2003. Adopted Specification, ptc/03-09-15.

[12] IBM Rational. Rational Rose. Available at: `http://www.rational.com/`.