Un Procedimiento de Medición de Tamaño Funcional para Modelos Conceptuales en entornos MDA

Tesis de Máster en Ingeniería del Software, Métodos Formales y Sistemas de Información

Beatriz Marín Campusano

Directores:
Dra. Nelly Condori-Fernández
Dr. Oscar Pastor López

Departamento de Sistemas Informáticos y Computación

Septiembre 2008

UNIVERSIDAD POLITECNICA DE VALENCIA

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Tesis de Máster en Ingeniería del Software, Métodos Formales y Sistemas de Información

Un Procedimiento de Medición de Tamaño Funcional para Modelos Conceptuales en entornos MDA

Beatriz Marín Campusano

Directores: Nelly Condori-Fernández
Oscar Pastor López

A mi amado Giovanni

Agradecimientos

Esta tesis es el resultado de muchos años de trabajo, reuniones, y ricas discusiones en las que he estado acompañada por mucha gente, a quienes en esta ocasión les quiero expresar mi gratitud.

En primer lugar quiero agradecer a mi director Oscar Pastor, por haber confiado en mí incluso antes de conocerme, por darme la oportunidad de conocer el fascinante mundo de la investigación, por ser un gran director y un excelente amigo. ¡Muchísimas gracias Oscar!

También, quiero agradecer a mi directora Nelly Condori-Fernández, por compartir la motivación de investigar en calidad de software y por enseñarme el sentido de la paciencia en el trabajo.

De igual forma, quiero agradecer a mis compañeros de trabajo con los que comparto día a día en el centro PROS:

A Vicente Pelechano, por el cariño que me ha entregado y por mostrarme luces cada vez que me encontré en un túnel oscuro.

A Vicky, Ana y Pedro, por la ayuda y el cariño que siempre me han otorgado, y también, por sus valiosos consejos a lo largo de estos años. ¡Chicos, ustedes son luciérnagas que iluminan mi camino!

A Pau, J-Lu y Paco, por ser unos excelentes amigos y compañeros en el mundo de la investigación.

A Nathalie, Fani, Paqui y Carlos, por demostrarme su cariño y buena onda cada día.

Al resto de amigos y compañeros por estar siempre ahí: Marta, Manoli, Gonzalo, Joan, Juan Sánchez, Luis, Sergio, Ignacio, Javi, Isabel e Ismael.

Asimismo, quiero agradecer a Juan Carlos y Barbe de la empresa CARE Technologies por las experiencias compartidas, y en especial a cada uno de los amigos que he conocido ahí.

Además, quiero agradecer a mis sobrinos, hermanos, cuñados y padres por intentar que no se noten las distancias físicas y entregarme siempre todo su amor y apoyo.

Finalmente, quiero agradecer a mi marido Giovanni, por su amor y apoyo incondicional, y por permitir que mi sueño de estudiar un doctorado se convierta en una realidad. ¡Te amo Perri!

Tabla de contenidos

Capítulo 1
Introducción

Durante los últimos años, el proceso de producción de software ha estado evolucionando para dejar de centrarse en el espacio de la solución (producto de software) y centrarse en el espacio del problema (modelos conceptuales). La arquitectura MDA (Model Driven Architecture) [Miller03] separa la lógica de las aplicaciones de la plataforma tecnológica que las soporta, permitiendo la generación de código mediante transformaciones de modelo a modelo y transformaciones de modelo a código. En este contexto, la arquitectura MDA se apoya en un proceso de producción de software centrado en el espacio del problema, donde los modelos conceptuales son usados como entrada en el proceso de generación de la solución.

Para que sea posible utilizar procesos de producción de software centrados en el espacio del problema, como es el caso de MDE (Model Driven Engineering) [Schmidt2006], los modelos conceptuales que se utilicen deben tener la formalización semántica necesaria para especificar toda la funcionalidad que tendrán las aplicaciones generadas, eliminando la posibilidad de realizar diferentes interpretaciones de un mismo modelo.

La adopción de procesos de producción MDE que utilizan tecnologías basadas en el enfoque MDA ha presentado nuevos desafíos para la ingeniería del software, como es el caso de la correcta gestión de los proyectos que utilizan estas tecnologías, ya que los procesos de desarrollo se acortan debido a la automatización de la generación del producto de software, y por consiguiente la utilización de recursos y el costo de la aplicación generada varía.

Actualmente, es ampliamente aceptado que el tamaño funcional de las aplicaciones es esencial para aplicar modelos de estimación, modelos de esfuerzo y modelos de presupuesto de proyectos [Meli2000] que permiten al jefe de proyecto generar indicadores para facilitar la gestión de los proyectos. Por esta razón, en esta tesis nos hemos centrado en la medición del tamaño funcional, proponiendo un procedimiento de medición para medir el tamaño funcional de las aplicaciones generadas en entornos MDA a partir de sus modelos conceptuales.

El resto de este capítulo introductorio está organizado de la siguiente manera: la Sección 1.1 presenta las razones que inspiran la realización de esta tesis y la Sección 1.2 presenta el problema abordado en este trabajo. La Sección 1.3 presenta los objetivos que se pretenden alcanzar con esta tesis y la Sección 1.4 presenta el contexto de investigación en el que se ha desarrollado esta tesis. Finalmente, la Sección 1.5 presenta la estructura de todo el documento, describiendo brevemente el contenido de cada capítulo que compone la tesis.

1.1 Motivación

Actualmente, la sociedad depende cada día más de los productos de software. Se puede observar que se utilizan productos de software para realizar transacciones bancarias, para comunicarse con otras personas, para localizar puntos geográficos, para buscar información, para realizar trámites gubernamentales, para controlar los dispositivos en el hogar, e incluso para realizar actividades críticas como son operaciones, tratamientos de salud, control de energías nucleares, etc. La lista de productos de software y campos de acción en donde se utilizan es extensa y variada. Esto pone de manifiesto que los productos de software se han convertido en un bien preciado de nuestra era (denominada comúnmente la era digital), un bien en el que los gobiernos, las empresas y las personas invierten grandes cantidades de dinero para obtener más y mejores servicios.

Dado que los productos de software son bienes preciados que muchas veces apoyan actividades críticas para personas, empresas y gobiernos, resulta crucial la administración y el control de este tipo de productos. Sin embargo, como dijo Tom de Marco [DeMarco82]: "No se puede controlar lo que no se puede medir" y como dijo Norman Fenton [Fenton97]: "No se puede predecir lo que no se puede medir". Estas frases célebres expresan claramente que la medición de los productos de software es crucial para el control y la predicción de los proyectos de desarrollo de software.

La medición de los productos de software puede ser realizada desde diversas perspectivas, las cuales se pueden agrupar principalmente en dos grupos de aproximaciones: a priori y a posteriori [TranCao02]. Las aproximaciones a posteriori (por ejemplo: tamaño de disco, líneas de código [Conte86], etc.) tienen la desventaja de ser tardías y poco significativas para la gestión dada su alta dependencia de la tecnología. Por el contrario, las

aproximaciones a priori (por ejemplo: tamaño funcional) ganan cada vez más atención en la comunidad de la medición de productos de software, ya que son independientes de la plataforma tecnológica en la que se explote o desarrolle el producto y permiten la generación de indicadores en etapas tempranas del ciclo de desarrollo del software.

Teniendo en cuenta las ventajas de la medición del tamaño funcional (a priori), en 1979 Albrecht definió la unidad *puntos de función* [Albrecht79] para cuantificar el tamaño funcional de los productos de software. Además de la unidad de puntos de función, Albrecht propuso el método de medición Análisis de Puntos de Función (Function Point Analysis - FPA) [Albrecht79]. En base a este método inicial de medición de tamaño funcional, en la literatura se pueden encontrar varias propuestas que pretenden mejorar este método de medición. De estas propuestas, cuatro han sido reconocidas como estándares por las organizaciones ISO (International Organization for Standarization) e IEC (International Electrotechnical Commission):

- ISO/IEC 19761 [ISO03]: COSMIC-FFP – A Functional Size Measurement Method
- ISO/IEC 20926 [ISO03a]: IFPUG 4.1 Unadjusted Functional Size Measurement Method - Counting Practices Manual
- ISO/IEC 20968 [ISO02]: Mk II Function Point Analysis – Counting Practices Manual
- ISO/IEC 24570 [ISO04]: NESMA Functional Size Measurement Method version 2.1 – Definitions and Counting Guidelines for the application of Function Point Analysis

Con la utilización de los métodos estándares de medición de tamaño funcional es posible obtener el tamaño de los productos de software, que puede ser utilizado para controlar el desarrollo esos productos mediante el cálculo de indicadores y la comparación con indicadores obtenidos en otros proyectos. Por ejemplo puede ser utilizado para estimar casos de prueba, para entender la productividad, para entender el crecimiento de los proyectos, para calcular el costo real de software, para estimar el costo de nuevos proyectos, para estimar el esfuerzo necesario en nuevos proyectos, para entender los costos de mantenimiento de los

productos de software, para definir indicadores de la calidad del software según los defectos, etc.

Además, junto a la evolución de la sociedad con respecto a la utilización de las tecnologías de información, la ingeniería del software ha evolucionado también en los procesos de desarrollo, centrándose en los modelos conceptuales que permiten generar correctamente las aplicaciones finales mediante transformaciones de modelos, como es el caso de MDE y MDA. En términos prácticos, es muy importante conocer el tamaño exacto de los modelos conceptuales construidos con aproximaciones basadas en MDE o MDA, dado que a partir del tamaño funcional de los modelos conceptuales se pueden obtener las ventajas de la medición del tamaño funcional en etapas tempranas del desarrollo de software. Finalmente, debido a que en la actualidad se está masificando el uso de arquitecturas MDA que permiten generar la aplicación final mediante transformaciones de modelo, es muy relevante la automatización de la aplicación de los métodos de medición de tamaño funcional al proceso de desarrollo de software.

1.2 Planteamiento del Problema

La medición del tamaño funcional (Functional Size Measurement - FSM) fue definida en los últimos años de la década de los 70's a través del método de Análisis de Puntos de Función (FPA) definido por Albrecht [Albrecht79]. Posteriormente, el método FPA ha sido aplicado masivamente a nivel industrial en base a la propuesta del Grupo Internacional de Usuarios de Puntos de Función (Internacional Function Point Users Group – IFPUG) [IFPUG]. La propuesta de IFPUG ha sido adoptada como el estándar ISO/IEC 20926 [ISO03a]. Además del método definido por IFPUG, otros autores han definido variantes al método de Análisis de Puntos de Función que también han sido reconocidos como estándares (ISO/IEC 20968 [ISO02] y ISO/IEC 24570 [ISO04]).

Dado que el desarrollo de productos de software ha evolucionado para centrarse en el espacio del problema en vez del espacio de la solución, el método de análisis de puntos de función (FPA) y sus variantes han sido adaptadas por diversos autores para aplicarlo a modelos conceptuales orientados a objetos (por ejemplo [Lehne97], [Uemura99], etc.). Sin embargo, las aproximaciones basadas en FPA presentan limitaciones para la medición del tamaño funcional de modelos conceptuales en entornos MDA. Una de las limitaciones de las aproximaciones

basadas en FPA es que sólo permiten la medición del tamaño funcional de las funcionalidades que el usuario (humano) de la aplicación observa, ignorando todas las demás funcionalidades que deben ser construidas para el correcto funcionamiento de la aplicación, y que generalmente el usuario (humano) no puede observar. Otra limitación de estas aproximaciones es que no consideran las funcionalidades que permiten conectar las capas de las aplicaciones que tienen más de una capa (por ejemplo las aplicaciones con estructura de tres capas: cliente, servidor y base de datos). Otra limitación importante es que el tamaño de cada proceso elemental de un modelo está limitado a sólo tres intervalos de clasificaciones para DET (Data Element Types), RET (Record Element Types) o FTR (File Types Referenced), los cuales asignan un valor específico al tamaño funcional de acuerdo a las tablas de complejidad del estándar FPA. De esta manera, el tamaño funcional de una aplicación no variará cuando sobrepase los valores máximos de DET, RET o FTR [Abran94] [Kitchenham97] [Giachetti07]. A pesar de las limitaciones que presenta la medición de modelos conceptuales en entornos MDA con FPA, en la literatura se han encontrado propuestas que realizan esta medición [Abrahao01] [Abrahao07].

Para superar las limitaciones del diseño inicial del método de medición FPA, el método de medición COSMIC [Abran03] fue definido en los últimos años de la década de los 90's como una segunda generación de métodos de medición de tamaño funcional. Este método ha sido adoptado como el estándar internacional ISO 19761 [ISO03]. El método de medición COSMIC utiliza una función matemática para agregar el tamaño funcional de los procesos funcionales especificados en un modelo conceptual. Esta función matemática no está limitada por valores máximos para medir el tamaño funcional de modelos conceptuales, ayudando así a distinguir mejor el tamaño funcional de grandes modelos conceptuales. Además, COSMIC permite la medición de toda la funcionalidad que ha sido desarrollada para el correcto funcionamiento de las aplicaciones, sin limitarse a medir sólo la funcionalidad que el usuario (humano) ve, ya que considera todos los movimientos de datos que se deben realizar para mostrar cierta información al usuario. Otra ventaja de COSMIC es que permite la medición de aplicaciones generadas en capas, permitiendo la medición de toda la aplicación o de cada capa de la aplicación.

Actualmente, existen algunas aproximaciones que aplican COSMIC para estimar el tamaño funcional de futuras aplicaciones desde especificaciones de alto nivel (requisitos) [Condori07a] [Jenner02]. Sin embargo, dado que la funcionalidad a ser medida

por estas propuestas no está suficientemente detallada para generar la aplicación final, el tamaño funcional que obtienen dista bastante del tamaño funcional de la aplicación final.

Otros autores han utilizado COSMIC para definir procedimientos de medición de tamaño funcional a partir de modelos conceptuales, ya que éstos sí presentan los detalles necesarios para generar la aplicación final en entornos MDA. Este es el caso de las propuestas de Poels [Poels02] y Diab [Diab05]. Ambos procedimientos de medición de tamaño funcional fueron definidos estableciendo un mapping entre los conceptos de COSMIC y las primitivas del modelo conceptual MERODE o de las primitivas de los modelos de tiempo real de Rational Rose (RRRT) respectivamente. Las propuestas de Poels y Diab están basadas en modelos conceptuales que permiten la generación de aplicaciones de manera parcial, pues no permiten la generación de aplicaciones que funcionen correctamente a partir de las especificaciones de los modelos conceptuales. En este sentido, el tamaño funcional que obtienen tampoco corresponde al tamaño funcional de la aplicación final.

En resumen, no existe un método de medición de tamaño funcional que permita medir correctamente el tamaño funcional de aplicaciones generadas en entornos MDA a partir de modelos conceptuales, los cuales deben tener la expresividad necesaria para representar de manera abstracta toda la funcionalidad de la aplicación generada.

Para resolver el problema de expresividad de los modelos conceptuales se ha seleccionado la aproximación OO-Method [Pastor01]. Esta aproximación es un método orientado a objetos que se basa en las transformaciones de modelos y provee la formalización semántica necesaria para definir completamente y sin ambigüedades los modelos conceptuales implicados, permitiendo la generación automática de productos de software [Pastor07] usando una tecnología basada en MDA. Este método ha sido implementado en una suite de herramientas industriales por la empresa CARE Technologies [CARE].

Finalmente, para resolver el problema planteado, esta tesis presenta un procedimiento de medición de tamaño funcional para los modelos conceptuales OO-Method basado en COSMIC, que permitirá una mejor medición del tamaño funcional de las aplicaciones generadas automáticamente en entornos MDA a partir de modelos conceptuales OO-Method.

1.3 Objetivos

El objetivo que intenta conseguir esta tesis es *proveer un procedimiento de medición*[1] *basado en el método de medición COSMIC, que permita la medición automática del tamaño funcional de aplicaciones generadas automáticamente a partir de modelos conceptuales OO-Method*.

Para lograr el objetivo general de esta tesis, es necesario satisfacer los siguientes objetivos específicos:

- Definir un conjunto de reglas que permitan cuantificar el tamaño funcional de los elementos del modelo conceptual OO-Method.

- Definir un proceso que guíe la aplicación de esas reglas sobre modelos conceptuales OO-Method.

- Automatizar la aplicación de las reglas que permiten medir los modelos conceptuales OO-Method.

1.4 Contexto de Investigación

Esta tesis ha sido desarrollada dentro del grupo de investigación OO-Method, perteneciente al Centro de Investigación en Métodos de Producción de Software (PROS) de la Universidad Politécnica de Valencia. Las principales áreas de investigación del centro PROS son Ingeniería de Requisitos, Ingeniería Web, Interacción Persona-Ordenador, Calidad de Software, y Desarrollo de Sistemas Ubicuos, todas ellas abordadas desde la perspectiva del modelado conceptual. El trabajo de esta tesis ha sido realizado dentro del sub-grupo de Calidad de Software del grupo OO-Method.

El trabajo realizado ha sido posible gracias a convenios de investigación que tiene el centro PROS con la empresa CARE Technologies, y gracias a proyectos financiados por el Centro de Investigación Científica y Tecnológica de España (CICYT) y por el Ministerio de Industria, Turismo y Comercio de España (MITyC). A

[1] Según el vocabulario internacional de términos básicos y generales de metrología (International Vocabulary of Basic and General Terms of Metrology - VIM) [ISO04a], un procedimiento de medición consiste en una descripción detallada de una medición de acuerdo a uno o más principios de medición y a un método de medición determinado. Asimismo, el VIM define un método de medición como una descripción genérica de una secuencia lógica de operaciones realizadas en una medición.

continuación se listan los convenios que han soportado este trabajo de investigación:

- CONCOM: Construcción de Compiladores. Proyecto de I+D financiado por CARE Technologies desde 2006 a 2008.

- PISA: Producción Industrial de Software en Ambientes MDA. PROFIT de MITyC desde 2007 a 2008.

- SESAMO: Construcción de Servicios Software a partir de Modelos. Proyecto CICYT referenciado como TIN2007-62894 desde 2008 a 2010.

El resultado de esta tesis también tiene una perspectiva industrial, ya que será utilizado por la empresa CARE Technologies para la estimación del coste de nuevas aplicaciones generadas con su compilador de modelos OO-Method.

1.5 Estructura de la Tesis

Esta tesis está estructurada en siete capítulos, los cuales se describen brevemente a continuación:

- **Capítulo 1: Introducción**
  El presente capítulo introduce el contexto tecnológico en el que se desarrolla la tesis, exponiendo claramente el problema y la motivación que se tiene para realizar esta tesis. Además, se muestran detalladamente los objetivos que persigue la realización de esta tesis, el contexto de investigación en que se ha realizado la tesis. En último lugar, se describe la estructura de la tesis.

- **Capítulo 2: Estado del Arte**
  En este capítulo se presentan brevemente los cuatro métodos estándares de medición de tamaño funcional, y se presenta una revisión sistemática de las propuestas de procedimientos de medición de tamaño funcional de modelos conceptuales.

- **Capítulo 3: Fundamentos**
  En este capítulo se presentan los trabajos que se han utilizado como base para el desarrollo de la tesis. Inicialmente se introduce la versión 3.0 del método de medición de tamaño funcional COSMIC, luego se

introduce el método de desarrollo de aplicaciones OO-Method, en donde se explica detalladamente el modelo conceptual de OO-Method. Finalmente se presenta el modelo de procesos para la medición del software utilizado para el desarrollo sistemático del procedimiento de medición que propone esta tesis.

- **Capítulo 4: Diseño de un Procedimiento de Medición de Tamaño Funcional**
  En este capítulo se presenta el diseño del procedimiento de medición OOmCFP (OO-Method COSMIC Function Points). El diseño de OOmCFP comprende una serie de reglas que permiten seleccionar los elementos del modelo conceptual de OO-Method que contribuyen al tamaño funcional de las aplicaciones generadas según el método de medición COSMIC, y también un conjunto de reglas que permiten calcular el tamaño funcional de las aplicaciones generadas utilizando OO-Method.

- **Capítulo 5: Aplicación del Procedimiento de Medición OOmCFP**
  En este capítulo se describe el conjunto de pasos que se deben seguir para aplicar OOmCFP a modelos conceptuales OO-Method. Estos pasos son ilustrados en un caso de estudio.

- **Capítulo 6: Automatización de OOmCFP**
  En este capítulo se presenta la herramienta que automatiza la aplicación de OOmCFP, detallando los mecanismos que se han tenido en cuenta para que la medición sea eficiente. Además, se describe como se utiliza la herramienta que implementa OOmCFP, y se muestran los resultados de 5 modelos reales medidos utilizando la herramienta OOmCFP.

- **Capítulo 7: Conclusiones**
  En este capítulo se presentan las contribuciones que aporta esta tesis. Además se presentan los trabajos actuales y futuros, y las publicaciones generadas con el trabajo de esta tesis.

Capítulo 2
Estado del Arte

En este capítulo se presenta un resumen de los métodos estándares de medición de tamaño funcional y luego se realiza una revisión sistemática de los procedimientos de medición de tamaño funcional que permiten medir o estimar el tamaño funcional de aplicaciones de software a partir de modelos conceptuales.

2.1 Métodos de Medición Estándares

Actualmente existen cuatro métodos de medición reconocidos como estándares según las organizaciones ISO (International Organization for Standarization) e IEC (International Electrotechnical Commission): IFPUG FPA, MK II FPA, NESMA FPA y COSMIC.

Los métodos IFPUF FPA, MK II FPA y NESMA FPA son conocidos como métodos de primera generación debido a que se han basado en el método de Análisis de Puntos de Función (FPA) definido por Albretch [Albretch79]. Por otro lado, el método COSMIC es conocido como método de segunda generación porque extiende el punto de vista de la medición. Los métodos basados en FPA sólo contemplan la funcionalidad que puede observar el usuario final de la aplicación, en cambio los métodos de segunda generación contemplan la funcionalidad del sistema desde el punto de vista del usuario final y desde el punto de vista de otros usuarios de la aplicación (por ejemplo otras aplicaciones).

A continuación se describe brevemente cada uno de estos métodos de medición estándares, destacando su origen y su evolución en el tiempo.

2.1.1 IFPUG FPA (ISO/IEC 20926)

Este estándar se basa en el método FPA definido por Albretch y publicado por primera vez en 1979 [Albretch79]. Luego, Albretch refinó el método y publicó una nueva versión en 1984. En 1986, fue fundado el Grupo Internacional de Usuarios de Puntos de Función (International Function Point User Group – IFPUG), que se

encarga de la difusión del método y de la creación de manuales de uso del método, publicando varias versiones de manuales (3.4, 4.0, 4.1, 4.1.1, y 4.2).

En 2003, el manual de conteo versión 4.1 [IFPUG99] fue reconocido como estándar mediante la norma ISO/IEC 20926 [ISO03a]. Cabe destacar que esta norma no incluye la parte del ajuste de los puntos de función calculados, ya que esta parte no reúne los requisitos especificados en la norma ISO/IEC 14143-1 [ISO98].

La medición del tamaño funcional especificada por IFPUG FPA es calculada mediante la identificación de cinco elementos: archivos lógicos internos (Internal Logical Files – ILF), archivos de interfaz externos (Interface Files – EIF), entradas externas (External Inputs – EI), salidas externas (External Outputs – EO), y consultas externas (External Inquiries – EQ). Luego, se asigna un peso a los elementos identificados de acuerdo a su complejidad (alta, baja, o media). Finalmente, los pesos de los elementos identificados son agregados para obtener la medida del tamaño funcional.

El manual de IFPUG versión 4.1 sugiere que la identificación de elementos que contribuyen al tamaño funcional debe ser realizada a partir de uno o varios de los siguientes componentes: requisitos de usuarios, modelos de datos, modelos de procesos, ventanas, pantallas o reportes de la aplicación de software.

Una vez que los elementos que contribuyen al tamaño funcional han sido identificados, el proceso de medición es realizado de acuerdo a las reglas que asignan la complejidad a cada elemento. La complejidad de cada ILF o EIF es calculada mediante el conteo de tipos elementales de datos (Data Element Types – DET) y tipos elementales de registros (Record Element Types – RET). La complejidad de cada EI, EO o EQ es calculada mediante el conteo de DETs y tipos de archivos referenciados (File Types Referenced – FTR).

2.1.2 MK II FPA (ISO/IEC 20968)

MK II FPA es un método de medición definido en 1988 por Charles Symons como una variante del método FPA propuesto por Albretch. Posteriormente, este método ha sido mantenido y difundido por la Asociación de Métricas del Reino Unido (UKSMA) en varias versiones.

A finales del 2002, la versión 1.3.1 de MK II FPA [UKSMA98] fue reconocida como estándar mediante la norma ISO/IEC 20968 [ISO02]. El método MK II FPA ha sido el primer método de medición de tamaño funcional reconocido como estándar, debido a que la versión 1.3.1 fue desarrollada conforme al estándar ISO/IEC 14143-1 [ISO98].

La medición del tamaño funcional especificada por MK II FPA es calculada mediante la identificación de transacciones lógicas, las cuales corresponden a una combinación única de Entrada, Proceso, y Salida, que es desencadenada por la ejecución de un evento por el usuario. Luego, se asigna un peso a los elementos identificados, y finalmente, los pesos de los elementos identificados son agregados para obtener la medida del tamaño funcional.

La identificación de transacciones lógicas puede ser realizada desde los siguientes componentes: modelos de datos, modelos de diseño, ventanas, pantallas o reportes de la aplicación de software.

Una vez que se hayan identificado las transacciones lógicas, se calcula el tamaño de cada transacción lógica mediante el tamaño de los componentes de entrada y de salida a través del número de tipos elementales de datos (DETs), y el tamaño de los componentes de proceso a través del número de referencias a entidades de un sistema, cada uno de ellos respectivamente pesado.

2.1.3 NESMA FPA (ISO/IEC 24570)

La asociación holandesa de usuarios de métricas de software (Netherlands Software Metrics Users Association – NESMA) publicó una primera versión de un manual de aplicación del método FPA en 1990. Este manual se basó en la versión del manual 3.4 de IFPUG FPA. En esta primera versión se entregaban guías concretas, consejos y ejemplos para aplicar el método de medición definido en el manual 3.4 de IFPUG.

Posteriormente, una nueva versión del manual de conteo NESMA fue publicada en 1996, que contenía guías, consejos y ejemplos sobre características complejas de conteo.

En 2004, la versión 2.0 de NESMA FPA fue reconocida como estándar mediante la norma ISO/IEC 24570 [ISO04], debido a que su definición es conforme a la norma ISO/IEC 14143-1.

La medición del tamaño funcional utilizando el método estándar NESMA es prácticamente igual al método estándar IFPUG FPA, por lo que no se detallará la forma de calcular el tamaño funcional.

2.1.3 COSMIC FFP (ISO/IEC 19761)

En 1997, St-Pierre et al. [StPierre97] propusieron el método Full Function Points (FFP) para la medición de sistemas de control, tiempo real, y embebidos. Luego, en 1999 el Consorcio Internacional para la Medición Común de Software (Common Software Measurement International Consortium – COSMIC) publica la versión 2.0 del método de medición COSMIC-FFP [Abran99a].

Posteriormente, se realizaron algunas modificaciones a la versión 2.0 de COSMIC FFP, publicándose la versión 2.1 en 2001 [Abran01] y la versión 2.2 en 2003 [Abran03]. Esta última versión fue reconocida como estándar mediante la norma ISO/IEC 19761 [ISO03]. Durante 2007 se publicó una nueva versión de COSMIC, que se diferencia de la anterior versión porque agrega una fase para definir la estrategia de medición, la cual contempla la identificación de los usuarios funcionales de la aplicación, dejando de lado la relevancia del punto de vista desde donde se realiza la medición.

La medición del tamaño funcional especificada por COSMIC es calculada mediante la identificación procesos funcionales y movimientos de datos que ocurren en esos procesos funcionales. Los movimientos de datos pueden ser de entrada al proceso funcional, de salida al proceso funcional, de lectura de la base de datos y de escritura a la base de datos. Cada movimiento de datos mueve un grupo de datos particular. Una vez identificados los movimientos de datos, se agregan los movimientos para obtener el tamaño funcional de cada proceso funcional.

2.2 Revisión Sistemática de Procedimientos de Medición de Modelos Conceptuales

En base a los estándares de métodos de medición presentados anteriormente, varios autores han definido propuestas que permitan aplicar esos métodos en modelos conceptuales. Para identificar estas propuestas, se ha realizado una revisión sistemática siguiendo el procedimiento definido por Kitchenham [Kitchenham04]. Este procedimiento se compone tres fases: una

fase para planificar la revisión, una fase para realizar la revisión y una fase para reportar los resultados de la revisión. A continuación se detalla cada una de estas fases.

2.2.1 Planificando la revisión

La fase para planificar la revisión comprende dos pasos: (1) identificar la necesidad de realizar la revisión y (2) desarrollar un protocolo de revisión.

Identificar la necesidad de realizar una revisión

La necesidad de realizar una revisión sistemática surge de la necesidad de resumir toda la información existente sobre procedimientos de medición de tamaño funcional que permiten medir el tamaño funcional en modelos conceptuales.

Si bien existen revisiones iniciales de de procedimientos de medición de tamaño funcional para modelos conceptuales (por ejemplo en [Condori07] y [Abrahao04]), es importante realizar una revisión sistemática para resumir la evidencia actual de procedimientos de medición de tamaño funcional para modelos conceptuales, para identificar brechas existentes entre las diferentes propuestas y determinar investigaciones futuras, para identificar las propuestas de procedimientos de medición de tamaño funcional evitando sesgos, y para proveer evidencia robusta y transferible de procedimientos de medición de tamaño funcional para modelos conceptuales.

Desarrollar un protocolo de revisión

El protocolo de revisión incluye antecedentes para la revisión, la especificación de las preguntas de investigación, la especificación de la estrategia de búsqueda, la especificación de los criterios de selección de estudios, la especificación de preguntas para asegurar la calidad de los estudios, la especificación de la estrategia de extracción de datos, la especificación de la síntesis de los datos y la especificación de la planificación (calendario) de la revisión sistemática.

El protocolo de la revisión debe ser especificado y documentado antes de realizar la revisión para reducir desviaciones en los estudios encontrados. Además, el protocolo debe ser revisado y corregido varias veces por el equipo que realizará la revisión sistemática, de manera de obtener un protocolo de revisión tan

bueno como sea posible. A continuación se explican las diferentes partes y contenidos del actual protocolo de revisión que fue utilizado para realizar la revisión sistemática.

Antecedentes

Durante los últimos años, el proceso de producción de software ha ido evolucionando para dejar de centrarse en el espacio de la solución (producto final de software) y centrarse en el espacio del problema (modelos conceptuales). Para dar soporte a esta evolución, los procesos de producción de software utilizan tecnologías como MDA (Model Driven Architecture) [Miller03] o MDE (Model Driven Engineering) [Schmidt06], las cuales utilizan modelos conceptuales a diferentes niveles de abstracción para permitir la generación del producto final de software mediante transformaciones de modelos. La adopción de estos nuevos procesos de producción de software presenta nuevos desafíos para la ingeniería del software, entre los cuales se encuentra la adecuada gestión de proyectos, ya que al generar el producto de software de manera automática, los recursos utilizados y el costo de la aplicación varían en comparación a un proceso de desarrollo tradicional.

Hoy en día, es ampliamente aceptado que el tamaño funcional de los productos de software es esencial para aplicar modelos de estimación, modelos de esfuerzo y modelos de presupuesto de proyectos [Meli00], los cuales permiten a los jefes de proyecto generar indicadores para facilitar la gestión de proyectos. Para una mejor gestión, es deseable tener indicadores en etapas tempranas del desarrollo de los sistemas software, sin embargo, los métodos estándares para la medición del tamaño funcional (IFPUG FPA [ISO03a], COSMIC FFP [ISO03], MK II FPA [ISO02], y NESMA FPA [ISO04]) han sido ilustrados en la medición de aplicaciones finales. Teniendo en cuenta que un procedimiento de medición corresponde a la aplicación de un método de medición en cualquier fase de desarrollo del producto de software, es posible encontrar en la literatura una gran cantidad de propuestas de procedimientos de medición que permiten medir el tamaño funcional en los modelos conceptuales. Por esta razón, existe la necesidad de realizar una revisión sistemática para evidenciar los procedimientos de medición de tamaño funcional en modelos conceptuales existentes.

En base a estos antecedentes se han desarrollado preguntas de investigación que serán detalladas en la siguiente sub-sección.

Preguntas de investigación

En esta sub-sección se presenta la pregunta de investigación formulada para dirigir la revisión sistemática. Es importante destacar que cuando se formula la pregunta de investigación, se deben identificar los resultados importantes asociados a la pregunta en base a poblaciones, intervenciones y resultados del estudio.

En la revisión sistemática se realiza un estudio de procedimientos de medición de tamaño funcional para modelos conceptuales en base a la siguiente pregunta:

¿Qué evidencia existe de procedimientos de medición de tamaño funcional en modelos conceptuales?

De acuerdo a Khan et al. [Khan01], la revisión sistemática puede ser pensada como un análisis de datos existentes dentro de un conjunto de poblaciones, de intervenciones, de resultados y de diseños de estudios. Kitchenham define y ejemplifica estos conceptos en el ámbito de la ingeniería del software en [Kitchenham04], y recomienda considerarlos como parte de la estructura de la pregunta de investigación. En base a estas definiciones, la Tabla 1 presenta las poblaciones, intervenciones, resultados y diseños de estudios que serán considerados en esta revisión sistemática.

Tabla 1. Estructura de la pregunta de investigación.

Concepto	**Definición**
Poblaciones	Procedimientos de medición para ser aplicados a modelos conceptuales, ya sean de uso industrial o académico.
Intervenciones	Medición de tamaño funcional del software.
Resultados	Diseño, validación y automatización del procedimiento.
Diseño de estudios	Casos de estudio, experimentos, reportes de experiencias, reportes técnicos.

Estrategia de búsqueda

La estrategia de búsqueda es construida en base a los componentes de la pregunta de investigación (poblaciones, intervenciones, resultados y diseño de estudios). La estrategia de búsqueda define los términos para realizar la búsqueda y las fuentes en donde se realizará la búsqueda. Teniendo en cuenta

que la investigación está enfocada principalmente a buscar procedimientos de medición de tamaño funcional en modelos conceptuales, a continuación se describen los términos y las fuentes definidas para la estrategia de búsqueda.

1) Términos de búsqueda

Los términos de búsqueda permiten localizar literatura importante a ser incluida en el estudio. Los términos de búsqueda fueron seleccionados desde los términos que tiene la pregunta de investigación (considerando toda su estructura). Los términos seleccionados y sus variantes fueron identificados por el diseñador del protocolo en lecturas de estudios primarios realizadas en etapas anteriores a la revisión sistemática.

Además, dado que la revisión sistemática está enfocada en procedimientos de medición de tamaño funcional, se han incluido los nombres de los métodos estándares de medición de tamaño funcional en los términos de búsqueda. Finalmente, se ha incluido el término Method dentro de los términos de búsqueda ya que la mayoría de procedimientos de medición recibe equívocamente el nombre de método. De esta manera, al incluir este término en la búsqueda se obtendrán procedimientos que estén erróneamente nombrados como métodos, según las definiciones de método de medición y procedimiento de medición del Vocabulario Internacional de Términos Básicos y Generales de Metrología [ISO04a].

Los términos utilizados para extraer los estudios primarios están escritos en inglés, ya que este lenguaje ha sido seleccionado para realizar la búsqueda. A continuación se listan los términos de búsqueda:

- Function Point
- Functional Size
- Measurement
- Procedure
- Conceptual Model
- Software Size
- Design
- Validation
- Application
- Automation
- Rule
- Tool
- COSMIC
- IFPUG FPA
- MK II FPA

- NESMA FPA
- Method

Con los términos antes listados se elaboraron las siguientes consultas:

- Function Point
- (Function Point OR Functional Size OR Software Size) AND Measurement
- (COSMIC OR IFPUG FPA OR MK II FPA OR NESMA FPA) AND Measurement
- (COSMIC OR IFPUG FPA OR MK II FPA OR NESMA FPA) AND Measurement AND Rule
- (Function Point OR Functional Size OR Software Size) AND Measurement AND (Model OR Conceptual Model)
- (Function Point OR Functional Size OR Software Size) AND Measurement AND Conceptual Model AND (Procedure OR Method)
- (Function Point OR Functional Size OR Software Size) AND Measurement AND (Model OR Conceptual Model) AND (Design OR Validation OR Application OR Automation)
- (Function Point OR Functional Size OR Software Size) AND Measurement AND (Model OR Conceptual Model) AND Rule
- (Function Point OR Functional Size OR Software Size) AND Measurement AND (Model OR Conceptual Model) AND Tool
- (Function Point OR Functional Size OR Software Size) AND Measurement AND (Model OR Conceptual Model) AND (COSMIC OR IFPUG FPA OR MK II FPA OR NESMA FPA)

2) Fuentes de estudios

Para la revisión sistemática se utilizarán varias bases de datos digitales que proveen información de diferentes revistas técnicas y científicas, conferencias, workshops, proceedings de conferencias y reportes técnicos.

A continuación, se listan las fuentes utilizadas para buscar los estudios primarios:

- IEEE Xplore
- ACM Digital Library
- Springer Link
- Science Direct
- Google Scholar

Criterios de selección de estudios

Los criterios de selección (inclusión y exclusión de estudios) se determinan en base a los componentes de la pregunta de investigación (poblaciones, intervenciones, resultados y diseño de estudios). A continuación se describen los criterios de inclusión de estudios, los criterios de exclusión de estudios, y el proceso de selección de estudios.

1) Criterios de inclusión

Para que un estudio califique para ser incluido en la revisión sistemática debe cumplir con los siguientes criterios:

- El estudio corresponde a un caso de estudio, un experimento, un reporte de experiencias, o un reporte técnico.
- El estudio debe describir el uso de un procedimiento de medición de tamaño funcional en modelos conceptuales.
- El estudio presenta un procedimiento de medición de tamaño funcional en modelos conceptuales usado en la industria o en la academia.
- El estudio presenta correspondencias entre los conceptos de un método de medición y los conceptos de un modelo conceptual.

2) Criterios de exclusión

Los criterios de exclusión de estudios en la revisión sistemática son:

- Si el estudio no cumple con los criterios de inclusión.
- Si el estudio es un artículo de discusión, de opinión, de recopilación de trabajos (survey), de lecciones aprendidas, de keynote, o de introducción a conferencias, a proceedings de conferencias, a workshops y a libros.
- Si es un estudio ya seleccionado (debido a repeticiones de estudios encontrados en distintas búsquedas o diferentes estudios que aportan exactamente la misma contribución en términos del procedimiento de medición de tamaño funcional).

3) Proceso de selección

El proceso de selección es un proceso previo a la revisión, donde todos los estudios deben ser leídos. Para seleccionar un estudio para la revisión sistemática, el revisor lee el título y el resumen del estudio. Si la naturaleza del estudio no está clara en este momento, el revisor lee la introducción y la conclusión del estudio. Si la naturaleza del artículo aún no está clara, el revisor debe leer el artículo completo. Luego de la lectura, el

estudio será incluido si cumple con los criterios de inclusión y será excluido si satisface alguno de los criterios de exclusión.

Aseguramiento de la calidad de los estudios

El aseguramiento de la calidad de los estudios seleccionados se realiza en base al siguiente listado de preguntas:

- ¿El contexto de investigación ha sido descrito de manera adecuada?
- ¿El diseño del estudio es apropiado?
- ¿El estudio cumple con los objetivos planteados en ese estudio?
- ¿Los resultados del estudio se han logrado siguiendo criterios objetivos (no subjetivos)?
- ¿El estudio presenta evidencias originales para justificar la conclusión?

Estrategia de extracción de datos

La extracción de datos es el proceso de extraer información de los estudios primarios (estudios incluidos en la revisión). La extracción de datos es un proceso subjetivo que está propenso a errores. Para realizar este proceso de manera más objetiva se diseñó un formulario para extraer los datos (vea el Anexo A).

El formulario de extracción de datos consta de cinco secciones: una sección con información de la búsqueda, una sección con información general del estudio, una sección con información específica del estudio, una sección para anotar información sobre la calidad del estudio y una sección para poner información sobre otras características. Las dos últimas secciones permiten al revisor anotar libremente los comentarios que crea convenientes.

En cuanto a la información de la búsqueda, por cada estudio se almacena la fecha de la extracción de datos, la consulta realizada para encontrar el estudio, y la base de datos desde la cual se ha recuperado el estudio.

En cuanto a la información general de cada estudio, se almacena el título, los autores, el año de la publicación y la conferencia o proceedings donde fue publicada.

Finalmente, en la sección de información específica de cada estudio se realiza un pequeño resumen de la propuesta y se extrae la siguiente información:

1) Método de medición y versión del método en el que está basado el procedimiento.

2) Artefacto de software. El artefacto de software corresponde al modelo conceptual que se utiliza para realizar la medición. Las propuestas pueden utilizar uno o varios modelos conceptuales, los cuales pueden ser modelos utilizados en la fase de requisitos (casos de uso, diagramas de secuencia, etc.) o modelos utilizados en las fases de análisis y diseño (diagrama de clases, diagrama de estados, diagramas de interacción, etc.).

3) Diseño del procedimiento. Esta característica estará presente en aquellas propuestas que presenten un diseño sistemático del procedimiento de medición, incluyendo el proceso de aplicación del procedimiento y algún ejemplo de aplicación.

4) Validación del procedimiento. Esta característica estará presente en aquellas propuestas que presenten validaciones teóricas o empíricas del procedimiento de medición.

5) Herramienta. Esta característica estará presente en aquellas propuestas que presenten una herramienta que automatice (parcial o completamente) la aplicación de dicho procedimiento.

Síntesis de datos

La síntesis de los estudios será presentada utilizando una tabla para facilitar el entendimiento de la información recopilada. Esta tabla permite al lector ver las similitudes y diferencias entre los estudios recopilados en la revisión sistemática.

Los estudios serán agrupados según procedimiento de medición para obtener una visión completa de las características de cada procedimiento de medición (debido a que en un estudio se pueden presentar las reglas, en otro estudio se puede presentar la validación, etc.).

De esta manera, por cada procedimiento de medición, la tabla presentará información del método de medición base, del artefacto de software, del diseño del procedimiento, de la validación del procedimiento y de la automatización del procedimiento.

Calendario de la revisión sistemática

En el calendario de la revisión sistemática se especifican los hitos de la revisión y las fechas tentativas para su finalización. La Tabla 2 presenta el calendario de la revisión sistemática presentada en esta tesis.

Tabla 2. Calendario revisión sistemática.

Hito	Fecha Inicio	Fecha Finalización
Recuperación de estudios	01/05/2008	03/05/2008
Selección de estudios	04/05/2008	19/05/2008
Aseguramiento de la calidad de los estudios	21/05/2008	25/05/2008
Extracción de datos	28/05/2008	15/06/2008
Síntesis de datos	16/06/2008	20/06/2008
Reporte	23/06/2008	27/06/2008

2.2.2 Realizando la revisión

Esta fase es la más importante de la revisión sistemática. Esta fase se realiza cuando el protocolo para realizar la revisión ya ha sido diseñado, revisado y acordado entre las personas del equipo que realiza la revisión sistemática. Esta fase comprende cinco pasos: (1) identificación de la investigación, (2) selección de estudios primarios, (3) evaluación de la calidad de los estudios, (4) extracción y observación de datos, y (5) síntesis de datos.

Identificación de la investigación

La identificación de la investigación corresponde a la localización de los estudios primarios. El proceso de buscar y localizar estudios es un proceso complejo y que consume tiempo. Por esta razón, este proceso requiere dedicación, tiempo y una estrategia de búsqueda definida de manera objetiva.

El proceso de identificar los estudios primarios fue realizado de manera iterativa, y la estrategia de búsqueda fue redefinida a medida que se realizaban las iteraciones para hacerla más rigurosa y comprensiva, y para localizar la mayor cantidad de estudios primarios posible. El proceso de identificar los estudios primarios relevantes fue realizado desde el 01/05/2008 al 03/05/2008, incluidos ambos días. La búsqueda de estudios primarios fue realizada en las bases de datos digitales que estaban especificadas en el protocolo de la revisión.

Dado que la mayoría de los estudios están publicados en inglés, se realizaron las búsquedas en este lenguaje para localizar la mayor cantidad de estudios posibles. Además, mientras se realizaba la búsqueda se agregaron nuevos términos de búsqueda, por ejemplo se agregaron los términos Rules, Tool y Method. La lista de términos presentada en el protocolo de la revisión ya está actualizada con estos términos.

Cuando se realizan búsquedas en base de datos digitales, existe el riesgo de pasar por alto estudios relevantes debido a que algunas bases de datos pueden tener indexaciones incompletas. Para evitar esto, se debe realizar una búsqueda manual de los estudios. Sin embargo, como se describe en el protocolo de la revisión, esta revisión sistemática se realizó con consultas electrónicas en bases de datos digitales. Esta situación podría ser vista como una amenaza a la validez de la revisión sistemática presentada, ya que podrían no estar incluidos todos los estudios relevantes. Para disminuir esta amenaza, se han formulado las consultas de investigación de manera exhaustiva y se han seleccionado varias bases de datos.

Una vez que se han identificado los estudios primarios, se realiza la selección de estudios primarios. Este paso será detallado a continuación. Sin embargo, es importante destacar que la identificación y selección de estudios primarios se realizó de manera iterativa.

Selección de estudios primarios

El objetivo de la selección de estudios es identificar los estudios que responden a la pregunta de investigación formulada para la revisión sistemática. Es importante que la selección de estudios esté libre de aspectos subjetivos, por lo que se utilizaron los criterios de inclusión y exclusión definidos en el protocolo de la revisión. De esta manera, sólo fueron seleccionados los estudios que cumplen con todos los criterios de inclusión y ninguno de los criterios de exclusión.

El proceso de selección de estudios primarios fue realizado en varias etapas. Inicialmente se realizó la búsqueda en las bases de datos electrónicas con el término *function point*. Posteriormente los resultados recuperados fueron restringidos utilizando las consultas definidas en el protocolo de la revisión para identificar un conjunto de estudios candidatos. El proceso de selección fue

realizado desde el 04/05/2008 al 19/05/2008, ambas fechas incluidas.

La siguiente tabla resume los estudios recuperados en las búsquedas y los estudios candidatos seleccionados en cada base de datos.

Tabla 3. Estudios recuperados e identificados.

Base de datos	Estudios recuperados	Estudios identificados
IEEE Xplore	160	48
ACM Digital Library	115279	104
Springer Link	548361	141
Science Direct	122960	62
Google Scholar	5060000	824
Total	**5850910**	**1179**

Luego los estudios identificados como candidatos fueron analizados según el proceso de selección descrito en el protocolo de revisión para evaluar el cumplimiento de los criterios de inclusión y exclusión. Además, es importante destacar que sólo fueron seleccionados aquellos estudios que estaban disponibles electrónicamente. La búsqueda se realizó en el orden en que se listan las bases de datos, por lo que los estudios que fueron seleccionados desde la base de datos inicial fueron excluidos de los estudios identificados nuevamente en las bases de datos posteriores sucesivamente. La Tabla 4 resume los estudios seleccionados en cada base de datos.

Tabla 4. Estudios seleccionados.

Base de datos	Estudios seleccionados
IEEE Xplore	11
ACM Digital Library	2
Springer Link	4
Science Direct	2
Google Scholar	23
Total	**42**

Para facilitar el análisis, almacenamiento y la recuperación de los estudios seleccionados se ha utilizado el gestor de referencias JabRef. En el Anexo B se listan los 42 estudios seleccionados, agrupados por la base de datos desde la cual fueron recuperados.

Evaluación de la calidad de los estudios

Mientras se realizaba la selección de estudios primarios, se realizó un aseguramiento de la calidad basado en la pertinencia de los estudios para asegurar un mínimo nivel de calidad. Luego, a los estudios que cumplían con el nivel de calidad mínimo le fueron aplicados los criterios de inclusión y exclusión. De esta manera, los 42 estudios seleccionados presentan el nivel mínimo de calidad.

Posteriormente, se analizó individualmente el nivel de calidad de cada uno de los estudios seleccionados. Este proceso se realizó desde el 21/05/2008 al 25/05/2008, ambas fechas incluidas.

Para asegurar el nivel de calidad de cada estudio seleccionado se aplicaron las preguntas especificadas en el protocolo de la revisión para el aseguramiento de la calidad. Los resultados obtenidos para cada pregunta se detallan a continuación:

- ¿El contexto de investigación ha sido descrito de manera adecuada?
  Para cada estudio seleccionado se revisó si el contexto de la investigación se había descrito de manera adecuada. En los 42 estudios seleccionados se describe correctamente el contexto de la investigación, el cual generalmente es presentado en el resumen y en la introducción del estudio. Por esta razón, los 42 estudios seleccionados son adecuados para la revisión sistemática.

- ¿El diseño del estudio es apropiado?
  El diseño de los 42 estudios seleccionados corresponde a reportes técnicos. Por esta razón, el diseño de los estudios es adecuado para la revisión sistemática.

- ¿El estudio cumple con los objetivos de ese estudio?
  La mayoría de los estudios seleccionados cumple con los objetivos planteados en el estudio. Sin embargo, la propuesta de Levesque et al. [Levesque08] no cumple los objetivos planteados, que consistían en automatizar la medición del tamaño funcional ya que no presentan ninguna herramienta. Sin embargo, presentan intentos para automatizar el procedimiento de medición y una sección en que discuten los resultados obtenidos. Por esta razón, el estudio de Levesque et al. se ha considerado con calidad suficiente para ser incluido en la revisión sistemática.

- ¿Los resultados del estudio se han logrado siguiendo criterios objetivos?
  Cada una de las propuestas seleccionadas muestra como satisface sus objetivos planteados siguiendo criterios objetivos. Por este motivo, los 42 estudios seleccionados son adecuados para la revisión sistemática.

- ¿El estudio presenta evidencias originales para justificar la conclusión?
  La mayoría de los estudios presenta suficiente evidencia original para justificar la inclusión del estudio en la revisión sistemática. Sin embargo, las propuestas de Abrahao et al. [Abrahao06] y Diab et al. [Diab01] no presentan suficiente evidencia original del procedimiento de medición, por lo que no han sido considerados en la revisión sistemática.

Finalmente, luego de asegurar el nivel de calidad de los estudios seleccionados, se ha decidido incluir 40 estudios en la revisión sistemática y excluir 2 estudios.

Extracción y observación de datos

El proceso de extracción y observación de datos corresponde al proceso de obtener información desde los estudios primarios seleccionados. Este proceso es subjetivo, y para minimizar sesgos en la extracción de la información se ha utilizado el formulario diseñado en la definición del protocolo de la revisión (vea el Anexo A).

Por cada estudio seleccionado se rellenó el formulario de extracción de datos. Este proceso fue realizado entre el 28/05/2008 al 15/06/2008, ambas fechas incluidas. Antes de extraer la información, los estudios fueron ordenados según su año de publicación. A continuación se presentan los datos extraídos por cada estudio seleccionado, los cuales están ordenados por el año de publicación y alfabéticamente por el primer autor del estudio.

Propuesta de Fetcke et al. (1997)

1) Información de la Búsqueda
   - Fecha de extracción de datos: 28/05/2008
   - Términos utilizados en la búsqueda: Function Point Measurement

- Base de datos donde se encontró: IEEE Xplore

2) Información General
   - Título: Mapping the OO-Jacobson Approach into Function Point Analysis.
   - Autores: Fetcke, T.; Abran, A.; Nguyen, T.
   - Año publicación: 1997
   - Lugar de publicación: 23th Technology of Object-Oriented Languages and Systems (TOOLS 1997).

3) Información Específica
   - Resumen: Fetcke et al. [Fetcke97] presentan un procedimiento de medición basado en IFPUG FPA para medir el tamaño funcional de aplicaciones modeladas con el método OOSE [Jacobson92].Estos autores no presentan un diseño sistemático del procedimiento de medición, pero sí presentan una correspondencia detallada entre los conceptos de IFPUG FPA y las primitivas conceptuales de los modelos del método OOSE. Además, estos autores aplican su propuesta a tres proyectos reales.
   - Método de medición y versión: IFPUG FPA 4.0
   - Artefacto de software: Modelo de casos de uso OOSE, modelo de objetos OOSE, y modelo de análisis de OOSE.
   - Diseño del procedimiento: No presenta diseño.
   - Validación del procedimiento: No presenta validación.
   - Herramienta: No presenta herramienta.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio describe correctamente el contexto de la investigación, describiendo en una sección los trabajos relacionados, en otra sección la utilización del método de medición IFPUG FPA en métodos de diseño orientados a objetos y en otra sección el método OOSE de manera resumida. Posteriormente presenta las reglas de correspondencias. Además, el estudio cumple con los objetivos de la propuesta y reconoce las limitaciones que tiene respecto de otras propuestas.

Propuesta de Gramantieri et al. (1997)

1) Información de la Búsqueda
   - Fecha de extracción de datos: 28/05/2008
   - Términos utilizados en la búsqueda: MK II Function Point Measurement
   - Base de datos donde se encontró: Google Scholar

2) Información General
    - Título: A System for Measuring Function Points from Specifications.
    - Autores: Gramantieri, F.; Lamma, E.; Riguzzi, F.; Mello, P.
    - Año publicación: 1997
    - Lugar de publicación: No encontrado.

3) Información Específica
    - Resumen: Gramantieri et al. [Gramantieri97] presentan un procedimiento de medición basado en IFPUG FPA para medir el tamaño funcional de aplicaciones modeladas con diagramas de Entidad-Relación y diagramas de flujo de datos. El artefacto de entrada utilizado por estos autores es una integración de los diagramas de Entidad-Relación y los diagramas de flujo de datos denominada ER-DFD. Estos autores presentan el proceso de medición y un conjunto de reglas formales que permiten calcular el tamaño funcional de las aplicaciones modeladas. Estas reglas formales han sido automatizadas para obtener el tamaño funcional de las aplicaciones modeladas. Además, estos autores aplican su propuesta a un caso de estudio.
    - Método de medición y versión: IFPUG FPA versión 4.0
    - Artefacto de software: Diagrama ER-DFD.
    - Diseño del procedimiento: No presenta diseño.
    - Validación del procedimiento: No presenta validación.
    - Herramienta: Tiene una herramienta automática.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio describe correctamente el contexto de la investigación, mencionando los trabajos relacionados durante la introducción. Luego describe el proceso de medición de tamaño funcional. En otra sección presenta una breve descripción de los modelos ER-DFD. El estudio cumple con los objetivos planteados en la propuesta.

Propuesta de Shoval et al. (1997)

1) Información de la Búsqueda
    - Fecha de extracción de datos: 29/05/2008
    - Términos utilizados en la búsqueda: MK II Function Point Measurement
    - Base de datos donde se encontró: Google Scholar

2) Información General
   - Título: A Combination of the Mk-II Function Points Software Estimation Method with the ADISSA Methodology for Systems Analysis and Design.
   - Autores: Shoval, P.; Feldman, O.
   - Año publicación: 1997
   - Lugar de publicación: Journal of Information and Software Technology, Vol(39) Nro(13), 1997.

3) Información Específica
   - Resumen: Shoval et al. [Shoval97] presentan un procedimiento de medición basado en MK II FPA para medir el tamaño funcional de aplicaciones modeladas con el método ADISSA [Shoval88]. Estos autores no presentan un diseño sistemático del procedimiento de medición. Sin embargo, presentan las correspondencias entre los conceptos de MK II FPA y los conceptos de los diagramas de flujos de datos para la medición del tamaño funcional. Además, estos autores presentan el proceso de aplicación del procedimiento y una herramienta que permite medir de manera semi-automática los diagramas DFD que especifican un sistema con el método ADISSA.
   - Método de medición y versión: MK II FPA 4.0
   - Artefacto de software: Diagrama de flujo de datos.
   - Diseño del procedimiento: No presenta diseño.
   - Validación del procedimiento: No presenta validación.
   - Herramienta: Tiene una herramienta semi-automática.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio describe correctamente el contexto de la investigación, mencionando los trabajos relacionados y los diferentes enfoques para medir costo y esfuerzo. Además, se describen métodos de estimación basados en funcionalidad. En otra sección presenta el método ADISSA y luego presenta las reglas para aplicar MK II en ADISSA y una herramienta de estimación. El estudio cumple con los objetivos planteados en la propuesta.

Propuesta de Caldiera et al. (1998)

1) Información de la Búsqueda
   - Fecha de extracción de datos: 29/05/2008
   - Términos utilizados en la búsqueda: Function Point Measurement
   - Base de datos donde se encontró: IEEE Xplore

2) Información General
   - Título: Definition and Experimental Evaluation of Function Points for Object-Oriented Systems.
   - Autores: Caldiera, G.; Antoniol, G.; Fiutem, R.; Lokan, C.
   - Año publicación: 1998
   - Lugar de publicación: 5th International Symposium on Software Metrics (METRICS 1998).

3) Información Específica
   - Resumen: Caldiera et al. [Caldiera98] presentan un procedimiento de medición basado en IFPUG FPA para medir el tamaño funcional de aplicaciones orientadas a objeto modeladas con el método OMT [Rumbaugh91]. Estos autores no presentan un diseño sistemático del procedimiento de medición. No obstante, presentan el proceso de aplicación del procedimiento y las correspondencias entre los conceptos de IFPUG FPA y el modelo de objetos del método OMT. El procedimiento es aplicado a un caso de estudio. Además, este procedimiento tiene una herramienta que automatiza su aplicación, con la ayuda de la cual el procedimiento ha sido aplicado en casos reales.
   - Método de medición y versión: IFPUG FPA versión 4.0
   - Artefacto de software: Modelo de objetos OMT.
   - Diseño del procedimiento: No presenta diseño.
   - Validación del procedimiento: No presenta validación.
   - Herramienta: Tiene una herramienta automática.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio describe correctamente el contexto de la investigación, mencionando los trabajos relacionados detalladamente en una sección. También, el estudio cumple con los objetivos planteados, presentando una adaptación de IFPUG para aplicarla a modelos orientados a objeto.

5) Otras Notas

El procedimiento recibe el nombre OOFP (Object Oriented Function Points).

Propuesta de Bévo et al. (1999)

1) Información de la Búsqueda
   - Fecha de extracción de datos: 30/05/2008
   - Términos utilizados en la búsqueda: COSMIC Software

Measurement
- Base de datos donde se encontró: Google Scholar

2) Información General
- Título: Application de la méthode FFP à partir d'une spécification selon la notation UML: compte rendu des premiers essais d'application et questions.
- Autores: Bévo, V.; Lévesque, G.; Abran, A.
- Año publicación: 1999
- Lugar de publicación: 9th International Workshop Software Measurement (IWSM 1999).

3) Información Específica
- Resumen: Bévo et al. [Bevo99] presentan un procedimiento de medición basado en COSMIC FFP para medir el tamaño funcional de aplicaciones modeladas con UML. Estos autores no presentan un diseño sistemático del procedimiento de medición. Sin embargo, presentan las correspondencias entre los conceptos de COSMIC FFP y los conceptos de los diagramas de UML y la aplicación del procedimiento en un caso de estudio.
- Método de medición y versión: COSMIC FFP versión 2.0
- Artefacto de software: Diagrama de casos de uso UML 1.0, escenarios UML 1.0 y diagrama de clases UML 1.0.
- Diseño del procedimiento: No presenta diseño.
- Validación del procedimiento: No presenta validación.
- Herramienta: No presenta herramienta.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta un resumen de los conceptos de los diagramas de casos de uso, de escenarios y de artefactos. Finalmente, este estudio cumple con los objetivos planteados, que correspondían a aplicar el método de medición COSMIC FFP a diagramas UML.

Propuesta de Uemura et al. (1999)

1) Información de la Búsqueda
- Fecha de extracción de datos: 30/05/2008
- Términos utilizados en la búsqueda: Function Point Measurement
- Base de datos donde se encontró: IEEE Xplore

2) Información General
- Título: Function Point Measurement Tool for UML Design

Specification.
- Autores: Uemura, T.; Kusumoto, S.; Inoue, K.
- Año publicación: 1999
- Lugar de publicación: 6th International Symposium on Software Metrics (METRICS 1999).

3) Información Específica
- Resumen: Uemura et al. [Uemura99] presentan un procedimiento de medición basado en IFPUG FPA para medir el tamaño funcional de aplicaciones modeladas con la herramienta Rational Rose. Para esto, estos autores utilizan como artefacto de entrada los diagramas de secuencia y los diagramas de clases UML. Estos autores no presentan un diseño sistemático del procedimiento de medición, pero presentan las correspondencias entre los conceptos de IFPUG FPA y los conceptos de los diagramas se secuencia y los diagramas de clases. Además, estos autores presentan el proceso de aplicación del procedimiento y una herramienta que permite medir de manera automática las aplicaciones.
- Método de medición y versión: IFPUG FPA versión 4.0
- Artefacto de software: Diagrama de casos de uso UML 1.1 y diagrama de clases UML 1.1.
- Diseño del procedimiento: No presenta diseño.
- Validación del procedimiento: No presenta validación.
- Herramienta: Tiene una herramienta automática.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio describe correctamente el contexto de la investigación, detallando en una sección de preliminares el método de medición FPA, la versión de IFPUG FPA, y los conceptos claves de los diagramas de clases y de secuencia UML. Finalmente, este estudio cumple con los objetivos de la propuesta.

Propuesta de Kusumoto et al. (2000)

1) Información de la Búsqueda
- Fecha de extracción de datos: 31/05/2008
- Términos utilizados en la búsqueda: Function Point Measurement
- Base de datos donde se encontró: IEEE Xplore

2) Información General
- Título: Function Point Measurement for Object-Oriented

Requirements Specification.
- Autores: Kusumoto, S.; Inoue, K.; Kasimoto, T.; Suzuki, A.; Yuura, K.; Tsuda, M.
- Año publicación: 2000
- Lugar de publicación: 24th International Computer Software and Applications Conference (COMPSAC 2000).

3) Información Específica
- Resumen: Kusumoto et al. [Kusumoto00] presentan un procedimiento de medición basado en IFPUG FPA para medir el tamaño funcional de sistemas a partir de sus especificaciones de requisitos. Esta propuesta utiliza como artefacto de entrada los modelos de requisitos de un sistema orientado a objetos para el análisis de requisitos llamado REQUARIO [Saito95]. Estos autores no presentan un diseño sistemático del procedimiento de medición, pero presentan la correspondencia entre los conceptos de IFPUG FPA y las primitivas conceptuales de REQUARIO. Además, estos autores presentan el proceso de aplicación de su propuesta. En cuanto a la automatización, estos autores presentan una herramienta que permite medir automáticamente el tamaño funcional de las especificaciones de requisitos. Esta propuesta ha sido validada manualmente por expertos, mediante la aplicación a casos reales de la empresa Hitachi, y posteriormente comparando los resultados obtenidos por la herramienta con los resultados obtenidos por los expertos.
- Método de medición y versión: IFPUG FPA versión 4.0
- Artefacto de software: Modelo de requisitos de REQUARIO.
- Diseño del procedimiento: No presenta diseño.
- Validación del procedimiento: Validación de las medidas por expertos.
- Herramienta: Tiene una herramienta automática.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta detalladamente el método de medición IFPUG FPA y las especificaciones de requisitos con REQUARIO. Este estudio cumple con los objetivos planteados, que consistían en aplicar el método de medición IFPUG FPA a las especificaciones de requisitos de la empresa Hitachi Ltda.

Propuesta de Ram et al. (2000)

1) Información de la Búsqueda
    - Fecha de extracción de datos: 31/05/2008
    - Términos utilizados en la búsqueda: Function Point Measurement
    - Base de datos donde se encontró: Google Scholar

2) Información General
    - Título: Object Oriented Design Function Points.
    - Autores: Ram, D.J.; Raju, S.
    - Año publicación: 2000
    - Lugar de publicación: 1st Asia-Pacific Conference on Quality Software, 2000.

3) Información Específica
    - Resumen: Ram et al. [Ram00] presentan un procedimiento de medición basado en IFPUG FPA para medir el tamaño funcional de sistemas orientados a objetos. Los autores de OODFP no presentan un diseño sistemático del procedimiento de medición, pero sí presentan una correspondencia detallada entre los conceptos de IFPUG FPA y las primitivas conceptuales del diseño orientado a objetos. Además, estos autores aplican su propuesta a un caso de estudio.
    - Método de medición y versión: IFPUG FPA versión 4.1
    - Artefacto de software: Modelo de clases.
    - Diseño del procedimiento: No presenta diseño.
    - Validación del procedimiento: No presenta validación.
    - Herramienta: No presenta herramienta.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta detalladamente el contexto de la investigación y los trabajos relacionados. Este estudio cumple con los objetivos planteados, que consistían en desarrollar un procedimiento de medición de tamaño funcional que permita medir el tamaño funcional de aplicaciones en la fase de diseño.

5) Otras Notas

El procedimiento recibe el nombre OODFP (Object Oriented Design Function Points).

Propuesta de Diab et al. (2001)

1) Información de la Búsqueda

- Fecha de extracción de datos: 31/05/2008
- Términos utilizados en la búsqueda: COSMIC Function Point Measurement
- Base de datos donde se encontró: Google Scholar

2) Información General
   - Título: A formal definition of COSMIC FFP for automated measurement of ROOM specifications.
   - Autores: Diab, H.; Frappier, M.; St-Denis, R.
   - Año publicación: 2001
   - Lugar de publicación: 4th European Conference on Software Measurement and ICT Control (FESMA 2001).

3) Información Específica
   - Resumen: Diab et al. [Diab01a] presentan un procedimiento de medición basado en COSMIC FFP para medir el tamaño funcional de aplicaciones de tiempo real. Este procedimiento se compone de reglas de medición formales que permiten medir el tamaño funcional de aplicaciones especificadas con ROOM (Real-time Object Oriented Modelling) [Selic94]. Estos autores no presentan un diseño sistemático del procedimiento de medición, pero sí presentan una correspondencia detallada entre los conceptos de COSMIC FFP y ROOM.
   - Método de medición y versión: COSMIC FFP versión 2.2
   - Artefacto de software: Modelo RRRT (especificaciones ROOM).
   - Diseño del procedimiento: No presenta diseño.
   - Validación del procedimiento: No presenta validación.
   - Herramienta: No presenta herramienta.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta detalladamente el método de medición COSMIC y las especificaciones del lenguaje ROOM. Este estudio cumple con los objetivos planteados, que consistían en aplicar el método de medición COSMIC FFP a especificaciones para aplicaciones de tiempo real.

Propuesta de Uemura et al. (2001)

1) Información de la Búsqueda
   - Fecha de extracción de datos: 01/06/2008
   - Términos utilizados en la búsqueda: FPA Measurement Design
   - Base de datos donde se encontró: Google Scholar

2) Información General
    - Título: Function Point Analysis using Design Specifications based on the Unified Modelling Language.
    - Autores: Uemura, T.; Kusumoto, S.; Inoue, K.
    - Año publicación: 2001
    - Lugar de publicación: Journal of Software Maintenance and Evolution: Research and Practice, Vol(13) Nro(4), 2001.

3) Información Específica
    - Resumen: Uemura et al. [Uemura01] presentan un procedimiento de medición basado en IFPUG FPA para medir el tamaño funcional de aplicaciones modeladas con la herramienta Rational Rose. Estos autores presentan el proceso de aplicación del procedimiento y una herramienta que permite medir de manera automática las aplicaciones. Finalmente estos autores presentan la validación de los resultados obtenidos por la herramienta en dos casos de estudio con los resultados obtenidos por expertos.
    - Método de medición y versión: IFPUG FPA versión 4.0
    - Artefacto de software: Diagrama de casos de uso UML 1.1 y diagrama de clases UML 1.1.
    - Diseño del procedimiento: No presenta diseño.
    - Validación del procedimiento: Validación de las medidas por expertos.
    - Herramienta: Tiene una herramienta automática.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio describe correctamente el contexto de la investigación, detallando en una sección de preliminares el método de medición FPA, la versión de IFPUG FPA utilizada, y los conceptos claves de los diagramas de clases y de secuencia UML. Este estudio es similar al estudio presentado por Uemura et al. en 1999 [Uemura99], sin embargo presenta la validación de la herramienta en dos casos de estudio como novedad. Dado que el estudio presenta la validación de la herramienta como evidencia original, este estudio ha sido incluido en la revisión. Finalmente, este estudio cumple con los objetivos planteados.

Propuesta de Abrahao et al. (2002)

1) Información de la Búsqueda
    - Fecha de extracción de datos: 01/06/2008

- Términos utilizados en la búsqueda: Functional Size Measurement
- Base de datos donde se encontró: Google Scholar

2) Información General
   - Título: A Methodology for Evaluating Quality and Functional Size of Operative WebApps.
   - Autores: Abrahao, S.; Olsina, L.; Pastor, O.
   - Año publicación: 2002
   - Lugar de publicación: 2nd International Workshop on Web Oriented Software Technology (IWWOST 2002).

3) Información Específica
   - Resumen: Abrahao et al. [Abrahao02] presentan una propuesta para medir el tamaño funcional de aplicaciones web a partir de modelos conceptuales utilizando IFPUG FPA. Estos autores utilizan como artefacto de entrada el modelo navegacional del método de ingeniería web OOWS [Fons03]. Estos autores no presentan un diseño riguroso del procedimiento de medición, aunque sí presentan las correspondencias entre los conceptos de IFPUG FPA y el modelo navegacional OOWS.
   - Método de medición y versión: IFPUG FPA versión 4.1
   - Artefacto de software: Modelo navegacional OOWS.
   - Diseño del procedimiento: No presenta diseño.
   - Validación del procedimiento: No presenta validación.
   - Herramienta: No presenta herramienta.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta el contexto de la investigación de manera detallada durante la introducción, y luego presenta un resumen de las fases de WEBFP_QEM (una metodología para analizar y evaluar la calidad de aplicaciones web complejas que ya estén operativas). El estudio cumple con los objetivos planteados, que correspondían a asegurar la calidad de aplicaciones web operativas, considerando los aspectos funcionales y los aspectos no funcionales.

5) Otras Notas

El procedimiento recibe el nombre OOmFPWeb (OO-Method Function Points for the Web).

Propuesta de Tavares et al. (2002)

1) Información de la Búsqueda
    - Fecha de extracción de datos: 02/06/2008
    - Términos utilizados en la búsqueda: Function Point Measurement
    - Base de datos donde se encontró: Google Scholar

2) Información General
    - Título: Medicao de Pontos por Funcao a partir da Especificacao de Requisitos.
    - Autores: Tavares, H.; Carvalho, A.; Castro, J.
    - Año publicación: 2002
    - Lugar de publicación: Workshop on Requirements Engieneering (WER 2002).

3) Información Específica
    - Resumen: Tavares et al. [Tavares02] presentan un procedimiento de medición basado en IFPUG FPA para medir el tamaño funcional de aplicaciones orientadas a objeto a partir de su especificación de requisitos. Para esto, los autores utilizan como artefacto de entrada el diagrama de casos de uso UML y el diagrama de clases. Estos autores no presentan un diseño sistemático del procedimiento de medición. Sin embargo, presentan la correspondencia de conceptos de IFPUG FPA y los diagramas de casos de uso y diagramas de clases UML. Además, estos autores presentan el procedimiento mediante su aplicación a un caso de estudio. Este procedimiento ha sido aplicado a casos reales de la empresa SERPRO, que tiene una herramienta que automatiza su aplicación.
    - Método de medición y versión: IFPUG FPA versión 4.1.1
    - Artefacto de software: Diagrama de casos de uso UML y diagrama de clases UML.
    - Diseño del procedimiento: No presenta diseño.
    - Validación del procedimiento: No presenta validación.
    - Herramienta: Tiene una herramienta.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta el contexto de la investigación durante la introducción, y luego presenta detalladamente los trabajos relacionados y un resumen de IFPUG FPA. El estudio cumple con los objetivos planteados, que correspondían a proponer un procedimiento de medición a partir de casos de uso.

Propuesta de Antoniol et al. (2003)

1) Información de la Búsqueda
   - Fecha de extracción de datos: 02/06/2008
   - Términos utilizados en la búsqueda: Function Point Measurement Validation
   - Base de datos donde se encontró: Google Scholar

2) Información General
   - Título: Object-Oriented Function Points: An Empirical Validation.
   - Autores: Antoniol, G.; Fiutem, R.; Lokan, C.
   - Año publicación: 2003
   - Lugar de publicación: Journal of Empirical Software Engineering, Vol(8) Nro(3), 2003.

3) Información Específica
   - Resumen: Antoniol et al. [Antoniol03] presentan una validación empírica para el procedimiento de medición presentado por Caldiera et al. [Caldiera98]. La validación empírica que presentan corresponde a la comparación de los resultados obtenidos con valores base. Además, estos autores presentan lecciones aprendidas sobre la estimación del tamaño usando la propuesta OOFP.
   - Método de medición y versión: IFPUG FPA versión 4.0
   - Artefacto de software: Modelo de objetos OMT.
   - Diseño del procedimiento: No presenta diseño.
   - Validación del procedimiento: Validación Empírica.
   - Herramienta: No presenta herramienta.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio describe correctamente el contexto de la investigación, mencionando detalladamente los trabajos relacionados y el procedimiento de medición definido por Caldiera et al. [Caldiera98]. Este estudio cumple con los objetivos de la propuesta, respondiendo a las preguntas de investigación planteadas.

5) Otras Notas

La validación es realizada al procedimiento OOFP (Object Oriented Function Points).

Propuesta de Bertolami et al. (2003)

1) Información de la Búsqueda
    - Fecha de extracción de datos: 02/06/2008
    - Términos utilizados en la búsqueda: Functional Size Measurement Conceptual Model
    - Base de datos donde se encontró: Google Scholar

2) Información General
    - Título: Análisis de Puntos Función en la Elicitación de Requerimientos.
    - Autores: Bertolami, M.; Oliveros, A.
    - Año publicación: 2003
    - Lugar de publicación: 6th Workshop on Requirements Engineering (WER 2003).

3) Información Específica
    - Resumen: Bertolami et al. [Bertolami03] presentan una propuesta para medir el tamaño funcional a partir de escenarios descritos en Léxico Extendido del Lenguaje (LEL) utilizando MK II FPA. Estos autores utilizan como artefacto de entrada los escenarios, y mediante la aplicación de una serie de reglas es posible obtener el tamaño funcional. Estos autores no presentan un diseño sistemático del procedimiento, sin embargo, indican el proceso de aplicación de la propuesta. Esta propuesta ha sido aplicada en cuatro casos de estudio. Este procedimiento no está automatizado.
    - Método de medición y versión: MK II FPA versión 1.3.1
    - Artefacto de software: Escenarios descritos en Léxico Extendido del Lenguaje.
    - Diseño del procedimiento: No presenta diseño.
    - Validación del procedimiento: No presenta validación.
    - Herramienta: No presenta herramienta.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta el contexto de la investigación durante la introducción, y luego presenta detalladamente los trabajos relacionados y un resumen de Léxico Extendido del Lenguaje (LEL) y escenarios. Posteriormente presenta un resumen de MK II. El estudio cumple con los objetivos planteados, que correspondían a disponer de una estimación del tamaño del producto de software a construir en una etapa muy temprana del proyecto de desarrollo.

Propuesta de Nagano et al. (2003)

1) Información de la Búsqueda
    - Fecha de extracción de datos: 03/06/2008
    - Términos utilizados en la búsqueda: COSMIC Measurement
    - Base de datos donde se encontró: Google Scholar

2) Información General
    - Título: Functional Metrics using COSMIC FFP for Object-Oriented Real-Time Systems.
    - Autores: Nagano, S.; Ajisaka, T.
    - Año publicación: 2003
    - Lugar de publicación: 13th International Workshop on Software Measurement (IWSM 2003).

3) Información Específica
    - Resumen: Nagano et al. [Nagano03] presentan una propuesta para medir el tamaño funcional de aplicaciones de tiempo real utilizando COSMIC FFP. Para modelar estas aplicaciones utilizan el método Shlaer-Mellor [Shlaer92] y especificaciones xUML (UML ejecutable) [Mellor02]. Estos autores no presentan un diseño sistemático del procedimiento, aunque presentan guías para identificar los conceptos de COSMIC en las especificaciones de los diagramas de clase, de estado y de colaboración realizadas con UML ejecutable. Este procedimiento ha sido aplicado a un caso de estudio de manera manual, sin embargo concluyen que este procedimiento puede ser automatizado.
    - Método de medición y versión: COSMIC FFP versión 2.0
    - Artefacto de software: Diagrama de clases, diagrama de estado y diagrama de colaboración, todos ellos especificados con xUML.
    - Diseño del procedimiento: No presenta diseño.
    - Validación del procedimiento: No presenta validación.
    - Herramienta: No presenta herramienta.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta el contexto de la investigación durante la introducción. Este estudio no presenta trabajos relacionados y tiene pocas referencias. Sin embargo, presenta el procedimiento de medición de manera clara y evalúa los resultados del procedimiento con los resultados obtenidos desde especificaciones textuales. Finalmente, este estudio cumple con los objetivos planteados.

Propuesta de Poels (2003-1)

1) Información de la Búsqueda
    - Fecha de extracción de datos: 03/06/2008
    - Términos utilizados en la búsqueda: Function Point Measurement Validation
    - Base de datos donde se encontró: Google Scholar

2) Información General
    - Título: Definition and Validation of a COSMIC-FFP Functional Size Measure for Object-Oriented Systems.
    - Autores: Poels G.
    - Año publicación: 2003
    - Lugar de publicación: 7th International ECOOP Workshop QAOOSE (QAOOSE 2003).

3) Información Específica
    - Resumen: Poels [Poels03a] presenta un procedimiento de medición basado en COSMIC FFP para medir el tamaño funcional de aplicaciones orientadas a objeto modeladas con el método basado en eventos MERODE [Dedene94]. Este autor no presenta un diseño sistemático del procedimiento de medición. Sin embargo, presenta la correspondencia de conceptos de COSMIC y el modelo de negocios y el modelo de servicios de MERODE. El modelo de negocios se compone del diagrama de clases, tablas evento-objeto, y diagramas de transición de estados. El modelo de servicios especifica la generación de eventos por el usuario y su transmisión al modelo de negocio.
    - Método de medición y versión: COSMIC FFP versión 2.1
    - Artefacto de software: Modelo de negocio MERODE (diagrama de clases, tablas evento-objeto, y diagramas de transición de estados) y modelo de servicios MERODE.
    - Diseño del procedimiento: No presenta diseño.
    - Validación del procedimiento: Validación teórica del procedimiento.
    - Herramienta: No presenta herramienta.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta el contexto de la investigación durante la introducción, y luego presenta detalladamente los trabajos relacionados en una sección. Posteriormente presenta un resumen con los conceptos de COSMIC FFP. El estudio cumple con los objetivos planteados, que

correspondían a validar teóricamente COSMIC en basándose en distancias de las mediciones [Poels00].

Propuesta de Poels (2003-2)

1) Información de la Búsqueda
   - Fecha de extracción de datos: 03/06/2008
   - Términos utilizados en la búsqueda: Functional Size Measurement Conceptual Model
   - Base de datos donde se encontró: Google Scholar

2) Información General
   - Título: Functional Size Measurement of Multi-Layer Object-Oriented Conceptual Models.
   - Autores: Poels G.
   - Año publicación: 2003
   - Lugar de publicación: 9th International Object-Oriented Information Systems Conference.

3) Información Específica
   - Resumen: Poels [Poels03b] presenta un procedimiento de medición basado en COSMIC FFP para medir el tamaño funcional de aplicaciones multicapas modeladas con el método basado en eventos MERODE [Dedene94]. Este autor no presenta un diseño sistemático del procedimiento de medición, pero presenta las correspondencias de conceptos de COSMIC y el modelo de negocios y el modelo de servicios de MERODE.
   - Método de medición y versión: COSMIC FFP versión 2.1
   - Artefacto de software: Modelo de negocio MERODE (diagrama de clases, tablas evento-objeto, y diagramas de transición de estados) y modelo de servicios MERODE.
   - Diseño del procedimiento: No presenta diseño.
   - Validación del procedimiento: No presenta validación.
   - Herramienta: No presenta herramienta.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta el contexto de la investigación durante la introducción, y luego presenta detalladamente los trabajos relacionados en una sección. Posteriormente presenta un resumen con los conceptos de COSMIC FFP y en otra sección presenta la arquitectura en capas del método MERODE. El estudio cumple con los objetivos planteados, que correspondían a aplicar COSMIC a modelos conceptuales organizados en una arquitectura de capas.

Propuesta de Abrahao et al. (2004-1)

1) Información de la Búsqueda
    - Fecha de extracción de datos: 04/06/2008
    - Términos utilizados en la búsqueda: Function Point Measurement
    - Base de datos donde se encontró: IEEE Xplore

2) Información General
    - Título: Assessing the reproducibility and accuracy of functional size measurement methods through experimentation.
    - Autores: Abrahao, S.; Poels, G.; Pastor, O.
    - Año publicación: 2004
    - Lugar de publicación: Proceedings of the International Symposium on Empirical Software Engineering (ISESE 2004).

3) Información Específica
    - Resumen: Abrahao et al. [Abrahao04a] presentan un experimento controlado para comparar los resultados de aplicar FPA y el procedimiento de medición OOmFP definido por los mismos autores. El experimento evalúa la reproducibilidad y la exactitud de OOmFP, resultando el procedimiento OOmFP más consistente y exacto que IFPUG FPA.
    - Método de medición y versión: IFPUG FPA versión 4.1
    - Artefacto de software: Modelo de objetos OO-Method, modelo funcional OO-Method, modelo dinámico OO-Method y modelo de presentación OO-Method.
    - Diseño del procedimiento: No presenta diseño.
    - Validación del procedimiento: Validación empírica con respecto a la reproducibilidad y exactitud del procedimiento.
    - Herramienta: No presenta herramienta.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta detalladamente los trabajos relacionados, y presenta un resumen del método de medición IFPUG FPA y del procedimiento de medición OOmFP. Este estudio cumple con los objetivos planteados, que consistían en evaluar OOmFP mediante un experimento controlado.

5) Otras Notas

La validación se realiza al procedimiento OOmFP (OO-Method Function Points).

Propuesta de Abrahao et al. (2004-2)

1) Información de la Búsqueda
   - Fecha de extracción de datos: 04/06/2008
   - Términos utilizados en la búsqueda: Functional Size Measurement
   - Base de datos donde se encontró: IEEE Xplore

2) Información General
   - Título: Evaluating a functional size measurement method for Web applications: an empirical analysis.
   - Autores: Abrahao, S.; Poels, G.; Pastor, O.
   - Año publicación: 2004
   - Lugar de publicación: 10th International Symposium on Software Metrics (METRICS 2004).

3) Información Específica
   - Resumen: Abrahao et al. [Abrahao04b] presentan un experimento controlado para evaluar OOmFPWeb de acuerdo a su eficiencia, reproducibilidad, exactitud, facilidad de uso percibida e intención de uso.
   - Método de medición y versión: IFPUG FPA versión 4.1
   - Artefacto de software: Modelo navegacional OOWS.
   - Diseño del procedimiento: No presenta diseño.
   - Validación del procedimiento: Validación respecto a la eficiencia, reproducibilidad, exactitud, facilidad de uso percibida e intención de uso.
   - Herramienta: No presenta herramienta.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta el contexto de la investigación de manera detallada durante la introducción, y luego presenta un resumen OOmFPWeb y OOWS. El estudio cumple con los objetivos planteados, obteniendo como resultado del experimento que OOmFPWeb es eficiente comparado con las prácticas de la industria, provee medidas consistentes, es y se percibe como fácil de usar y útil por los usuarios.

5) Otras Notas

La validación es realizada al procedimiento OOmFPWeb (OO-Method Function Points for the Web).

Propuesta de Azzouz et al. (2004)

1) Información de la Búsqueda
    - Fecha de extracción de datos: 04/06/2008
    - Términos utilizados en la búsqueda: COSMIC Measurement
    - Base de datos donde se encontró: Google Scholar

2) Información General
    - Título: A Proposed Measurement Role in the Rational Unified Process and its Implementation with ISO 19761: COSMIC FFP.
    - Autores: Azzouz, S.; Abran, A.
    - Año publicación: 2004
    - Lugar de publicación: Software Measurement European Forum (SMEF 2004).

3) Información Específica
    - Resumen: Azzouz et al. [Azzouz04] presentan un procedimiento de medición basado en COSMIC FFP para medir el tamaño funcional de aplicaciones desarrolladas utilizando RUP [Kruchten00]. Estos autores utilizan artefactos de RUP para medir el tamaño funcional en las fases de modelado de negocios, de análisis de requisitos, y de análisis y diseño. Los artefactos utilizados por estos autores son el diagrama de casos de uso, los escenarios y los escenarios detallados. Estos autores no presentan un diseño sistemático del procedimiento de medición, ya que ellos sólo presentan la correspondencia entre los conceptos de COSMIC FFP y UML. Además, estos autores desarrollan una herramienta que permite medir de manera semi-automática el tamaño funcional de aplicaciones modeladas con UML.
    - Método de medición y versión: COSMIC FFP versión 2.2
    - Artefacto de software: Diagramas de casos de uso UML, escenarios UML, escenarios detallados UML.
    - Diseño del procedimiento: No presenta diseño.
    - Validación del procedimiento: No presenta validación.
    - Herramienta: Tiene una herramienta semi-automática.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta el contexto de la investigación durante la introducción. Este estudio no presenta trabajos relacionados y tiene pocas referencias. Sin embargo, presenta el método de medición COSMIC y la

metodología RUP de manera detallada. Finalmente, este estudio cumple con los objetivos planteados que consistían en el desarrollo de una herramienta que permita medir el tamaño funcional en RUP.

Propuesta de Cantone et al. (2004)

1) Información de la Búsqueda
    - Fecha de extracción de datos: 05/06/2008
    - Términos utilizados en la búsqueda: Function Point Measurement
    - Base de datos donde se encontró: IEEE Xplore

2) Información General
    - Título: Applying function point to unified modeling language: conversion model and pilot study.
    - Autores: Cantone, G.; Pace, D.; Calavaro, G.
    - Año publicación: 2004
    - Lugar de publicación: 10th International Symposium on Software Metrics (METRICS 2004).

3) Información Específica
    - Resumen: Cantone et al. [Cantone04] presentan una propuesta para medir el tamaño funcional de especificaciones UML utilizando el método de medición IFPUG FPA. Estos autores utilizan como artefacto de entrada el diagrama de casos de uso, el diagrama de clases y el diagrama de secuencia de UML. Para la medición del tamaño funcional, estos autores presentan las correspondencias de los conceptos de IFPUG FPA y los conceptos de los diagramas UML sin realizar un diseño sistemático del procedimiento. Además, estos autores presentan una herramienta denominada PAAC (Function Point Parametric Automatic Analyzer and Counter) y realizan un estudio empírico para comparar los resultados obtenidos por expertos con los resultados obtenidos por la herramienta que implementa el procedimiento propuesto.
    - Método de medición y versión: IFPUG FPA versión 4.1.1
    - Artefacto de software: Diagramas de casos de uso UML, diagramas de clase UML y diagramas de secuencia UML.
    - Diseño del procedimiento: No presenta diseño.
    - Validación del procedimiento: Validación de los resultados obtenidos por la herramienta con resultados obtenidos por expertos.
    - Herramienta: Tiene una herramienta automática.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta el contexto de la investigación durante la introducción, y luego presenta los trabajos relacionados. Este estudio cumple los objetivos planteados.

Propuesta de Condori-Fernández et al. (2004)

1) Información de la Búsqueda
   - Fecha de extracción de datos: 05/06/2008
   - Términos utilizados en la búsqueda: Function Point Measurement
   - Base de datos donde se encontró: IEEE Xplore

2) Información General
   - Título: Towards a Functional Size Measure for Object-Oriented Systems from Requirements Specifications.
   - Autores: Condori-Fernandez, N.; Abrahao, S.; Pastor, O.
   - Año publicación: 2004
   - Lugar de publicación: 4th International Conference on Quality Software (QSIC 2004).

3) Información Específica
   - Resumen: Condori-Fernández et al. [Condori04] presentan un procedimiento de medición basado en COSMIC FFP para medir el tamaño funcional de sistemas orientados a objeto desde sus especificaciones de requisitos. Estos autores utilizan artefactos del modelo de requisitos del método de desarrollo de software OO-Method [Pastor01]. Los artefactos utilizados por estos autores son el árbol de refinamiento de funciones, el diagrama de casos de uso, y el diagrama de secuencia. Estos autores no presentan un diseño riguroso, pero presentan reglas para identificar las correspondencias de conceptos para cada fase de COSMIC. Además, estos autores muestran la aplicación del procedimiento de medición a un caso de estudio.
   - Método de medición y versión: COSMIC FFP versión 2.2
   - Artefacto de software: Árbol de refinamiento de funciones OO-Method, diagramas de casos de uso OO-Method, diagrama de secuencia OO-Method.
   - Diseño del procedimiento: No presenta diseño.
   - Validación del procedimiento: No presenta validación.
   - Herramienta: No presenta herramienta.

4) Notas sobre la Calidad del Estudio
El estudio corresponde a un reporte técnico. El estudio presenta el contexto de la investigación durante la introducción, además presenta de manera resumida el método OO-Method, el modelo de requisitos de OO-Method, y el método de medición COSMIC. Este estudio presenta correctamente los trabajos relacionados y cumple con los objetivos planteados, que consistían en aplicar COSMIC FFP al modelo de requisitos de OO-Method.

5) Otras Notas
El procedimiento recibe el nombre RmFFP.

Propuesta de Bertolami et al. (2005)

1) Información de la Búsqueda
   - Fecha de extracción de datos: 06/06/2008
   - Términos utilizados en la búsqueda: Functional Size Measurement
   - Base de datos donde se encontró: Google Scholar

2) Información General
   - Título: Estimate of the Functional Size in the Requirements Elicitation.
   - Autores: Bertolami, M.; Oliveros, A.
   - Año publicación: 2005
   - Lugar de publicación: Journal in Computer Science & Technology, 2005.

3) Información Específica
   - Resumen: Bertolami et al. [Bertolami05] presentan un estudio empírico para validar la aplicación de un procedimiento de medición (especificado por los mismos autores) en la estimación del tamaño funcional en etapas tempranas del desarrollo de software.
   - Método de medición y versión: MK II FPA versión 1.3.1
   - Artefacto de software: Escenarios descritos en Léxico Extendido del Lenguaje.
   - Diseño del procedimiento: No presenta diseño.
   - Validación del procedimiento: Validación empírica del procedimiento.
   - Herramienta: No presenta herramienta.

4) Notas sobre la Calidad del Estudio
El estudio corresponde a un reporte técnico. El estudio presenta el contexto de la investigación durante la introducción, y en otra sección presenta los trabajos relacionados. Luego presenta un

resumen de Léxico Extendido del Lenguaje (LEL) y del método de medición MK II FPA. Este estudio cumple con los objetivos planteados, que consistían en aplicar MK II FPA para medir el tamaño funcional en etapas tempranas del desarrollo de software.

Propuesta de van den Berg et al. (2005)

1) Información de la Búsqueda
    - Fecha de extracción de datos: 06/06/2008
    - Términos utilizados en la búsqueda: Functional Size Measurement
    - Base de datos donde se encontró: Google Scholar

2) Información General
    - Título: Functional Size Measurement applied to UML-based user requirements.
    - Autores: van den Berg, K.G.; Dekkers, T.; Oudshoorn, R.
    - Año publicación: 2005
    - Lugar de publicación: 2nd Software Measurement European Forum (SMEF2005).

3) Información Específica
    - Resumen: van den Berg et al. [Berg05] presentan una propuesta para medir el tamaño funcional de modelos UML utilizando NESMA FPA o COSMIC FFP. Estos autores utilizan como artefactos los diagramas de casos de uso, la descripción de los casos de uso, el diagrama de actividades, el diagrama de interacción, el diagrama de transición de estados, el diagrama de secuencia, el diagrama de colaboración y el diagrama de clases. Estos autores no presentan un diseño sistemático de los procedimientos de medición. Sin embargo, esta propuesta ha sido utilizada en un caso de estudio definido en el estándar ISO/IEC 14143-4 [ISO02a] y en casos de estudio ampliamente utilizados por otros autores que proponen procedimientos de medición para modelos UML. En términos de la validación, estos procedimientos no presentan validación teórica o empírica. La aplicación de estos procedimientos es manual.
    - Método de medición y versión: NESMA FPA versión 2.0, COSMIC FFP versión 2.2
    - Artefacto de software: Diagramas de casos de uso, descripción de los casos de uso, diagrama de actividades, diagrama de interacción, diagrama de

transición de estados, diagrama de secuencia, diagrama de colaboración y el diagrama de clases.
- Diseño del procedimiento: No presenta diseño.
- Validación del procedimiento: No presenta validación.
- Herramienta: No presenta herramienta.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta el contexto de la investigación durante la introducción, y luego presenta el enfoque utilizado para especificar los procedimientos de medición, un resumen del método de medición NESMA FPA y un resumen del método de medición COSMIC FFP. Este estudio cumple con los objetivos planteados, demostrando la aplicabilidad de NESMA FPA y COSMIC a las especificaciones de requisistos UML.

Propuesta de Diab et al. (2005)

1) Información de la Búsqueda
   - Fecha de extracción de datos: 07/06/2008
   - Términos utilizados en la búsqueda: Function Point Measurement Tool
   - Base de datos donde se encontró: Google Scholar

2) Información General
   - Título: µcROSE: Automated Measurement of COS-MIC-FFP for Rational Rose Real Time.
   - Autores: Diab, H.; Koukane, F.; Frappier, M.; St-Denis, R.
   - Año publicación: 2005
   - Lugar de publicación: Journal of Information and Software Technology, Vol(47) Nro(3), 2005.

3) Información Específica
   - Resumen: Diab et al. [Diab05] presentan una herramienta que permite medir de manera automática el tamaño funcional de aplicaciones de tiempo real modeladas con Rational Rose Real Time (RRRT). Esta herramienta ha sido validada por un experto para evaluar su conformidad con COSMIC, y se ha utilizado para medir varios modelos RRRT.
   - Método de medición y versión: COSMIC FFP versión 2.2
   - Artefacto de software: Modelo RRRT (especificaciones ROOM).
   - Diseño del procedimiento: No presenta diseño.
   - Validación del procedimiento: Validación de conformidad

con COSMIC FFP versión 2.2 realizada por un experto, validación de la herramienta utilizando varios modelos RRRT, y validación de repetibilidad y exactitud de las medidas ya que este procedimiento está automatizado.
- Herramienta: Tiene una herramienta automática.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta los trabajos relacionados y un overview del método de medición COSMIC FFP y los modelos RRRT. Este estudio cumple con los objetivos de la propuesta, que consistían en desarrollar una herramienta que midiera automáticamente el tamaño funcional a partir de especificaciones RRRT.

Propuesta de Habela et al. (2005)

1) Información de la Búsqueda
   - Fecha de extracción de datos: 07/06/2008
   - Términos utilizados en la búsqueda: COSMIC Measurement
   - Base de datos donde se encontró: Google Scholar

2) Información General
   - Título: Adapting Use Case Model for COSMIC-FFP Based Measurement.
   - Autores: Habela, P.; Glowacki, E.; Serafinski, T.; Subieta, K.
   - Año publicación: 2005
   - Lugar de publicación: 15th International Workshop on Software Measurement (IWSM 2005).

3) Información Específica
   - Resumen: Habela et al. [Habela05] presentan una propuesta para medir el tamaño funcional de modelos de casos de uso UML utilizando COSMIC FFP. Estos autores no presentan un diseño sistemático del procedimiento de medición y tampoco presentan casos en los que se haya aplicado el procedimiento de medición. El procedimiento de medición se aplica manualmente sobre especificaciones de casos de uso detalladas.
   - Método de medición y versión: COSMIC FFP versión 2.2
   - Artefacto de software: Diagramas de casos de uso UML versión 1.5.
   - Diseño del procedimiento: No presenta diseño.
   - Validación del procedimiento: No presenta validación.

- Herramienta: No presenta herramienta.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta el contexto de la investigación durante la introducción, y luego presenta las necesidades, las características y el desarrollo de métodos de tamaño funcional. También presenta un resumen de los Use Case Function Points y el método de medición COSMIC. Este estudio cumple con los objetivos planteados.

Propuesta de Harput et al. (2005)

1) Información de la Búsqueda
   - Fecha de extracción de datos: 07/06/2008
   - Términos utilizados en la búsqueda: Function Point Measurement
   - Base de datos donde se encontró: Google Scholar

2) Información General
   - Título: Extending Function Point Analysis to Object-Oriented Requirements Specifications.
   - Autores: Harput, V.; Kaindl, H.; Kramer, S.
   - Año publicación: 2005
   - Lugar de publicación: 11th International Symposium on Software Metrics (METRICS 2005).

3) Información Específica
   - Resumen: Harput et al. [Harput05] presentan un procedimiento de medición para modelos conceptuales utilizados en la fase de requisitos del desarrollo de software. Este procedimiento utiliza como artefacto de entrada los escenarios, diagramas de secuencia y diagramas de clases de UML, y el modelo de información y los requisitos funcionales modelados con el método RETH (Requirements Engineering Through Hipertext). Estos autores no presentan un diseño sistemático, pero sí presentan las reglas de correspondencias entre los modelos conceptuales y los conceptos de IFPUG FPA. Además, estos autores presentan una herramienta semi-automática denominada RETH. Para validar la consistencia de los resultados obtenidos por esta herramienta con los resultados obtenidos aplicando directamente el método de medición IFPUG FPA se ha aplicado la herramienta a dos casos de estudio, uno de los cuales corresponde a un caso real.

- Método de medición y versión: IFPUG FPA versión 4.1.1
- Artefacto de software: Escenarios, diagramas de secuencia y diagramas de clases de UML.
- Diseño del procedimiento: No presenta diseño.
- Validación del procedimiento: Validación de la consistencia de los resultados.
- Herramienta: Tiene una herramienta semi-automática.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta el contexto de la investigación durante la introducción, y tiene una sección al final donde presenta los trabajos relacionados. Además, este estudio cumple los objetivos planteados, que consistían en el desarrollo de una herramienta semi-automática para medir el tamaño funcional de especificaciones.

Propuesta de Zivkovic et al. (2005)

1) Información de la Búsqueda
   - Fecha de extracción de datos: 08/06/2008
   - Términos utilizados en la búsqueda: Function Point Measurement Model
   - Base de datos donde se encontró: Google Scholar

2) Información General
   - Título: Automated Software Size Estimation based on Function Points using UML Models.
   - Autores: Zivkovic, A.; Rozman, I.; Hericko, M.
   - Año publicación: 2005
   - Lugar de publicación: Journal of Information & Software Technology, Vol(47), 2005.

3) Información Específica
   - Resumen: Zivkovic et al. [Zivkovic05] presentan una propuesta para medir el tamaño funcional de aplicaciones modeladas con UML utilizando IFPUG FPA. Este procedimiento consiste en la unificación y mejora de las propuestas de Fetcke et al., Uemura et al., Antoniol et al. y Ram et al. Los artefactos utilizados para la medición del tamaño funcional son los diagramas de actividades y diagramas de clases UML. Estos autores no presentan el diseño del procedimiento de medición, pero presentan un algoritmo para su aplicación. Además, estos autores han validado su procedimiento de medición mediante experimentos controlados, y han desarrollado una herramienta que permite realizar la

medición de manera automática.
- Método de medición y versión: IFPUG FPA versión 4.1
- Artefacto de software: Diagramas de actividad UML versión 1.4 y diagramas de clases UML versión 1.4.
- Diseño del procedimiento: No presenta diseño.
- Validación del procedimiento: Validación empírica del procedimiento.
- Herramienta: Tiene una herramienta automática.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta el contexto de la investigación durante la introducción, y en otra sección presenta los trabajos relacionados. Luego presenta un resumen de las propuestas de Fetcke et al., Uemura et al., Antoniol et al. y Ram et al. Este estudio unifica y mejora las propuestas mencionadas anteriormente, presentando suficiente evidencia original. Este estudio cumple con los objetivos planteados, que consistían en unificar propuestas que miden el tamaño funcional de modelos UML con IFPUG FPA.

Propuesta de Abrahao et al. (2006-1)

1) Información de la Búsqueda
- Fecha de extracción de datos: 08/06/2008
- Términos utilizados en la búsqueda: FPA Measurement
- Base de datos donde se encontró: Springer Link

2) Información General
- Título: A Functional Size Measurement Method for Object-Oriented Conceptual Schemas: Design and Evaluation Issues.
- Autores: Abrahao, S.; Poels, G.; Pastor, O.
- Año publicación: 2006
- Lugar de publicación: Journal of Software and System Modeling, Vol(5) Nro(1), 2006.

3) Información Específica
- Resumen: Abrahao et al. [Abrahao06a] presentan una propuesta para medir el tamaño funcional de aplicaciones MDA a partir de modelos conceptuales utilizando IFPUG FPA. Estos autores utilizan como artefacto de entrada el modelo conceptual del método de desarrollo de software OO-Method [Pastor01]. Este modelo conceptual está compuesto por un modelo de objetos, un modelo funcional, un modelo dinámico y un modelo de presentación. Estos autores presentan un

diseño sistemático del procedimiento, cumpliendo con todas las fases del modelo de procesos para métodos de medición de software propuesto por Jacquet y Abran [Jacquet97] [Abran99]. Además, presentan la aplicación del procedimiento de medición y la validación respecto a un modelo teórico para evaluar métodos de tamaño funcional. Finalmente, presentan una comparación entre los resultados obtenidos con la aplicación directa de IFPUG FPA y el procedimiento presentado.

- Método de medición y versión: IFPUG FPA versión 4.1
- Artefacto de software: Modelo de objetos OO-Method, modelo funcional OO-Method, modelo dinámico OO-Method y modelo de presentación OO-Method.
- Diseño del procedimiento: Presenta diseño siguiendo todos los pasos del modelo de procesos para los métodos de medición del software definido por Jacquet y Abran.
- Validación del procedimiento: Validación del procedimiento desde la teoría de la medición, validación teórica del procedimiento.
- Herramienta: No presenta herramienta.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta detalladamente los trabajos relacionados. Este estudio cumple con los objetivos de la propuesta que consistían en desarrollar un procedimiento de medición para aplicar IFPUG FPA a modelos conceptuales OO-Method.

5) Otras Notas

El procedimiento recibe el nombre OOmFP (OO-Method Function Points).

Propuesta de Bertolami et al. (2006)

1) Información de la Búsqueda
   - Fecha de extracción de datos: 09/06/2008
   - Términos utilizados en la búsqueda: Functional Size Measurement
   - Base de datos donde se encontró: Google Scholar

2) Información General
   - Título: SFP: Un Procedimento de Estimación de Puntos de Función de Escenarios.
   - Autores: Bertolami, M.; Oliveros, A.
   - Año publicación: 2006

- Lugar de publicación: 9th Workshop on Requirements Engineering (WER 2006).

3) Información Específica
   - Resumen: Bertolami et al. [Bertolami06] presentan una propuesta para medir el tamaño funcional a partir de escenarios, utilizando IFPUG FPA. Estos autores utilizan como artefacto de entrada los escenarios, y mediante la aplicación de una serie de reglas es posible obtener el tamaño funcional. Estos autores presentan un diseño sistemático del procedimiento de medición. Esta propuesta ha sido aplicada en ocho casos de estudio. Los resultados obtenidos en esos casos de estudio fueron obtenidos en el transcurso de 2 y 5 horas. Estos autores han concluido que el esfuerzo no es significativo, por lo tanto la aplicación del procedimiento no está automatizada.
   - Método de medición y versión: IFPUG FPA versión 4.1.1
   - Artefacto de software: Escenarios LEL.
   - Diseño del procedimiento: Presenta diseño según el proceso de aplicación de métodos de medición de software definido en el estándar ISO/IEC 14143-1.
   - Validación del procedimiento: No presenta validación.
   - Herramienta: No presenta herramienta.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta el contexto de la investigación durante la introducción, y en otra sección presenta los trabajos relacionados. Luego presenta un resumen del método de medición IFPUG FPA y el estándar ISO/IEC 14141-1. Este estudio cumple con los objetivos planteados, que consistían en aplicar IFPUG FPA para medir el tamaño funcional en etapas tempranas del desarrollo de software.

5) Otras Notas

El procedimiento de medición recibe el nombre de SFP.

Propuesta de Condori-Fernández et al. (2006-1)

1) Información de la Búsqueda
   - Fecha de extracción de datos: 09/06/2008
   - Términos utilizados en la búsqueda: Function Point Measurement
   - Base de datos donde se encontró: Google Scholar

2) Información General

- Título: Verifying the Construction of a Software Model from a Requirements Model.
- Autores: Condori-Fernández, N.; Pastor, O.
- Año publicación: 2006
- Lugar de publicación: 9th Workshop on Requirements Engineering (WER 2006).

3) Información Específica
   - Resumen: Condori-Fernández et al. [Condori06] presentan el proceso de aplicación del procedimiento de medición RmFFP (especificado por los mismos autores). En este trabajo los autores verifican la construcción del modelo de software cuando se aplica RmFFP. Además, estos autores presentan guías para evalúar la reproducibilidad del procedimiento RmFFP.
   - Método de medición y versión: COSMIC FFP versión 2.2
   - Artefacto de software: Árbol de refinamiento de funciones OO-Method, diagramas de casos de uso OO-Method, diagrama de secuencia OO-Method.
   - Diseño del procedimiento: No presenta diseño.
   - Validación del procedimiento: Validación de la construcción del modelo de software.
   - Herramienta: No presenta herramienta.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta el contexto de la investigación durante la introducción, y presenta de manera resumida los trabajos relacionados. Este estudio cumple con los objetivos planteados que consistían en verificar la construcción del modelo de software cuando se aplica RmFFP.

5) Otras Notas

La verificación se realiza al procedimiento RmFFP.

Propuesta de Condori-Fernández et al. (2006-2)

1) Información de la Búsqueda
   - Fecha de extracción de datos: 09/06/2008
   - Términos utilizados en la búsqueda: Function Point Measurement
   - Base de datos donde se encontró: Springer Link

2) Información General
   - Título: Evaluating the Productivity and Reproducibility of a Measurement Procedure.
   - Autores: Condori-Fernández, N.; Pastor, O.

- Año publicación: 2006
- Lugar de publicación: Proceedings of the ER Workshops (ER Wrokshops 2006).

3) Información Específica
   - Resumen: Condori-Fernández et al. [Condori06a] presentan un experimento controlado para evaluar la productividad y la reproducibilidad del procedimiento de medición RmFFP (especificado por los mismos autores). Con los resultados del experimento, estos autores concluyen que la productividad es aceptable comparada con otros procedimientos y que las medidas obtenidas por RmFFP son reproducibles.
   - Método de medición y versión: COSMIC FFP versión 2.2
   - Artefacto de software: Árbol de refinamiento de funciones OO-Method, diagramas de casos de uso OO-Method, diagrama de secuencia OO-Method.
   - Diseño del procedimiento: No presenta diseño.
   - Validación del procedimiento: Validación de la productividad y reproducibilidad de RmFFP.
   - Herramienta: No presenta herramienta.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta el contexto de la investigación durante la introducción, y luego presenta el procedimiento de medición RmFFP. Este estudio cumple con los objetivos planteados, evaluando la aplicación de RmFFP en términos de su productividad y reproducibilidad. Finalmente, en un anexo muestra los datos del experimento.

5) Otras Notas

La validación se realiza al procedimiento RmFFP.

Propuesta de Condori-Fernández et al. (2006-3)

1) Información de la Búsqueda
   - Fecha de extracción de datos: 10/06/2008
   - Términos utilizados en la búsqueda: COSMIC Measurement Procedure
   - Base de datos donde se encontró: Google Scholar

2) Información General
   - Título: Re-Assessing the Intention to Use a Measurement Procedure based on COSMIC-FFP.
   - Autores: Condori-Fernández, N.; Pastor, O.
   - Año publicación: 2006

- Lugar de publicación: 16th International Workshop on Software Measurement (IWSM 2006).

3) Información Específica
   - Resumen: Condori-Fernández et al. [Condori06c] presentan la replicación de un experimento realizado para evaluar la adopción en práctica del procedimiento de medición RmFFP. Los resultados del experimento muestran que existe una intención de uso de RmFFP, la cual está más influenciada por la utilidad que por la facilidad de uso del procedimiento.
   - Método de medición y versión: COSMIC FFP versión 2.2
   - Artefacto de software: Árbol de refinamiento de funciones OO-Method, diagramas de casos de uso OO-Method, diagrama de secuencia OO-Method.
   - Diseño del procedimiento: No presenta diseño.
   - Validación del procedimiento: Validación empírica de la aceptación en la práctica del procedimiento RmFFP.
   - Herramienta: No presenta herramienta.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta el contexto de la investigación durante la introducción, y luego presenta una descripción general del proceso de evaluación de la intención de uso del procedimiento de medición RmFFP. Este estudio cumple con los objetivos planteados, replicando la evaluación de la intención de uso de RmFFP.

5) Otras Notas

La validación se realiza al procedimiento RmFFP.

Propuesta de Fraternali et al. (2006)

1) Información de la Búsqueda
   - Fecha de extracción de datos: 10/06/2008
   - Términos utilizados en la búsqueda: FPA Measurement
   - Base de datos donde se encontró: ACM Digital Library

2) Información General
   - Título: Automating Function Point Analysis with Model Driven Development.
   - Autores: Fraternali, P.; Tisi, M.; Bongio, A.
   - Año publicación: 2006
   - Lugar de publicación: Proceedings of the 2006 conference of the Center for Advanced Studies on Collaborative research (CASCON 2006).

3) Información Específica
   - Resumen: Fraternali et al. [Fraternali06] presentan un procedimiento de medición para modelos conceptuales de aplicaciones web realizados con la herramienta WebML (esquema de datos y esquema de hipertexto). Estos autores aplican el método de medición IFPUG FPA. Estos autores no presentan un diseño sistemático del procedimiento de medición, pero sí presentan las correspondencias entre los conceptos del modelo WebML e IFPUG FPA. Además, estos autores desarrollan una herramienta que es incorporada a la herramienta WebRatio que implementa los modelos WebML. Los resultados obtenidos por la herramienta han sido verificados por expertos.
   - Método de medición y versión: IFPUG FPA versión 4.1.1
   - Artefacto de software: Esquema de datos WebML y esquema de hipertexto WebML.
   - Diseño del procedimiento: No presenta diseño.
   - Validación del procedimiento: Validación de los resultados obtenidos por la herramienta con los resultados obtenidos por expertos.
   - Herramienta: Tiene una herramienta automática.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta el contexto de la investigación durante la introducción, y luego presenta el método de medición IFPUG FPA y los modelos WebML con la herramienta que los implementa, denominada WebRatio. También, este estudio presenta una sección de trabajos relacionados. Este estudio cumple los objetivos planteados, que consistían en medir el tamaño funcional en modelos conceptuales de aplicaciones desarrolladas mediante un desarrollo dirigido por modelos.

Propuesta de Hericko et al. (2006)

1) Información de la Búsqueda
   - Fecha de extracción de datos: 11/06/2008
   - Términos utilizados en la búsqueda: Function Point Measurement
   - Base de datos donde se encontró: Science Direct

2) Información General
   - Título: A Formal Representation on Functional Size Measurement Methods.

- Autores: Hericko, M.; Rozman, I.; Zivkovic, A.
- Año publicación: 2006
- Lugar de publicación: Journal of Systems and Software, Vol(79) Nro(9), 2006.

3) Información Específica
   - Resumen: Hericko et al. [Hericko06] presentan un procedimiento de medición para modelos conceptuales utilizando un formulario universal denominado GASS. De esta manera, los modelos conceptuales (orientados a aspectos, orientados a objetos o estructurados) son transformados en un formulario GASS, desde el cual se puede obtener el tamaño funcional según IFPUG FPA, MK II FPA, o COSMIC. Estos autores no presentan un diseño sistemático del procedimiento de medición, pero sí presentan las correspondencias entre los conceptos del modelo GASS y los métodos de medición. Además presentan la transformación de modelos orientados a objeto a GASS utilizando los modelos de casos de uso, de clases, de secuencia, de actividades y de estados UML. Finalmente presentan la aplicación del procedimiento para obtener las medidas con IFPUG FPA, MK II FPA y COSMIC.
   - Método de medición y versión: IFPUG FPA versión 4.2
   - Artefacto de software: Modelos de casos de uso, de clases, de secuencia, de actividades y de estados UML.
   - Diseño del procedimiento: No presenta diseño.
   - Validación del procedimiento: No presenta validación.
   - Herramienta: No presenta herramienta.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta el contexto de la investigación durante la introducción, y luego presenta brevemente los trabajos relacionados. Además, este estudio presenta el modelo GASS. Finalmente, este estudio cumple con los objetivos planteados.

Propuesta de Abrahao et al. (2007-1)

1) Información de la Búsqueda
   - Fecha de extracción de datos: 11/06/2008
   - Términos utilizados en la búsqueda: Function Point Measurement
   - Base de datos donde se encontró: Science Direct

2) Información General

- Título: Experimental Evaluation of an Object-Oriented Function Point Measurement Procedure.
- Autores: Abrahao, S.; Poels, G.
- Año publicación: 2007
- Lugar de publicación: Journal of Information and Software Technology, Vol(49) Nro(4), 2007.

3) Información Específica
   - Resumen: Abrahao et al. [Abrahao07] presentan un experimento controlado para comparar IFPUG FPA y OOmFP de acuerdo a su eficiencia, reproducibilidad, exactitud, facilidad de uso percibida e intención de uso.
   - Método de medición y versión: IFPUG FPA versión 4.1
   - Artefacto de software: Modelo de objetos OO-Method, modelo funcional OO-Method, modelo dinámico OO-Method y modelo de presentación OO-Method.
   - Diseño del procedimiento: No presenta diseño.
   - Validación del procedimiento: Validación empírica con respecto a la eficiencia, reproducibilidad, exactitud, facilidad de uso percibida e intención de uso.
   - Herramienta: No presenta herramienta.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta el contexto de la investigación durante la introducción, y luego presenta un resumen de OO-Mehod y OOmFP, y de los métodos utilizados para evaluar métodos FSM. El estudio cumple con los objetivos planteados.

5) Otras Notas

La validación se realiza al procedimiento OOmFP (OO-Method Function Points).

Propuesta de Abrahao et al. (2007-2)

1) Información de la Búsqueda
   - Fecha de extracción de datos: 12/06/2008
   - Términos utilizados en la búsqueda: Function Point Measurement
   - Base de datos donde se encontró: Springer Link

2) Información General
   - Título: A Model-Driven Measurement Procedure for Sizing Web Applications: Design, Automation and Validation.
   - Autores: Abrahao, S.; Mendes, E.; Gomez, J.; Insfrán,

E.
- Año publicación: 2007
- Lugar de publicación: ACM/IEEE 10th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2007).

3) Información Específica
- Resumen: Abrahao et al. [Abrahao07a] presentan una propuesta para medir el tamaño funcional de aplicaciones web generadas en entornos MDA utilizando el método IFPUG FPA. Estos autores utilizan como artefacto de entrada el diagrama de clases UML y el diagrama de accesos navegacionales de OO-H [Schwabe95], y mediante la aplicación de una serie de reglas es posible obtener el tamaño funcional. Estos autores presentan el diseño del procedimiento desarrollando todas las fases del modelo de procesos para métodos de medición de software propuesto por Jacquet y Abran [Jacquet97] [Abran99]. Además, estos autores presentan la aplicación del procedimiento de medición a un caso de estudio, y su automatización. Además, esta propuesta ha sido validada en términos de la exactitud de sus medidas.
- Método de medición y versión: IFPUG FPA versión 4.1
- Artefacto de software: Diagrama de clases UML 2.1 y el diagrama de accesos navegacionales de OO-H.
- Diseño del procedimiento: Presenta diseño siguiendo todos los pasos del modelo de procesos para los métodos de medición del software definido por Jacquet y Abran.
- Validación del procedimiento: Validación de la exactitud de las medidas.
- Herramienta: Tiene una herramienta automática.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta el contexto de la investigación durante la introducción, y luego presenta el proceso de desarrollo OO-H. Este estudio cumple los objetivos planteados, que consistían en desarrollar, validar y automatizar un procedimiento de medición para aplicaciones web.

5) Otras Notas

El procedimiento recibe el nombre de OO-HFP (Object-Oriented Hypermedia Function Points).

Propuesta de Giachetti et al. (2007)

1) Información de la Búsqueda
    - Fecha de extracción de datos: 13/06/2008
    - Términos utilizados en la búsqueda: Function Point Measurement
    - Base de datos donde se encontró: IEEE Xplore

2) Información General
    - Título: Updating OO-Method Function Points.
    - Autores: Giachetti, G.; Marín, B.; Condori-Fernández, N.; Molina, J.C.
    - Año publicación: 2007
    - Lugar de publicación: 6th IEEE International Conference on the Quality of Information and Communications Technology (QUATIC 2007).

3) Información Específica
    - Resumen: Giachetti et al. [Giachetti07] presentan la actualización del procedimiento OOmFP según las nuevas primitivas conceptuales de OO-Method. Estos autores no presentan el diseño del procedimiento, pero sí presentan las correspondencias entre los conceptos de IFPUG FPA y las primitivas del modelo conceptual OO-Method. Además estos autores presentan una herramienta que automatiza OOmFP y su utilización en nueve casos reales.
    - Método de medición y versión: IFPUG FPA versión 4.2
    - Artefacto de software: Modelo de objetos OO-Method, modelo funcional OO-Method, modelo dinámico OO-Method y modelo de presentación OO-Method.
    - Diseño del procedimiento: No presenta diseño.
    - Validación del procedimiento: No presenta validación.
    - Herramienta: Tiene una herramienta automática.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta el contexto de la investigación durante la introducción, y luego presenta un resumen de IFPUG FPA, OO-Mehod y OOmFP. El estudio cumple con los objetivos planteados, que correspondían a actualizar y automatizar OOmFP.

5) Otras Notas

La validación se realiza al procedimiento OOmFP (OO-Method Function Points).

Propuesta de Grau et al. (2007)

1) Información de la Búsqueda
    - Fecha de extracción de datos: 14/06/2008
    - Términos utilizados en la búsqueda: Function Point Measurement
    - Base de datos donde se encontró: Springer Link

2) Información General
    - Título: Using the PRiM method to Evaluate Requirements Model with COSMIC-FFP.
    - Autores: Grau, G.; Franch, X.
    - Año publicación: 2007
    - Lugar de publicación: International Conference on Software Process and Product Measurement (IWSM-MENSURA 2007).

3) Información Específica
    - Resumen: Grau et al. [Grau07a] presentan una propuesta para medir el tamaño funcional a partir de modelos i* utilizando el método COSMIC FFP. Los modelos i* son generados mediante reingeniería de las aplicaciones, utilizando el método PRIM [Grau07b]. Estos autores utilizan como artefacto de entrada el modelo i*, y mediante la aplicación de una serie de reglas es posible obtener el tamaño funcional. Estos autores presentan el diseño del procedimiento enfocándose en las dos últimas fases del modelo de procesos para métodos de medición de software propuesto por Jacquet y Abran [Jacquet97] [Abran99]: la correspondencia de conceptos entre COSMIC FFP y el modelo i*, y la definición de reglas de medición. Además, estos autores presentan la aplicación sistemática del procedimiento de medición a un caso de estudio, y una herramienta que permite aplicar automáticamente el procedimiento a modelos i*.
    - Método de medición y versión: COSMIC FFP versión 2.2
    - Artefacto de software: Modelos i*.
    - Diseño del procedimiento: Presenta diseño según las dos últimas etapas del modelo de procesos para los métodos de medición del software propuesto por Jacquet y Abran.
    - Validación del procedimiento: No presenta validación.
    - Herramienta: Tiene una herramienta automática.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta el contexto de la investigación durante la introducción y luego presenta el framework i*. Este estudio cumple con los objetivos planteados, que consistían en aplicar el método de medición de tamaño funcional COSMIC en PRIM.

Propuesta de Levesque et al. (2008)

1) Información de la Búsqueda
    - Fecha de extracción de datos: 15/06/2008
    - Términos utilizados en la búsqueda: Function Point Measurement
    - Base de datos donde se encontró: ACM Digital Library

2) Información General
    - Título: Estimating software size with UML models.
    - Autores: Levesque, G.; Bevo, V.; Tran Cao, D.
    - Año publicación: 2008
    - Lugar de publicación: Proceedings of the C^3S^2E Conference (C3S2E 2008).

3) Información Específica
    - Resumen: Levesque et al. [Levesque08] presentan una propuesta para medir el tamaño funcional de especificaciones UML. Estos autores utilizan como artefacto de entrada el diagrama de secuencia de UML. Para la medición del tamaño funcional, estos autores presentan las correspondencias de los conceptos de COSMIC y los conceptos del diagrama de secuencia de UML 1.4 y 2.0 para identificar los movimientos y las manipulaciones de datos. Sin embargo, no presentan el diseño del procedimiento de medición. Además, estos autores presentan la aplicación del procedimiento al caso de estudio Rice Cooker [Abran00] y obtienen valores similares. En cuanto a la automatización, estos autores reconocen la importancia de automatizar el procedimiento de medición, pero no presentan el desarrollo de la herramienta que realice esta tarea.
    - Método de medición y versión: COSMIC FFP versión 2.2
    - Artefacto de software: Diagramas de secuencia UML 1.4 y 2.0.
    - Diseño del procedimiento: No presenta diseño.
    - Validación del procedimiento: No presenta validación.
    - Herramienta: No presenta herramienta.

4) Notas sobre la Calidad del Estudio

El estudio corresponde a un reporte técnico. El estudio presenta el contexto de la investigación durante la introducción, y luego presenta la motivación de automatizar la aplicación del procedimiento de medición de tamaño funcional. Posteriormente presentan un resumen del método COSMIC. Este estudio no cumple los objetivos planteados, que consistían en automatizar la medición del tamaño funcional, ya que no presentan ninguna herramienta. Sin embargo, presentan intentos de automatizar el procedimiento de medición y una sección en que discuten los resultados obtenidos.

Síntesis de Datos

Con la información extraída de los estudios seleccionados se ha realizado la Tabla 5. Los estudios han sido agrupados por procedimiento de medición, pues varias propuestas presentan las reglas de correspondencias en un estudio, la validación en otro estudio y la herramienta que automatiza el procedimiento en otro estudio. De esta manera, las propuestas de procedimientos de medición que se muestran en la Tabla 5 están ordenadas por el primer autor y el año de la publicación del primer estudio donde se menciona el procedimiento de medición.

En la Tabla 5, por cada propuesta se especifica el método de medición en el que se basa el procedimiento propuesto, el artefacto de software utilizado para realizar la medición con el procedimiento propuesto, y se detalla si el procedimiento de medición presenta el diseño, la validación, o la automatización del mismo. El proceso de síntesis de datos se realizó desde el 16/06/2008 al 20/06/2008, ambas fechas incluidas.

Tabla 5. Procedimientos de medición de tamaño funcional para modelos conceptuales.

Propuesta	Método Medición	Artefacto de Software	Diseño	Validación	Herramienta
Fetcke 1997	IFPUG FPA 4.0	Modelo de casos de uso, modelo de objetos, modelo de análisis	No	No	No
Gramantieri 1997	IFPUG FPA 4.0	Diagrama ER-DFD	No	No	Herramienta automática
Shoval 1997	MK II FPA 4.0	Diagrama de flujo de datos	No	No	Herramienta semi-automática
Caldiera 1998	IFPUG FPA	Modelo de objetos	No	Validación empírica	Herramienta automática

Tabla 5. (Continuación) Procedimientos de medición de tamaño funcional para modelos conceptuales.

Propuesta	Método Medición	Artefacto de Software	Diseño	Validación	Herramienta
Bévo 1999	COSMIC FFP 2.0	Diagrama de casos de uso	No	No	No
Uemura 1999	IFPUG FPA 4.0	Diagrama de casos de uso y diagrama de clases UML 1.1	No	Validación de la medición por expertos	Herramienta automática
Kusumoto 2000	IFPUG FPA 4.0	Modelo de requisitos de REQUARIO	No	Validación de la medición por expertos	Herramienta automática
Ram 2000	IFPUG FPA 4.1	Modelo de clases	No	No	No
Diab 2001	COSMIC FFP 2.2	Modelo de clases RRRT (especificaciones ROOM)	No	Validación de conformidad con COSMIC FFP versión 2.2 realizada por un experto, y validación de repetibilidad y exactitud de las medidas	Herramienta automática
Abrahao 2002	IFPUG FPA 4.1.1	Modelo navegacional OOWS	No	Validación respecto a la eficiencia, reproducibilidad, exactitud, facilidad de uso percibida e intención de uso	No
Tavares 2002	IFPUG FPA	Diagrama de casos de uso y diagrama de clases UML	No	No	Herramienta automática
Bertolami 2003	MK II FPA 1.3.1	Escenarios descritos en Léxico Extendido del Lenguaje	No	Validación empírica	No
Pagano 2003	COSMIC FFP	Diagrama de clases, diagrama de estado y diagrama de colaboración xUML	No	No	No
Poels 2003	COSMIC FFP	Modelo de negocio (diagrama de clases, tablas evento-objeto, y diagramas de transición de estados) y modelo de servicios MERODE	No	Validación teórica	No

Tabla 5. (Continuación) Procedimientos de medición de tamaño funcional para modelos conceptuales.

Propuesta	Método Medición	Artefacto de Software	Diseño	Validación	Herramienta
Abrahao 2004	IFPUG FPA 4.1.1	Modelo de objetos, modelo funcional, modelo dinámico y modelo de presentación OO-Method	Diseño según el modelo de procesos para los métodos de medición del software	Validación desde la teoría de la medición, validación teórica, validación empírica de eficiencia, reproducibilidad, exactitud, facilidad de uso percibida e intención de uso	Herramienta automática
Azzouz 2004	COSMIC FFP 2.2	Diagramas de casos de uso, escenarios, escenarios detallados	No	No	Herramienta semi-automática
Cantone 2004	IFPUG FPA 4.1.1	Diagramas de casos de uso, diagramas de clase y diagramas de secuencia.	No	Validación de la medición por expertos	Herramienta automática
Condori-Fernández 2004	COSMIC FFP 2.2	Árbol de refinamiento de funciones, diagramas de casos de uso, diagrama de secuencia OO-Method	No	Validación de la construcción del modelo de software, validación empírica de productividad, reproducibilidad, y aceptación en la práctica	No
Berg 2005	NESMA FPA 2.0, COSMIC FFP 2.2	Diagramas de casos de uso, descripción de los casos de uso, diagrama de actividades, diagrama de interacción, diagrama de transición de estados, diagrama de secuencia, diagrama de colaboración y el diagrama de clases	No	No	No
Habela 2005	COSMIC FFP 2.2	Diagramas de casos de uso	No	No	No

Tabla 5. (Continuación) Procedimientos de medición de tamaño funcional para modelos conceptuales.

Propuesta	Método Medición	Artefacto de Software	Diseño	Validación	Herramienta
Harput 2005	IFPUG FPA 4.1.1	Escenarios, diagramas de secuencia y diagramas de clases de UML	No	Validación de la consistencia de los resultados	Herramienta semi-automática
Zivkovic 2005	IFPUG FPA 4.1	Diagramas de actividad y diagramas de clases UML	No	Validación empírica	Herramienta automática
Bertolami 2006	IFPUG FPA 4.1.1	Escenarios	Diseño según el proceso de aplicación de métodos de medición de software definido en el estándar ISO/IEC 14143-1	No	No
Fraternali 2006	IFPUG FPA 4.1.1	Esquema de datos WebML y esquema de hipertexto WebML	No	Validación de los resultados obtenidos por la herramienta con los resultados obtenidos por expertos	Herramienta automática
Erico 2006	IFPUG FPA versión 4.2	Modelos de casos de uso, de clases, de secuencia, de actividades y de estados UML	No	No	No
Abrahao 2007	IFPUG FPA 4.1	Diagrama de clases UML y el diagrama de accesos navegacionales de OO-H	Diseño según el modelo de procesos para los métodos de medición del software	Validación de la exactitud de las medidas	Herramienta automática
Grau 2007	COSMIC FFP 2.2	Modelos i*	Diseño según las dos últimas etapas del modelo de procesos para los métodos de medición del software	No	Herramienta automática
Levesque 2008	COSMIC FFP 2.2	Diagramas de secuencia UML 1.4 y 2.0	No	No	No

De las 28 propuestas de procedimientos de medición encontrados en la revisión sistemática, sólo una utiliza NESMA FPA y dos utilizan MK II FPA. Esto tiene sentido, ya que estos métodos de medición son muy similares a IFPUG FPA, y en la literatura es posible encontrar más trabajos que aplican IFPUG FPA que pueden clarificar el desarrollo de un procedimiento de medición. De las propuestas restantes, 16 están basadas en IFPUG FPA y 9 están basadas en COSMIC. Además, la propuesta que utiliza NESMA FPA también utiliza COSMIC.

Del total de propuestas, 4 presentan nociones del diseño del procedimiento de medición, pero sólo 2 de ellas presentan un diseño riguroso. El diseño de un procedimiento de medición es una fase clave en el desarrollo del procedimiento, ya que en esta fase se especifica el objetivo de la medición, el artefacto que será medido, el concepto que será medido, las reglas de medición, y la estrategia que será utilizada para llevar a cabo la medición. Por esta razón, es muy importante realizar un diseño correcto de un procedimiento de medición (abstrayendo de manera correcta los elementos a ser medidos), ya que en caso contrario, el procedimiento podría no estar midiendo lo que debe medir según lo especificado en el método de medición estándar seleccionado como base. Además, es importante tener en cuenta la influencia directa que tiene el diseño de un procedimiento de medición sobre la aplicación de dicho procedimiento, ya que si el diseño es incorrecto, la aplicación del procedimiento puede ser confusa y finalmente se pueden obtener medidas erróneas. A pesar de esto, existen 12 propuestas de procedimientos de medición que presentan validación del procedimiento sin antes realizar el diseño, e incluso existen 4 propuestas que automatizan la aplicación del procedimiento de medición sin tener un diseño sistemático y sin haber validado el procedimiento. Esto pone de manifiesto que la aplicación, la validación y la automatización del procedimiento pueden requerir demasiado esfuerzo y pueden tender a ser erróneas debido que no se tiene un diseño correctamente definido del procedimiento.

Finalmente, sólo existen 2 propuestas de procedimientos de medición que han sido correctamente diseñados, validados y posteriormente automatizados. Estas dos propuestas son del autor Abrahao en 2004 y 2007, y permiten medir el tamaño funcional de aplicaciones tradicionales y web, respectivamente.

2.2.3 Reportando la revisión

En esta fase se documenta la revisión. Esta fase se ha realizado desde el 23/06/2008 al 27/06/2008, ambos días incluidos. La documentación de la revisión se ha presentado en esta tesis.

2.3 Conclusiones

En este capítulo se han presentado los métodos de medición estándares con sus respectivas versiones y características para calcular el tamaño funcional de aplicaciones de software.

De los métodos de medición estándares, tanto IFPUG FPA, NESMA FPA y MK II FPA obtienen el tamaño funcional de las aplicaciones según las funcionalidades que el usuario de la aplicación observa. Por este motivo, estos métodos presentan limitaciones a la hora de medir aplicaciones generadas en varias capas, ya que no consideran la funcionalidad que tienen estas aplicaciones para comunicar las piezas de software en las diferentes capas. Además, dado que estos métodos encapsulan las aplicaciones como un todo, con estos métodos es imposible calcular el tamaño funcional de cada pieza de software que compone la aplicación. Dado que el método de medición COSMIC permite medir el tamaño funcional considerando todos los usuarios que necesitan funcionalidad de cualquier pieza de software de la aplicación, este método de medición es más adecuado para obtener el tamaño funcional de aplicaciones compuestas por varias piezas de software (como es el caso de la mayoría de aplicaciones desarrollas en la industria para Sistemas de Información de Gestión).

En este capítulo también se ha presentado una revisión sistemática de procedimientos de medición que permiten medir el tamaño funcional de aplicaciones de software a partir de modelos conceptuales. Al realizar la revisión sistemática, se han encontrado 28 propuestas de procedimientos de medición. Se debe tener en cuenta que la revisión sistemática es un proceso riguroso para revisar literatura, evitando los sesgos que pueda tener la persona que realiza la revisión hacia determinados procedimientos de medición. Por este motivo, a pesar de que el autor de esta tesis tenga conocimiento de características del diseño, de la validación y de herramientas que automatizan algunos de los procedimientos de medición encontrados en la revisión sistemática, estas características sólo fueron reportadas cuando se encontró evidencia durante la revisión sistemática.

De los 28 procedimientos de medición encontrados en la revisión sistemática, sólo dos propuestas presentan un diseño sistemático, validación y automatización del procedimiento. Estas dos propuestas son presentadas por el autor Abrahao en 2004 y en 2007. Sin embargo, estas propuestas están basadas en IFPUG FPA, y por esta razón tienen las limitaciones mencionadas anteriormente para medir aplicaciones compuestas por varias piezas de software.

De las propuestas basadas en COSMIC, se pueden destacar la propuesta de Diab en 2001 que presenta validación y automatización, la propuesta de Poels en 2003 que presenta validación, la propuesta de Condori-Fernández el 2004 que presenta validación, y la propuesta de Grau en 2007 que presenta diseño y automatización.

Las propuestas de Condori-Fernández y Grau utilizan modelos conceptuales al nivel de requisitos (que no permiten especificar toda la funcionalidad que debe tener la aplicación final), por lo que sólo obtienen medidas estimadas del tamaño funcional de la aplicación final. Tanto la propuesta de Diab como la propuesta de Poels utilizan modelos conceptuales que posteriormente son transformados en la aplicación final en entornos MDD. La propuesta de Diab utiliza modelos conceptuales para aplicaciones de tiempo real, y sólo la propuesta de Poels utiliza modelos conceptuales que permiten generar sistemas de información de gestión. Sin embargo, este procedimiento no tiene un diseño claro que permita aplicar esta propuesta a otras metodologías MDD, ni tampoco para desarrollar una herramienta que automatice la medición. Además, los modelos conceptuales del método MERODE no tienen suficiente expresividad semántica para especificar toda la aplicación en el modelo conceptual (por ejemplo no permite especificar la interfaz gráfica de las aplicaciones). Esto pone de manifiesto la necesidad de contar con un procedimiento de medición correctamente diseñado para medir el tamaño funcional de aplicaciones compuestas por varias piezas de software, que son generadas en entornos MDD a partir de sus modelos conceptuales.

Capítulo 3
Fundamentos

En este capítulo se presentan los trabajos en los que se ha basado el desarrollo de esta tesis. Dado que el objetivo de esta tesis es presentar un procedimiento de medición basado en COSMIC para aplicaciones generadas automáticamente a partir de modelos conceptuales OO-Method, en este capítulo se presenta una breve descripción de ambos métodos.

Además, para el desarrollo sistemático del procedimiento de medición que propone esta tesis se ha utilizado un modelo de procesos para la medición del software que también es presentado en este capítulo.

3.1 El Método de Medición de Tamaño Funcional COSMIC (Version 3.0)

El método de medición de tamaño funcional COSMIC [Abran07] puede ser usado para medir cualquier tipo de software, como por ejemplo software que se encuentra en el ámbito de los negocios, software de tiempo real, software de varias capas, etc. La aplicación de este método incluye tres fases: la fase de estrategia de medición, la fase de correspondencia de conceptos y la fase de medición de los conceptos identificados.

En la *fase de estrategia de medición* se debe definir el propósito de la tarea de medición, es decir, se debe definir por qué es necesario medir y para qué será utilizado el resultado de la medición. Luego, se debe definir el alcance de la medición para poder seleccionar el conjunto de requisitos funcionales que serán considerados en esa tarea de medición. En la definición del alcance de la medición también se deben definir las piezas de software que serán consideradas en la tarea de medición. Posteriormente, se debe identificar los usuarios funcionales de la aplicación. Los usuarios funcionales son tipos de usuarios que envían (o reciben) datos a (desde) los procesos funcionales de una pieza de software. En esta fase también se identifican las fronteras, que son interfaces conceptuales entre el usuario funcional y la pieza de software que será medida. Finalmente, se determina el nivel de granularidad que tendrá la pieza de software que será medida.

En la *fase de correspondencia de conceptos* se deben identificar los procesos funcionales, es decir, se deben identificar los componentes elementales asociados a un conjunto de requisitos de usuarios. Cada proceso funcional comienza con un evento disparado por un usuario funcional a través de un movimiento de datos y termina cuando se han ejecutado todos los movimientos de datos que requiere el evento para su correcta ejecución. Es necesario tener en cuenta que un evento disparado es un evento que causa el usuario funcional de la pieza de software para iniciar uno o más procesos funcionales. Posterior a la identificación de los procesos funcionales, se deben identificar los grupos de datos, que son conjuntos de atributos distintos, no vacíos, no ordenados, no redundantes, y que participan en un proceso funcional. Finalmente se realiza de manera opcional la identificación de los atributos de los grupos de datos, que son la parte más pequeña de información que componen a un grupo de datos.

En la *fase de medición* se deben identificar los movimientos de datos (Entrada -E-, Salida -X-, Lectura -R-, y Escritura -W-) para cada proceso funcional. Los movimientos de datos corresponden a movimientos de grupos de datos entre usuarios funcionales y procesos funcionales (y viceversa), o entre procesos funcionales y la base de datos (y viceversa). Cuando todos los movimientos de datos han sido identificados, se aplica la función de medición para el proceso funcional. Esta función es una función matemática que asigna 1 CFP (Cosmic Function Point) a cada movimiento de datos que ocurre dentro de un proceso funcional. De esta manera, cuando todos los procesos funcionales han sido medidos, los resultados de las mediciones son agregados para obtener el tamaño funcional de la pieza de software que está siendo medida.

La Figura 1 esquematiza el proceso de medición de tamaño funcional utilizando COSMIC (versión 3.0), presentando claramente cada una de las fases del método y las actividades que se deben realizar en cada fase. Además, la Figura 1 muestra las entradas y las salidas de cada fase de este método de medición.

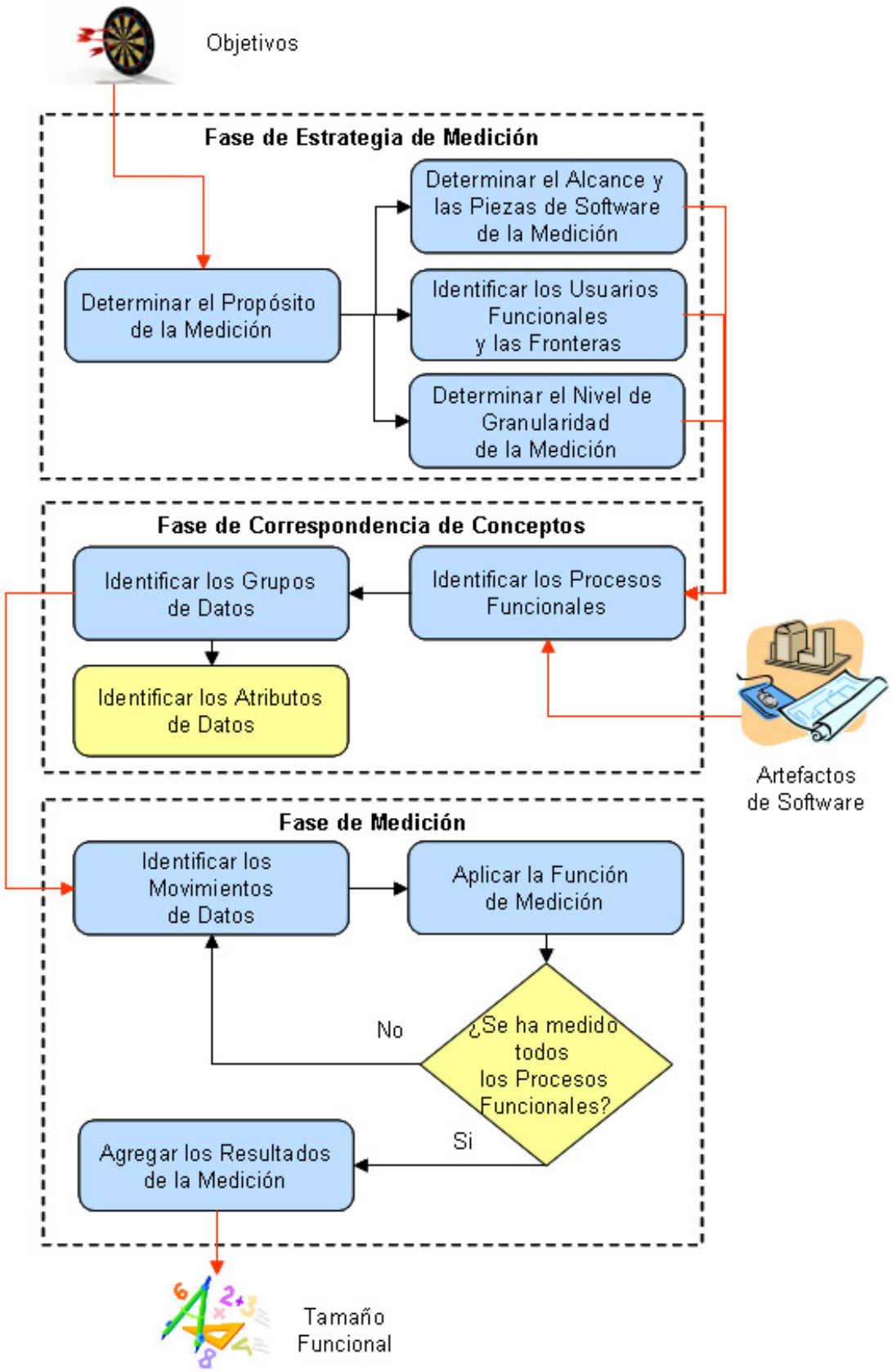


Figura 1. Proceso de medición de Tamaño Funcional usando COSMIC.

3.2 El Método de Desarrollo de Sistemas OO-Method

OO-Method es un método orientado a objetos que permite la generación automática de software a partir de modelos conceptuales [Pastor07]. Este método pone en práctica la tecnología MDA (Model Driven Architecture) [Miller03], separando claramente la lógica de las aplicaciones de la plataforma tecnológica en que serán desarrolladas esas aplicaciones. Para realizar esto, OO-Method provee un conjunto de modelos conceptuales que permiten abstraer la lógica de las aplicaciones de la plataforma de implementación, centrando el proceso productivo en el espacio del problema, que define qué es lo que hace el sistema, en lugar del espacio de la solución, que define cómo se implementa la solución.

El proceso de producción de software utilizando OO-Method comprende cuatro grandes fases, que en términos de MDA respectivamente corresponden al desarrollo de modelos independientes de computación (CIM), modelos independientes de plataforma (PIM), modelos específicos de plataforma (PSM) y modelos de implementación (IM). En la fase de desarrollo de modelos independientes de computación se encuentra el Modelo de Requisitos [Insfran02] [Diaz05], el cual es realizado por el analista del sistema con la ayuda de la herramienta RETO [RETO]. En la fase de desarrollo de los modelos independientes de plataforma se encuentra el Modelo Conceptual, el cual es desarrollado por el analista del sistema con la ayuda de herramientas CASE que facilitan el modelado [Pastor04], el intercambio de modelos [Marin07], la validación de los modelos y la posterior generación de la documentación asociada al sistema que se está modelando. Estas herramientas han sido desarrolladas por la empresa CARE Technologies y en conjunto son denominadas Olivanova Suite [CARE]. En la fase de desarrollo de modelos específicos de plataforma se encuentra el Modelo de Ejecución [Gómez98], que se realiza de manera completamente automática a partir del Modelo Conceptual y que está soportado por un compilador de modelos desarrollado por la empresa CARE Technologies denominado The Programming Machine [CARE]. La fase de desarrollo del modelo de implementación también se realiza de manera completamente automática por el compilador de modelos The Programming Machine [CARE].

La Figura 2 esquematiza el proceso de producción de software de OO-Method, mostrando claramente los diferentes modelos que lo componen y las transformaciones entre modelos. Esta figura

también refleja las correspondencias entre los modelos OO-Method y los modelos de la arquitectura MDA.

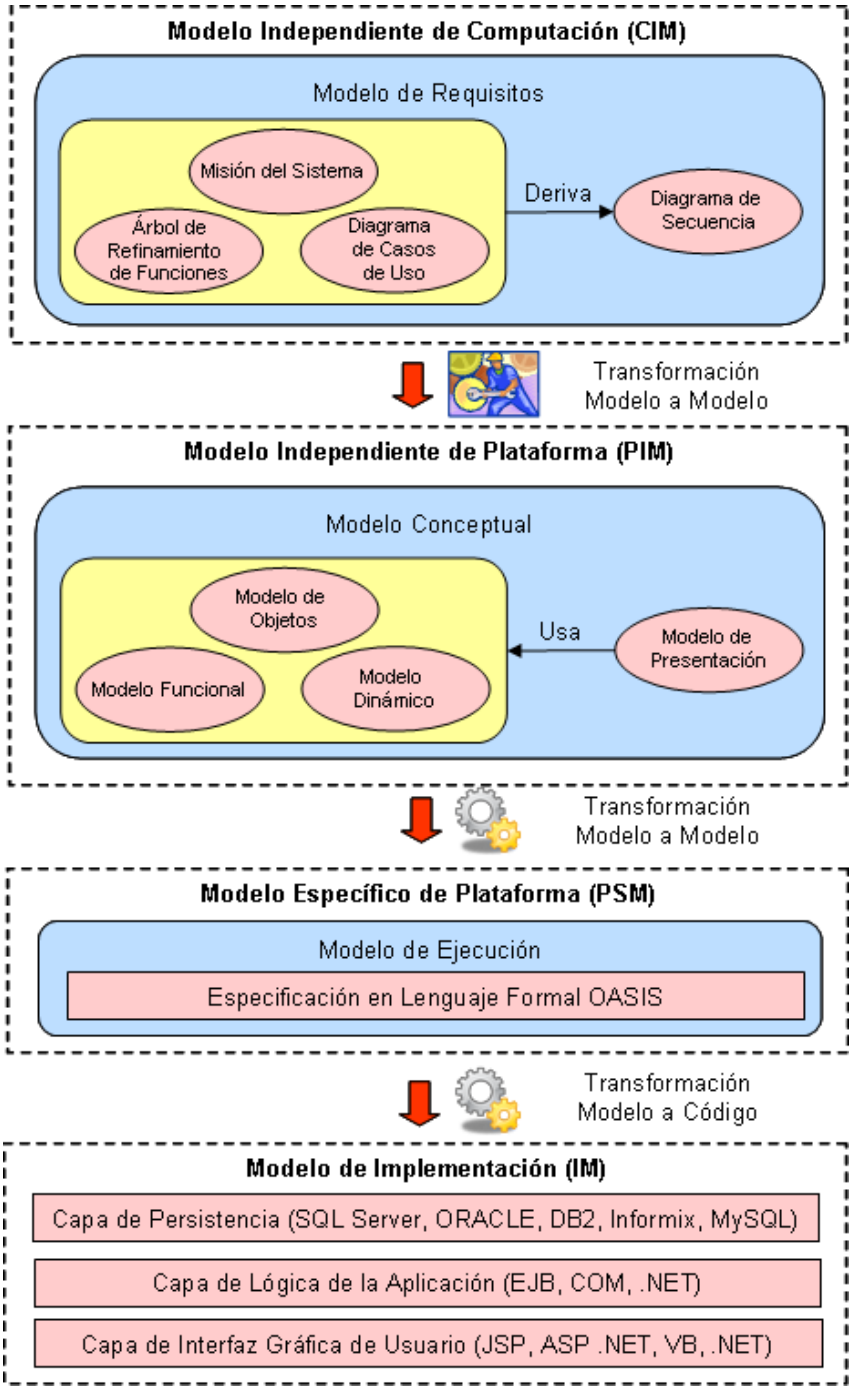


Figura 2. Proceso de Desarrollo de Software utilizando OO-Method.

Como se puede observar en la Figura 2, el modelo de requisitos se compone de técnicas como la definición de la Misión del Sistema, el Árbol de Refinamiento de Funciones, los Diagramas de Casos de Uso y los Diagramas de Secuencia. Con la utilización de estas técnicas es posible definir los requisitos funcionales del sistema, que mediante transformaciones de modelos permiten

conseguir las bases de los diagramas que componen el modelo conceptual.

El modelo conceptual (vea la Figura 2) captura las propiedades estáticas y dinámicas del sistema utilizando un Modelo de Objetos, un Modelo Dinámico y un Modelo Funcional. Además, el modelo conceptual permite la especificación de las interfaces de usuario de manera abstracta a través del Modelo de Presentación. Una vez que el analista ha especificado estos cuatro modelos, el modelo conceptual tiene todos los detalles necesarios para la generación automática de la aplicación de software. La definición completa de los elementos que componen el modelo conceptual de OO-Method está descrita en [Pastor07].

El modelo de ejecución permite especificar la forma en que será utilizado el sistema de manera abstracta, es decir, permite especificar cómo los usuarios accederán al sistema, qué acciones tendrán disponibles cuando accedan al sistema, y qué secuencia de acciones debe realizar para interactuar con el sistema. Para esto el modelo de ejecución utiliza un lenguaje formal denominado OASIS [Pastor92], que permite la especificación exacta de las características de implementación de los objetos del sistema de acuerdo a la forma en que serán utilizados estos objetos.

A partir del modelo de ejecución se obtiene automáticamente el modelo de implementación, que permite la transformación desde el espacio del problema (representado por el modelo conceptual) hacia el espacio de la solución (el producto de software correspondiente). El modelo de implementación realiza correspondencias entre las primitivas conceptuales y sus representaciones de software para un entorno de desarrollo específico, por ejemplo Java o C#. Cabe destacar que el compilador de modelos OO-Method genera aplicaciones en arquitecturas de tres capas: una capa para el componente cliente, que contiene la interfaz gráfica; una capa para el componente servidor, que contiene las reglas de negocio y las conexiones a la base de datos; y una capa para el componente base de datos, que contiene los aspectos de persistencia de las aplicaciones.

Para satisfacer el objetivo de esta tesis necesitamos enfocarnos sólo en el modelo conceptual OO-Method, ya que este modelo será el artefacto a partir del cual mediremos el tamaño funcional de las aplicaciones. La siguiente sección presenta un resumen de los principales conceptos del modelo conceptual.

3.2.1 El Modelo Conceptual OO-Method

El modelo conceptual OO-Method permite la especificación completa de un sistema de manera abstracta utilizando una combinación de técnicas gráficas de modelado orientado a objetos y técnicas de especificación formal. Gracias a las especificaciones formales, el modelo conceptual de OO-Method permite representar el sistema sin ambigüedades y de manera completamente independiente de la plataforma tecnológica.

El modelo conceptual se compone de cuatro modelos complementarios que serán detallados a continuación: el modelo de objetos, el modelo dinámico, el modelo funcional y el modelo de presentación.

Modelo de Objetos

El modelo de objetos permite especificar la estructura del sistema mediante clases, atributos, servicios, restricciones de integridad y relaciones entre clases. La Figura 3 muestra un ejemplo de un modelo de objetos para un sistema de facturación. En esta figura se pueden distinguir las clases *Cliente*, *Factura*, *Detalle* y *Admin*.

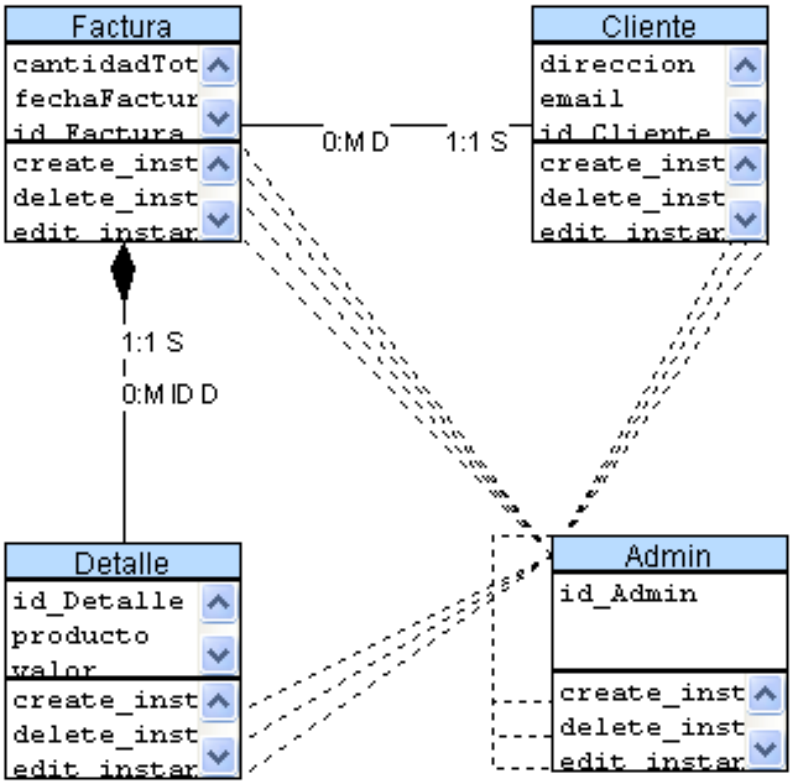


Figura 3. Modelo de Objetos para un Sistema de Facturación.

Los atributos de una clase representan las características de la clase. Los atributos de las clases son dato-valuados, es decir, su valor corresponde a un tipo de datos simple, como por ejemplo: string, real, int, etc. Además, los atributos pueden ser constantes, variables o derivados. Los atributos constantes no permiten cambiar su valor durante la vida del objeto, los atributos variables permiten asignarles diferentes valores durante la vida del objeto, y los atributos derivados calculan su valor según una fórmula de derivación. Finalmente, a cada atributo se le puede asignar un valor por defecto. En la Figura 3 se puede observar que algunos atributos de la clase *Cliente* son *direccion*, *email* e *id_Cliente*.

Los servicios definen el comportamiento de los objetos de una clase. En el modelo de objetos se pueden distinguir los eventos, las transacciones y las operaciones. Los eventos son servicios atómicos, indivisibles, que ocurren en un instante temporal y que pueden asignar valores a los atributos de las clases. Las transacciones son servicios que contemplan una secuencia de eventos y otras transacciones, y que sólo tienen dos formas de terminar la ejecución: se ejecuta correctamente de manera completa o se deshace toda la ejecución si existe algún fallo en la secuencia. Las operaciones son servicios que detallan una secuencia de eventos, transacciones y otras operaciones; y que se ejecutan paso a paso desde el inicio al final, independientemente de si alguno de los pasos ha fallado. En el modelo de objetos de OO-Method es posible distinguir los servicios de creación (que insertan un registro en la base de datos) y los servicios de destrucción (que eliminan un registro en la base de datos) de los demás servicios de la clase. A modo de ejemplo, en la Figura 3 se pueden observar los servicios de la clase *Detalle*: el servicio de creación *create_instance*, el servicio de destrucción *delete_instance* y el servicio de edición *edit_instance*.

Las propiedades básicas de cada servicio son denominadas argumentos del servicio, que pueden ser de entrada (necesarios para la ejecución del servicio) o de salida (respuesta del servicio). Los argumentos pueden ser dato-valuados u objeto-valuados y pueden tener asociado un valor por defecto.

Los servicios del modelo de objetos pueden tener asociadas un conjunto de precondiciones que permitirán verificar si es posible ejecutar un servicio cuando se cumplen un conjunto de condiciones y especificar el mensaje de error que debe ser entregado al usuario de la aplicación cuando las condiciones no se cumplen.

Las restricciones de integridad son restricciones que deben cumplir todos los objetos de una clase en cualquier instante de la

vida del objeto. El modelo de objetos permite definir las restricciones de integridad para cada clase y el mensaje de error que debe mostrarse al usuario de la aplicación cuando intenta realizar alguna acción que viola alguna de las restricciones de integridad especificadas para la clase que esté utilizando.

En el modelo de objetos, las relaciones entre clases se especifican a través de asociaciones, especializaciones y relaciones con agentes. Las asociaciones representan relaciones entre dos clases. Cada asociación tiene dos extremos, que respectivamente corresponden a las clases que participan en la asociación. Las asociaciones pueden ser especializadas como agregaciones o composiciones, las cuales son relaciones del tipo es-parte-de que se diferencian entre sí porque la composición incorpora dependencia de identificación entre las clases relacionadas. Por ejemplo, la Figura 3 muestra una *asociación* entre las clases *Cliente* y *Factura*, y una *composición* entre las clases *Factura* y *Detalle*. Un ejemplo de *agregación* sería la asociación existente entre la clase *Coche* y la clase *Rueda*, ya que las ruedas son parte de un coche, con la salvedad de que su existencia no depende de la existencia del coche.

Las especializaciones corresponden a asociaciones entre una clase genérica (padre) y una clase específica (hijo) que hereda las propiedades de la clase genérica. Para especificar las especializaciones del modelo de objetos, es necesario especificar los eventos que crean la especialización (evento *carrier*) entre la clase padre y la clase hija, y los eventos que destruyen esta especialización (evento *liberator*).

Finalmente, las relaciones con agentes permiten especificar qué clases (de tipo agente) pueden ejecutar los servicios de las clases del modelo de objetos. En la Figura 3, las relaciones de agentes están representadas con líneas punteadas desde la clase *Admin* hacia las demás clases del sistema de facturación.

Modelo Dinámico

El modelo dinámico permite especificar la comunicación entre los objetos del sistema. Cuando el modelo de objetos ya ha sido especificado, es posible definir las relaciones entre objetos del sistema, que pueden pertenecer a la misma clase o a clases diferentes. Para especificar estas relaciones, el modelo dinámico utiliza el diagrama de transición de estados y el diagrama de interacción de objetos. En los siguientes párrafos se detallan cada una de estas perspectivas de dinamismo de un sistema.

Diagrama de Transición de Estados

Este diagrama permite especificar la secuencia de servicios válida en la vida de los objetos de cada clase. Para esto, este diagrama consta de un estado inicial, un estado final, uno o más estados intermedios y transiciones que permiten pasar de un estado a otro. La Figura 4 muestra un ejemplo de diagrama de transición de estados para los objetos de la clase *Factura* del sistema de facturación.

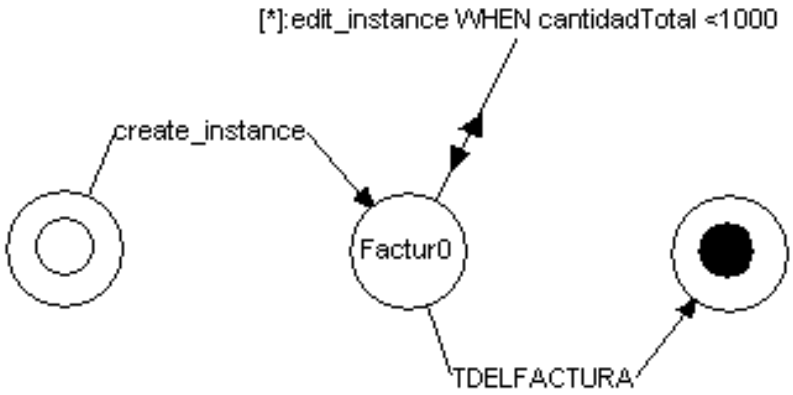


Figura 4. Diagrama de Transición de Estados para clase Factura.

El estado inicial del diagrama de transición de estados representa la situación de los objetos justo antes de ser creados. Su representación gráfica es un circulo con un circulo blanco dentro (vea la Figura 4).

El estado final representa la situación de los objetos inmediatamente después de ser destruidos. Su representación gráfica es un circulo con un circulo negro dentro (vea la Figura 4).

Los estados intermedios representan las situaciones en las que puede encontrarse un objeto a lo largo de su vida, es decir, los estados intermedios son los estados válidos para los objetos de una clase. En la Figura 4 se puede observar el estado intermedio *Factur0*. En este estado intermedio se pueden editar los objetos de la clase *Factura*.

Las transiciones representan el cambio de estado de un objeto, es decir, representan el movimiento de un objeto desde una situación a otra situación. Este movimiento se produce porque un agente ha activado un servicio de la clase a la que corresponde el objeto. Además, las transiciones pueden tener asociada una condición de control, que es una fórmula bien formada que permite

determinar bajo qué condiciones un agente puede activar un servicio. Por ejemplo, la Figura 4 muestra que el servicio *edit_instance* de la clase *Factura* puede ser ejecutado por cualquier agente (representado por el símbolo [*]) cuando la cantidad total de la factura sea menor a 1000 (representada por la sentencia WHEN cantidadTotal < 1000). Las transiciones que van desde el estado inicial a un estado intermedio siempre están asociadas a la activación de servicios de creación, y las que van desde un estado intermedio al estado final están asociadas a servicios de destrucción. Las transiciones que ocurren entre los estados intermedios pueden estar asociadas a cualquier servicio de la clase.

Diagrama de Interacción de Objetos

Este diagrama permite especificar cuándo es posible la comunicación entre objetos de diferentes clases del sistema. Para esto, este diagrama permite especificar triggers y servicios globales.

Los triggers son mecanismos de comunicación entre objetos que se producen cuando una condición es verdadera y el objeto que la cumple dispara eventos o transacciones sobre sí mismo o sobre otros objetos del sistema.

Los servicios globales, ya sean transacciones u operaciones, permiten especificar secuencias de servicios de múltiples clases constituyendo una unidad de ejecución. Al igual que los servicios locales, los servicios globales pueden tener argumentos de entrada y de salida, valores por defecto para esos argumentos, y precondiciones.

Modelo Funcional

El modelo funcional permite especificar la forma en que la ejecución de un servicio afecta al estado de los objetos relacionados a su ejecución. Para esto, el modelo funcional permite especificar evaluaciones de los atributos de una clase en los eventos de esa clase.

Las evaluaciones describen el efecto que produce un evento sobre un atributo mediante una fórmula bien formada. Además, es posible especificar condiciones que se deben cumplir antes de cambiar el valor de un atributo, también utilizando fórmulas bien formadas.

La forma de cambiar el valor de un atributo puede ocurrir de varias maneras. Por esta razón, las evaluaciones tienen asociada una categoría, que permite distinguir cómo se cambiará el valor del atributo. Las categorías que pueden tener las evaluaciones son: estado, cardinal y situación. Las evaluaciones de tipo estado permiten cambiar el valor de un atributo sin tener en cuenta el valor previo del atributo. Las evaluaciones de tipo cardinal permiten incrementar o decrementar el valor de un atributo cuantificable. Finalmente, las evaluaciones de tipo situación permiten cambiar el estado de un atributo teniendo en cuenta el valor actual del atributo.

Para el ejemplo del sistema de facturación (vea la Figura 3), el servicio *edit_instance* de la clase *Detalle* permite cambiar el atributo *producto* (que corresponde al nombre del producto) por un nombre ingresado por el usuario de la aplicación en el argumento *p_producto*. La siguiente tabla resume esta evaluación:

Tabla 6. Evaluación para cambiar el producto del detalle de la factura.

Clase	Evento	Atributo	Condición Evaluación	Efecto Evaluación	Categoría Evaluación
Detalle	edit_instance	Producto		p_producto	estado

Modelo de Presentación

El modelo de presentación permite especificar de manera abstracta la interfaz gráfica de usuario que tendrá la aplicación modelada. Para realizar esto, el modelo de presentación cuenta con un conjunto de patrones de presentación que están organizados jerárquicamente en tres niveles: estructura de acceso, unidades de interacción y elementos básicos.

El primer nivel permite especificar la estructura de acceso al sistema. En este nivel se especifica el menú que tendrá cada agente del sistema mediante un árbol jerárquico de acciones (Hierarchy Action Tree – HAT).

El segundo nivel permite especificar las unidades de interacción del sistema. Las unidades de interacción pueden ser:

- *Unidades de Interacción de Servicio (UIS)*: Representan la interacción entre el usuario de la aplicación y la ejecución de un servicio del sistema.

- *Unidades de Interacción de Instancia (UIS)*: Representa la interacción entre el usuario de la aplicación y la información de un objeto específico.

- *Unidades de Interacción de Población (UIP)*: Representa la interacción entre el usuario de la aplicación y la información de varios objetos de una clase.

- *Unidades de Interacción de Maestro Detalle (UIMD)*: Representa la interacción entre el usuario de la aplicación y un grupo de unidades de interacción del sistema, siguiendo una estructura de maestro – detalle.

El tercer nivel permite especificar los elementos que pueden componer las unidades de interacción del sistema. Estos elementos pueden ser:

- *Entrada*: Este elemento permite capturar los datos que deben ser ingresados por el usuario del sistema.

- *Selección definida*: Este elemento permite al usuario seleccionar un valor desde una colección de datos existente.

- *Agrupador de argumentos*: Este elemento permite definir la forma en que los argumentos de los servicios serán presentados al usuario.

- *Dependencia de argumentos*: Permite definir relaciones de dependencias entre los valores o estado de un argumento de entrada de un servicio y los valores o estado de otros argumentos de entrada del mismo servicio. La dependencia de argumentos utiliza reglas de tipo ECA (Evento – Condición – Acción) para cada argumento de entrada de un servicio, que tienen la siguiente semántica: cuando un evento de interfaz ocurre en un argumento de entrada (por ejemplo cuando el usuario ingresa un valor) se realiza una acción si se cumple una determinada condición.

- *Precarga de argumentos*: Este elemento permite definir un conjunto de objetos que pueden ser seleccionados como argumentos de entrada de un servicio.

- *Filtro*: Este elemento permite definir una condición de selección sobre una población de objetos de una clase. El filtro puede tener variables dato-valuadas y variables

objeto-valuadas que pueden tener definido un valor por defecto, una población asociada para seleccionar los valores de las variables objeto-valuadas y precarga de los valores de las variables objeto-valuadas.

- *Criterio de ordenación*: Este elemento permite definir el orden de los objetos presentados en una población, considerando el orden ascendente/descendente sobre los valores de los atributos de los objetos presentes en la población.

- *Conjunto de visualización*: Permite especificar las propiedades de una clase que serán visibles al usuario, y el orden en que se mostrarán dichas propiedades.

- *Navegación*: Permite la visualización de información de los objetos de las clases relacionadas a un objeto de una clase.

- *Acción*: Permite la ejecución de los servicios definidos en las clases sobre un objeto de la clase.

- *Filtrado Navegacional*: Permite acotar la navegación hacia los objetos de las clases relacionadas a un objeto según una condición de filtro.

- *Inicialización de Argumentos*: Permite asignar un valor inicial a los argumentos de entrada de un servicio que se accede directamente desde otro servicio.

- *Navegación Condicional*: Permite navegar a las unidades de interacción luego de la ejecución de un servicio que haya terminado de manera exitosa o fallida. Para seleccionar la unidad de interacción a la que se navegará, es necesario especificar una condición que debe cumplirse cuando se haya ejecutado el servicio.

Los elementos del tercer nivel se pueden agrupar en elementos *hoja* (que no se pueden descomponer en patrones de presentación) y elementos *intermedios* (que se pueden descomponer en patrones de presentación). Los elementos hoja son: entrada, selección definida, agrupador de elementos, dependencia de argumentos, criterio de ordenación, conjunto de visualización, filtrado navegacional, e inicialización de argumentos. Los elementos intermedios son: filtro, navegación, acción, precarga de argumentos, y navegación condicional.

La Figura 5 esquematiza los tres niveles jerárquicos del modelo de presentación y los patrones que se pueden utilizar en cada nivel, representando en color celeste los elementos que pueden contener a otros y en color amarillo los elementos hoja.

Como se puede ver en la Figura 5, una unidad de interacción de servicio puede tener patrones de entrada, selección definida, agrupador de argumentos, dependencia de argumentos, precarga de argumentos y navegación condicional. A su vez, la precarga de argumentos puede tener asociada una unidad de interacción de población y un criterio de ordenación. Asimismo, la navegación condicional puede tener asociadas unidades de interacción de servicio, de instancia, de población y de maestro detalle, y también inicialización de argumentos y filtrado navegacional.

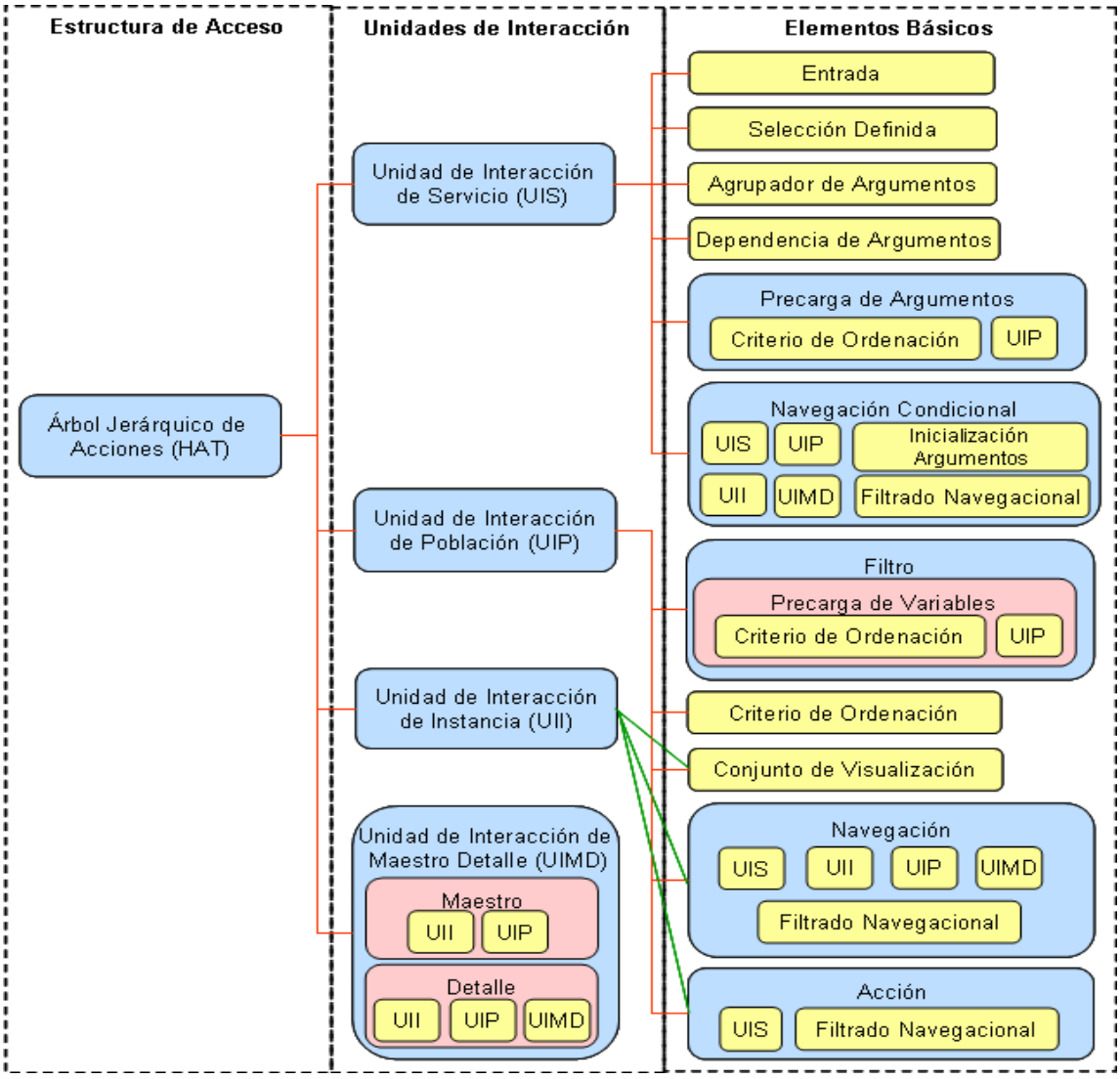


Figura 5. Niveles y elementos del modelo de presentación (Extendido de [Molina03]).

Las unidades de interacción de población pueden tener patrones de filtro, criterio de ordenación, conjunto de visualización, navegación y acción (vea la Figura 5). Los filtros pueden tener precarga en las variables de filtro, las acciones pueden tener asociadas unidades de interacción de instancia y filtrado navegacional, y por último, las navegaciones pueden tener asociadas unidades de interacción de servicio, de instancia, de población y de maestro detalle, y también filtrado navegacional.

Las unidades de interacción de instancia pueden tener patrones de conjunto de visualización, navegación y acción (vea la Figura 5). Al igual que en las unidades de interacción de población, los patrones de navegación y acción están compuestos por otros patrones de presentación.

En último lugar, la Figura 5 muestra que las unidades de interacción de maestros detalle pueden tener patrones de unidad de interacción de instancia y de población en su parte maestra; y unidades de interacción de instancia, de población, o de maestro detalle en su parte de detalle.

3.3 Modelo de Procesos para la Medición de Software

El modelo de procesos para la medición del software propuesto por Jacquet y Abran [Jacquet97] [Abran99] está formado por cuatro pasos: diseño del método de medición, aplicación del método de medición, análisis de los resultados de la medición, y explotación de los resultados (vea la Figura 6).

El *primer paso* consiste en diseñar el método de medición, y se basa en el supuesto de que para desarrollar un método de medición se debe empezar por la definición de objetivos y se debe finalizar con la caracterización numérica de un concepto de software. Para completar el diseño de un procedimiento de medición, es necesario realizar la definición de objetivos, la caracterización del concepto que será medido, el diseño o selección del metamodelo, y la definición de reglas de asignación numérica. En la definición de objetivos es importante por lo menos especificar qué es lo que se quiere medir, qué tipo de artefacto será medido y quién realizará la medición. En la caracterización del concepto que será medido se debe especificar claramente lo que se quiere medir, por ejemplo se pueden medir distancias, rangos matemáticos, funcionalidad, etc. En el diseño o selección del metamodelo se deben definir los conceptos que el método de

medición utilizará para medir los conceptos del software. Finalmente, en la definición de reglas de asignación numérica se deben relacionar los conceptos del metamodelo del método de medición con los conceptos del software, asignándoles un número.

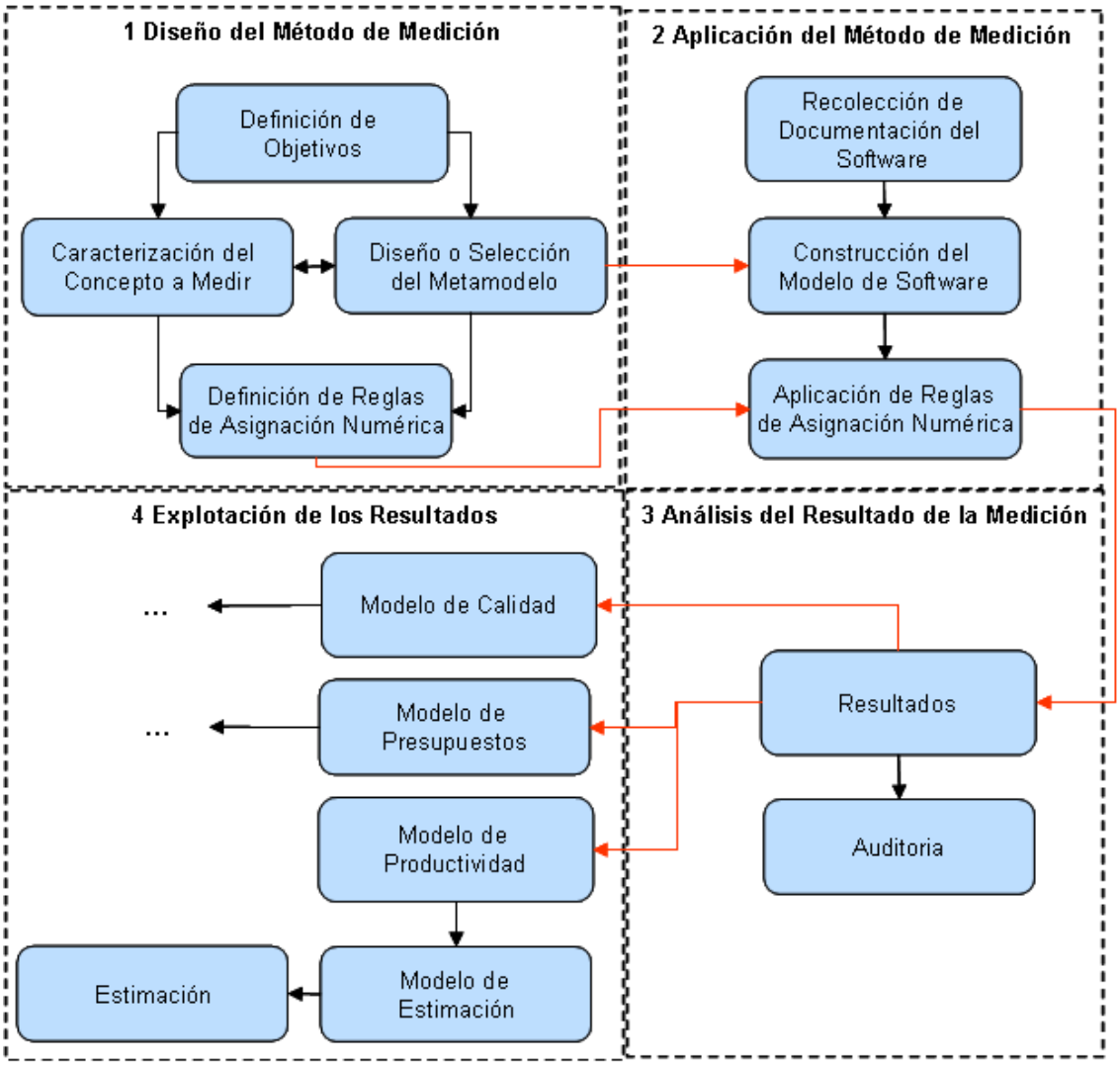


Figura 6. Modelo de Procesos para la Medición del Software.

En el *segundo paso*, el procedimiento de medición diseñado debe ser aplicado al producto de software o a una pieza del producto de software. La aplicación del procedimiento de medición se compone de: recolección de la documentación del software, construcción del modelo de software, y aplicación de reglas de asignación numérica. La recolección de la documentación del software a ser medido consiste en seleccionar los artefactos de software desde los cuales se realizará la medición. La construcción del modelo de software consiste en instanciar el metamodelo del método de medición con los conceptos identificados en la documentación recolectada. En último lugar, la aplicación de reglas

de asignación numérica consiste en aplicar las reglas diseñadas al modelo de software derivado en el paso anterior.

La aplicación del procedimiento de medición permite obtener un resultado de la medición. En *el tercer paso*, se documenta y audita este resultado. Este paso se divide en presentación de los resultados de la medición y auditoria de los resultados de la medición. La presentación de los resultados consiste en la documentación de los resultados, y la auditoria de los resultados consiste en asegurar la calidad del resultado obtenido según la evaluación de ese resultado a través de diferentes métodos, por ejemplo métodos matemáticos, métodos empíricos, etc.

En el *cuarto paso*, los resultados obtenidos de la aplicación del procedimiento son utilizados en diferentes áreas, por ejemplo: en modelos de calidad, en procesos de estimación, en modelos de productividad, en modelos de presupuestos, etc.

3.4 Conclusiones

En este capítulo se han presentado los trabajos relacionados al desarrollo de esta tesis: el método de medición COSMIC, el método de desarrollo de software OO-Method, y un modelo de procesos para la definición de procedimientos de medición.

Se ha seleccionado el método de medición COSMIC por las ventajas que presenta frente a otros métodos de medición basados en el Análisis de Puntos de Función (FPA) [Albretch79]. Algunas de estas ventajas son:

- Utiliza una función matemática para agregar el tamaño funcional de los procesos funcionales especificados en un modelo conceptual, sin estar limitada por valores máximos como es el caso de los métodos basados en FPA.
- Permite la medición de toda la funcionalidad de las aplicaciones debido a que considera los movimientos de datos en la aplicación, no sólo la funcionalidad que el usuario (humano) observa como es el caso de los métodos basados en FPA.
- Permite la medición de aplicaciones generadas en varias capas, permitiendo la medición de toda la aplicación o de cada capa de la aplicación.

Específicamente se ha seleccionado la versión 3.0 del método de medición COSMIC porque, a diferencia de las versiones anteriores,

presenta tres fases de medición, que permiten estructurar de mejor manera el procedimiento de medición, y porque en esta versión deja de tener relevancia el punto de vista desde el cual se realiza la medición, concentrándose en la funcionalidad requerida por los usuarios funcionales de cada pieza de software de la aplicación.

Se ha seleccionado el método de desarrollo de software OO-Method para aplicar el procedimiento de medición porque es un método que permite generar aplicaciones que funcionan correctamente a partir de las especificaciones conceptuales, sin la necesidad de intervenciones manuales. En otras palabras, gracias al lenguaje formal en el que se apoya este método, es posible definir completamente y sin ambigüedades toda la funcionalidad del sistema de software que será generado en las especificaciones conceptuales. De esta manera, se puede realizar la medición del tamaño funcional de las aplicaciones generadas a partir de las especificaciones conceptuales.

Finalmente, se ha seleccionado el modelo de procesos para la medición del software definido por Jacques y Abran [Jacquet97] [Abran99] porque describe detalladamente cómo se debe diseñar un procedimiento de medición. Aunque el estándar para la medición del tamaño funcional ISO 14143 presente los conceptos que deben ser considerados por un método de medición y los procesos para verificarlo, este estándar no presenta un proceso que indique cómo se debe diseñar el procedimiento de medición.

Esta tesis se ha enfocado a los dos primeros pasos del modelo de procesos para procedimientos de medición: el diseño y la aplicación del procedimiento de medición. Además, para estructurar el objetivo definido en el diseño del procedimiento de medición se ha utilizado la plantilla GQM (Goal/Question/Metric) definida por Basili [Basili88].

Capítulo 4
Diseño de un Procedimiento de Medición de Tamaño Funcional

El diseño de un procedimiento de medición es una fase clave, ya que en esta fase se especifica el objetivo de la medición, el artefacto que será medido, el concepto que será medido, las reglas de medición, y la estrategia que será utilizada para llevar a cabo la medición. Es muy importante realizar correctamente el diseño de un procedimiento de medición (abstrayendo de manera correcta los elementos a ser medidos), ya que en caso contrario, el procedimiento podría no estar midiendo lo que debe medir según lo especificado en el método de medición estándar seleccionado como base. Además, es importante tener en cuenta la influencia directa que tiene el diseño de un procedimiento de medición sobre la aplicación de dicho procedimiento, ya que si el diseño es incorrecto, la aplicación del procedimiento puede ser confusa y finalmente se pueden obtener medidas erróneas.

De acuerdo al modelo de procesos para la medición de software propuesto por Jacquet y Abran [Jacquet97] [Abran99], la fase de diseño de un procedimiento de medición está relacionada con la definición del concepto a medir, las reglas para medir ese concepto, y el artefacto a medir. Esta fase está dividida en cuatro pasos: definición de objetivos, caracterización del concepto a medir, selección del metamodelo, y definición de reglas de asignación numérica.

El procedimiento de medición que se presenta en este capítulo está basado en COSMIC versión 3.0 [Abran07]. Se ha observado que existe una correspondencia entre los pasos del modelo de procesos utilizado para el diseño del procedimiento de medición y los pasos de las fases del método de medición de tamaño funcional COSMIC. Para una mayor claridad en la lectura de este capítulo, a continuación se detallan las correspondencias entre los pasos del proceso y pasos de las fases de COSMIC utilizadas para el diseño del procedimiento de medición.

El paso de *definición de objetivos* se corresponde directamente con la definición del propósito de la fase de estrategia de medición del método COSMIC, pues en ambos se especifica el objetivo del

procedimiento de medición. Para una correcta definición de objetivos, en este paso también se debe determinar el artefacto de entrada para realizar la medición y el nivel de detalle requerido en este artefacto para que la medición sea viable. Por esta razón, este paso también se corresponde con la definición del alcance en la fase de estrategia del método COSMIC (en donde se define el artefacto de entrada y se identifican las capas y piezas de software de la aplicación que se quiere medir), y con la definición del nivel de granularidad en la fase de estrategia del método COSMIC (en donde se especifica el nivel de detalle del artefacto de entrada).

El paso de *caracterización del concepto a medir* no presenta correspondencias con ningún paso de las fases del método COSMIC. Esto tiene sentido, ya que el método COSMIC es un método de medición de tamaño funcional, que contempla fases para obtener el tamaño funcional de aplicaciones, sin entrar en detalles de las características de este concepto.

El paso de *selección del metamodelo* corresponde a la identificación de usuarios funcionales y fronteras de la fase de estrategia de COSMIC. También, este paso consiste en realizar la correspondencia de conceptos del metamodelo seleccionado con los conceptos del artefacto de software que será medido, por lo que se corresponde directamente con la fase de correspondencia de conceptos del método COSMIC, y la identificación de movimientos de datos de la fase de medición de COSMIC.

Finalmente, el paso de *definición de reglas de asignación numérica* corresponde a la definición y aplicación de las reglas de medición de la fase de medición del método COSMIC.

La Figura 7 esquematiza los pasos de la fase de diseño de un procedimiento de medición y su solape con las actividades de las fases de medición del método COSMIC.

Las siguientes secciones presentan los pasos que componen la fase de diseño del procedimiento de medición, que han sido seguidos rigurosamente para diseñar el procedimiento de medición de tamaño funcional *OO-Method COSMIC Function Points* (OOmCFP).

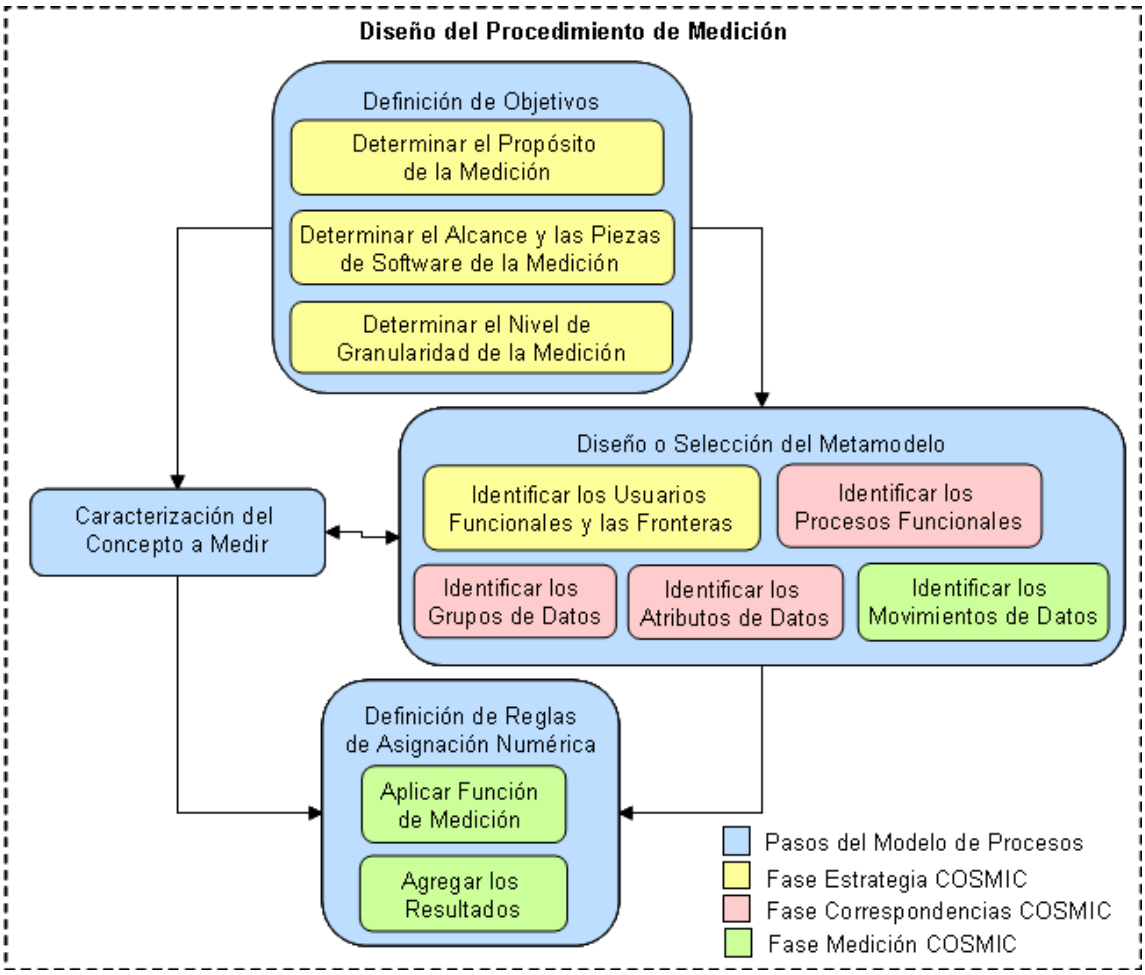


Figura 7. Fase de diseño de un procedimiento de medición y solape con las actividades de las fases del método COSMIC.

4.1 Definición de Objetivos

En este paso de la fase de diseño del procedimiento de medición se debe definir el objetivo del procedimiento. Este paso se corresponde directamente con la tarea de determinar el propósito de la medición, el alcance y las piezas de software, y el nivel de granularidad de la fase de estrategia de medición de COSMIC.

La fase de estrategia de medición de COSMIC es una fase relevante en el procedimiento de medición debido a que en esta fase se define qué se quiere medir, porqué se quiere medir, cuál es el nivel de detalle que deben tener los artefactos a ser medidos, cuáles son las piezas de software que contienen los procesos funcionales y quiénes son los usuarios de esas piezas funcionales.

Para completar la fase de estrategia de medición es necesario determinar el propósito, el alcance y el nivel de granularidad de las mediciones, e identificar los usuarios funcionales y las fronteras. Como se puede observar en la Figura 7, en este paso del diseño del procedimiento de medición se determina el propósito de la medición, el alcance de la medición y el nivel de granularidad del los artefactos de software que serán utilizados para la medición.

4.1.1 Propósito

Según la definición del propósito de la medición (que determina porqué se requiere la medición y qué se realizará con los resultados obtenidos), el *propósito* de una medición con el procedimiento de medición OOmCFP está definido en términos de la plantilla GQM (Goal/Question/Metric) [Basili88] para la medición del software orientada a objetivos como:

Definir un procedimiento de medición
con el propósito de medir una aplicación generada en un entorno MDA
con respecto a su tamaño funcional
desde el punto de vista del desarrollador
en el contexto del Modelo Conceptual OO-Method.

4.1.2 Alcance

El alcance de la medición define un conjunto de requisitos funcionales de usuario que serán incluidos en una medición particular. Como se puede desprender del objetivo planteado en la Sección 4.1.1, el procedimiento de medición OOmCFP utiliza modelos conceptuales OO-Method como artefacto de entrada para medir el tamaño funcional de las aplicaciones generadas en entornos MDA. Estos modelos conceptuales especifican los requisitos funcionales de las aplicaciones generadas de manera independiente de las características de diseño de las aplicaciones y de la tecnología en la se generará cada aplicación. Como se ha detallado en la Sección 3.2, el modelo conceptual OO-Method está compuesto por cuatro modelos (Objetos, Funcional, Dinámico y Presentación), que en conjunto permiten especificar de manera formal y sin ambigüedad los requisitos funcionales de aplicaciones que serán generadas automáticamente utilizando un enfoque basado en la arquitectura MDA.

Las aplicaciones generadas utilizando el enfoque OO-Method son correspondencias directas del modelo conceptual en la tecnología

que haya sido seleccionada por el analista para su generación automática. Estas aplicaciones no necesitan cambios manuales para su correcto funcionamiento, ya que la especificación completa del sistema queda plasmada en el modelo conceptual. Por esta razón, el *alcance* de la medición será el modelo conceptual OO-Method que mediante sus cuatro modelos (Objetos, Dinámico, Funcional, y Presentación) permite generar automáticamente la aplicación.

Una vez definido el alcance de las mediciones realizadas con el procedimiento de medición OOmCFP se han identificado las *capas* y las *piezas de software* que componen la aplicación. Muchas veces los conceptos de capa, pieza de software y componentes pares suelen ser difíciles de identificar correctamente. Para evitar esto, el manual de medición de COSMIC versión 3.0 [Abran07] define y ejemplifica cada uno de estos conceptos. En este manual, una capa está definida como una partición resultante de la división funcional de una arquitectura de software, que junto con el hardware necesario forma un sistema computacional completo, donde: las capas están organizadas jerárquicamente, existe sólo una capa en cada nivel en la jerarquía, existe una dependencia jerárquica entre los servicios funcionales de la capa superior con los servicios funcionales de la capas subordinadas, y el software que intercambia datos entre dos capas interpreta sólo la parte de los datos que intercambia. Además, este manual de medición define una pieza de software como cada parte del software de una aplicación implantado en cada capa, y cuando dos o más piezas de software que estén la misma capa colaboran para responder un requisito funcional, entonces esas piezas de software serán consideradas como componentes pares.

En base a estas definiciones se han identificado las capas, piezas de software y componentes pares de las aplicaciones OO-Method. Estas aplicaciones están estructuradas en una arquitecturas de tres capas: una capa Cliente, que contiene la interfaz gráfica; una capa Servidor, que contiene la lógica del negocio y las conexiones a la base de datos; y una capa de Base de Datos, que contiene la información persistente de la aplicación. Estas capas cumplen con las condiciones de las capas definidas en el manual de medición de COSMIC: están organizadas jerárquicamente, sólo existe una capa en cada nivel de la jerarquía, existe una dependencia jerárquica entre los servicios funcionales de la capa superior con los servicios funcionales de la capas subordinadas, y el software que intercambia datos entre dos capas interpreta sólo la parte de los datos que intercambia. Por estas razones, las capas Cliente, Servidor, y Base de Datos también son consideradas como capas del procedimiento de medición OOmCFP.

En cada capa de las aplicaciones generadas con OO-Method existe una pieza de software, que puede intercambiar información con las piezas de software de las demás capas y que puede ser generada para un entorno de software diferente. Por esta razón, para las aplicaciones OO-Method se distinguen tres piezas de software que se corresponden directamente con las capas Cliente, Servidor y Base de Datos.

Además, el modelo conceptual OO-Method permite la definición de vistas legadas (que son clases implementadas en otros sistemas) así como la definición de sus relaciones con las clases del sistema. Las vistas legadas representan aplicaciones externas a las aplicaciones OO-Method y corresponden a una pieza de software diferente, denominada pieza de software Sistema Legado. Esta pieza de software se encuentra al mismo nivel que la pieza de software Servidor, no existe jerarquía entre ellas, y se comunican para responder a un requisito funcional. Por esta razón la pieza de software Servidor y la pieza de software Sistema Legado son considerados *componentes pares*.

La siguiente figura esquematiza las capas, las piezas de software y los componentes pares que pueden ser identificados en una aplicación OO-Method:

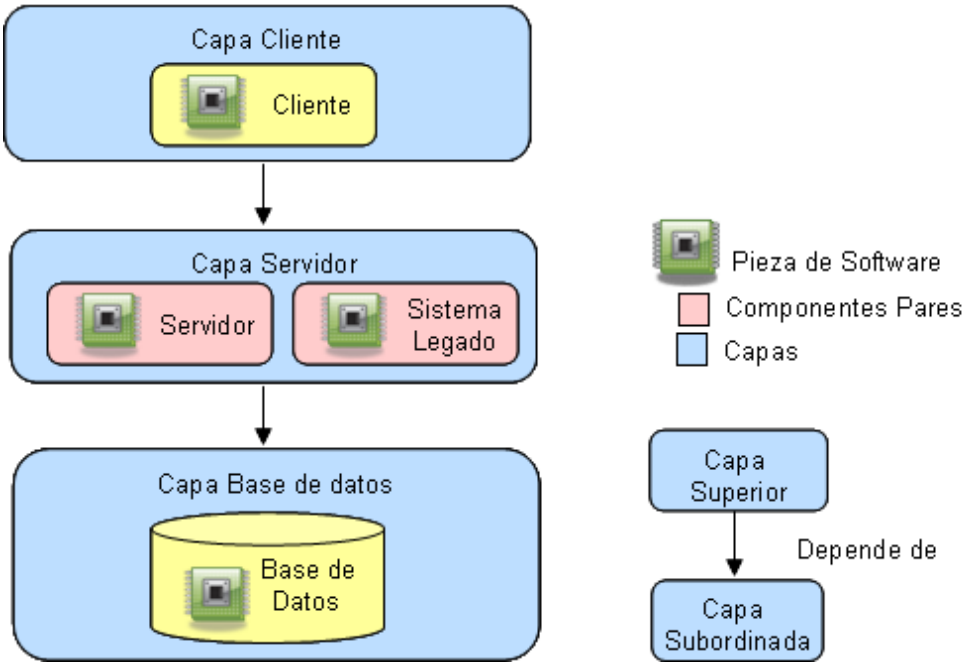


Figura 8. Capas, piezas de software y componentes pares de una aplicación OO-Method.

4.1.3 Nivel de Granularidad

El nivel de granularidad es el nivel de detalle que debe tener cada pieza de software. Dado que las medidas serán realizadas en modelos conceptuales que deben ser válidos para generar aplicaciones de software que satisfacen los requisitos de los usuarios, el *nivel de granularidad* es bajo, ya que se necesita que los requisitos funcionales estén completamente detallados en el modelo conceptual OO-Method para que sea un modelo válido que permita generar automáticamente la aplicación.

4.2 Caracterización del Concepto a Medir

Como se puede desprender del objetivo planteado en la Sección 4.1, el concepto a medir por el procedimiento OOmCFP es el tamaño funcional.

El tamaño funcional ha sido definido por el estándar ISO/IEC 14143-1 [ISO98] como el tamaño del software derivado de la cuantificación de los requisitos funcionales de usuarios. Los requisitos funcionales de los usuarios representan un subconjunto de los requisitos de los usuarios, que especifican qué es lo que la aplicación debe de realizar sin tener en cuenta las características tecnológicas y las características no funcionales de la aplicación (por ejemplo: rendimiento, seguridad, etc.). De esta manera, la *entidad* que será medida por OOmCFP será un Modelo Conceptual OO-Method (que contiene los requisitos funcionales de la aplicación), y el *atributo* que será medido por OOmCFP será el tamaño funcional.

Como se puede observar en la Figura 7, este paso del diseño del procedimiento de medición no se corresponde con ningún paso de las fases de medición del método COSMIC.

4.3 Diseño o Selección del Metamodelo

Un metamodelo es una definición estructural de los elementos que pueden componer un modelo, con sus propiedades y las relaciones entre ellos [Selic07]. Un metamodelo para un procedimiento de medición de tamaño funcional provee las bases para el diseño de las reglas de medición, que identifican y miden los elementos contenidos en el metamodelo.

Se ha seleccionado el metamodelo de COSMIC para el diseño de OOmCFP, ya que en contraste a otros métodos estándares de medición de tamaño funcional (como por ejemplo IFPUG FPA, NESMA FPA o MARK II FPA) presenta múltiples ventajas para medir aplicaciones generadas en entornos MDA a partir de sus modelos conceptuales, como son la simplicidad para cuantificar el tamaño funcional sin estar limitado por valores máximos, la posibilidad de medir toda la funcionalidad de la aplicación (no sólo la que el usuario ve), la posibilidad de medir aplicaciones generadas en varias capas, la posibilidad de medir separadamente cada pieza de software de la aplicación, etc.

La Figura 9 muestra el metamodelo de COSMIC, es decir, los elementos que componen el método de medición COSMIC y sus interrelaciones. Este metamodelo ilustra la información que debe ser representada por el artefacto de software que será medido. Este metamodelo ha sido diseñado a partir del manual de medición para COSMIC versión 3.0 [Abran07].

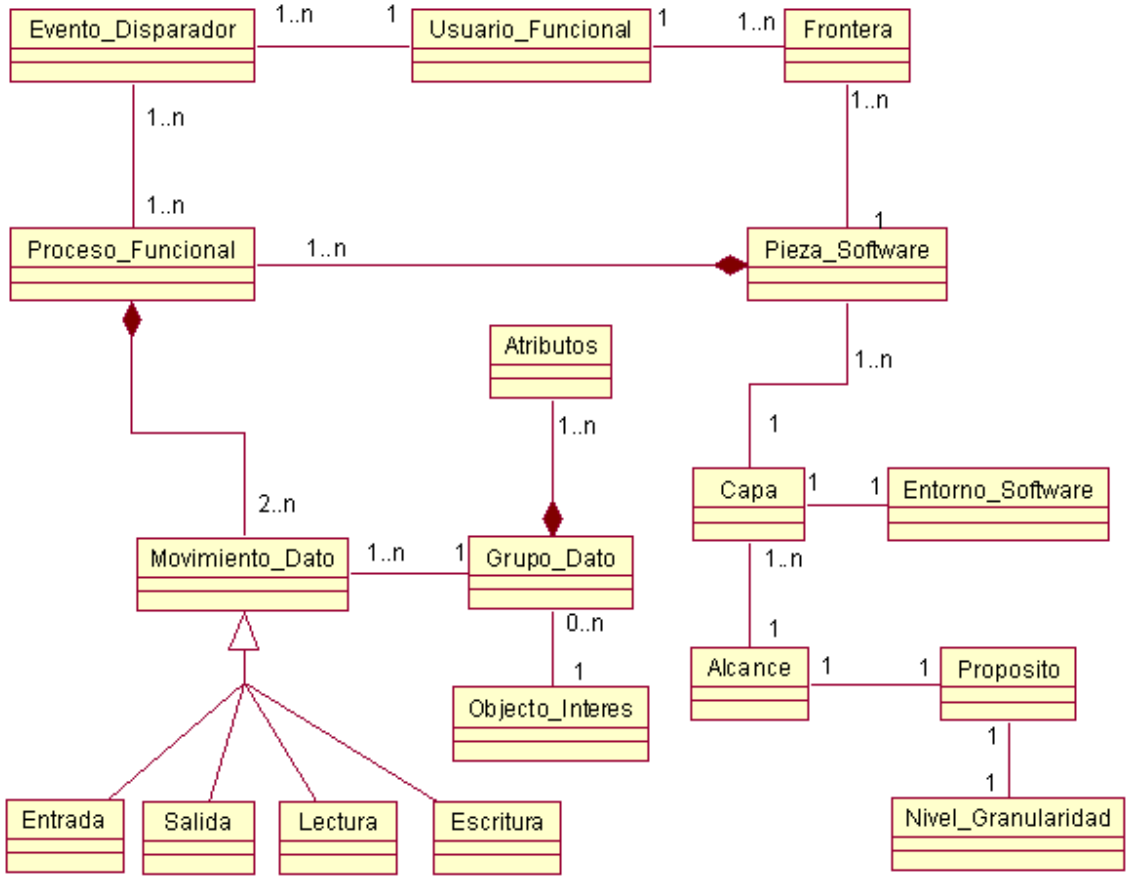


Figura 9. Metamodelo de COSMIC versión 3.0.

Como la Figura 9 muestra, el *alcance* de la medición está determinado por el *propósito* de la medición. El propósito y el

alcance de la medición definen un conjunto de *capas* que serán incluidas en la medición y el nivel de detalle (*nivel de granularidad*) de cada capa que será medida. Cada capa tiene asociado un entorno de software, y dentro de cada capa se pueden identificar diferentes piezas de software.

Analizando la Figura 9 se puede observar que cada medida está enfocada a un conjunto de *objetos de interés* que pueden ser físicos o conceptuales, y que están relacionados a *grupos de datos*. Cada grupo de datos tiene un conjunto de *atributos*. Además, cada grupo de datos participa en uno o más *movimientos de datos*, que pueden ser movimientos de entrada de datos (Entry -E-), movimientos de salida de datos (Exit -X-), movimientos de lectura de datos (Read -R-), y movimientos de escritura de datos (Write -W-). En cada *proceso funcional* ocurren dos o más movimientos de datos. Cada proceso funcional está contenido en las piezas de software identificadas en el propósito de la medición.

Finalmente, la Figura 9 muestra que cada proceso funcional es accionado por *eventos disparadores* que son llevados a cabo por los *usuarios funcionales*. Los eventos disparadores son eventos que ocurren en el mundo real y que incitan a los usuarios a iniciar un proceso funcional. Los usuarios funcionales son los usuarios de las piezas de software que contienen los procesos funcionales. Estos usuarios están separados por una *frontera* de esas piezas de software.

Como se puede observar en la Figura 7, en este paso del diseño del procedimiento de medición OOmCFP se concluye la fase de estrategia de medición del procedimiento mediante la identificación de los usuarios funcionales y las fronteras, se realiza la fase de correspondencia de conceptos, y se comienza con la fase de medición mediante la identificación de los movimientos de datos.

4.3.1 Usuarios Funcionales y Frontera

Los usuarios funcionales son tipos de usuarios que envían o reciben datos en un requisito funcional a una pieza de software [Abran07]. Aplicando esta definición a las aplicaciones OO-Method, los *usuarios funcionales* son los usuarios y las piezas de software de la aplicación.

Los usuarios de las aplicaciones generadas con OO-Method están representados por las clases agente del modelo de objetos de OO-Method. Estos usuarios generalmente son usuarios humanos, y envían o reciben datos a la capa cliente de las aplicaciones. Desde

aquí en adelante, nos referiremos a este usuario funcional como 'usuario funcional humano'.

La pieza de software cliente de las aplicaciones OO-Method es un usuario funcional de la capa de software servidor, ya que envía y recibe datos de esa capa. Por este motivo, la pieza de software cliente se considera como un usuario funcional de las aplicaciones OO-Method, que será llamado 'usuario funcional cliente'.

La pieza de software servidor de las aplicaciones OO-Method es un usuario funcional de la capa cliente de la aplicación y de la capa base de datos de la aplicación, debido a que envía y recibe datos de ambas capas. Este usuario funcional será denominado 'usuario funcional servidor'. Además, el 'usuario funcional servidor' es el encargado de comunicarse con los sistemas legados (enviando y recibiendo datos).

Para identificar los usuarios funcionales de las aplicaciones OO-Method se han definido las siguientes reglas:

Regla 1: Identificar a cada **clase agente** del modelo de objetos OO-Method como un usuario funcional de la capa cliente (usuario funcional humano).

Regla 2: Identificar el usuario funcional **servidor** para la capa cliente de una aplicación OO-Method.

Regla 3: Identificar el usuario funcional **cliente** para la capa servidor de una aplicación OO-Method.

Las fronteras son interfaces conceptuales que existen entre los usuarios funcionales y las piezas de software. En las aplicaciones generadas con OO-Method pueden existir las siguientes *fronteras*: una frontera que separa al 'usuario funcional humano' de la pieza de software cliente , una frontera que separa al 'usuario funcional cliente' de la pieza de software servidor, una frontera que separa al 'usuario funcional servidor' de la pieza de software cliente, una frontera que separa al 'usuario funcional servidor' de la pieza de software base de datos, y una frontera que separa al 'usuario funcional servidor' de la pieza de software legada.

Para identificar las fronteras que puede tener una aplicación OO-Method se ha definido la siguiente regla:

Regla 4: Identificar una frontera entre cada usuario funcional y cada pieza de software de la aplicación OO-Method.

La siguiente figura esquematiza los usuarios funcionales, las piezas de software y las fronteras que pueden ser identificadas en una aplicación OO-Method:

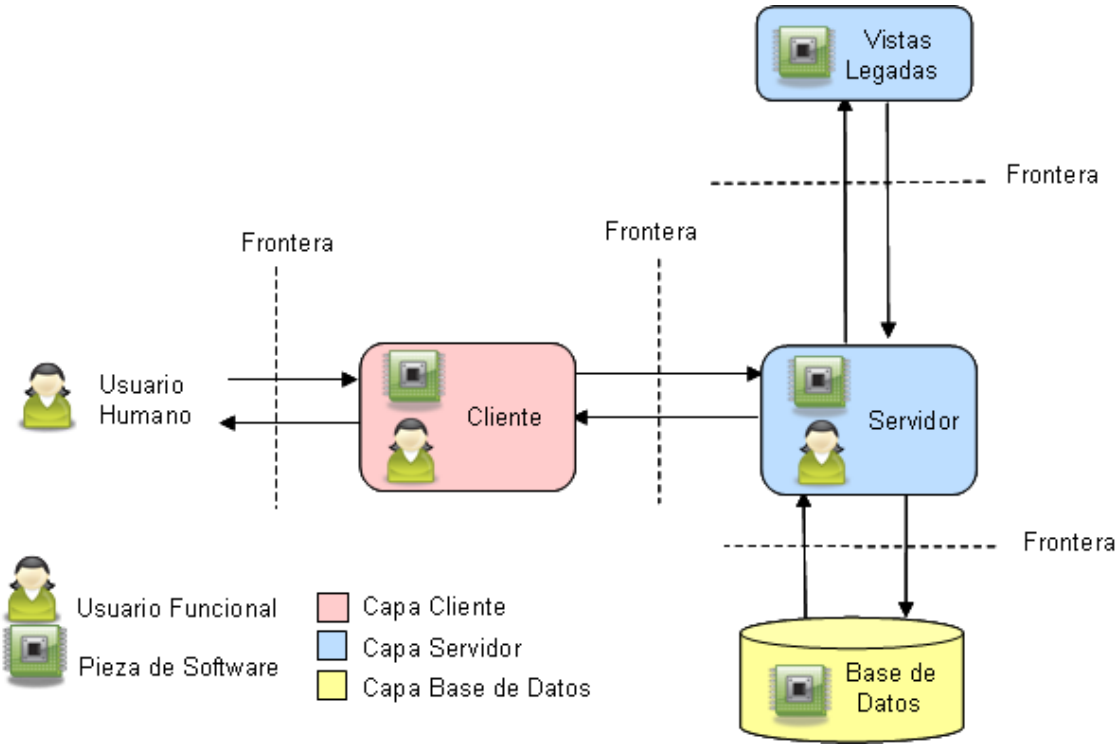


Figura 10. Usuarios funcionales y fronteras en una aplicación OO-Method.

Una vez identificados los usuarios funcionales de las aplicaciones OO-Method, es necesario definir las correspondencias entre los conceptos utilizados en COSMIC y las primitivas conceptuales utilizadas en los cuatro modelos que conforman en el modelo conceptual OO-Method. Para completar la fase de correspondencia de conceptos de COSMIC es necesario identificar los procesos funcionales, los grupos de datos y los atributos de los grupos de datos. A continuación, se presenta cada una de estas tareas.

4.3.2 Procesos Funcionales

Un proceso funcional es un componente elemental de un conjunto de requisitos funcionales, que está compuesto por un conjunto único, cohesivo e independiente de movimientos de datos [Abran07]. Los procesos funcionales se inician con un movimiento de datos de entrada realizado por el usuario funcional debido a que ha ocurrido un evento (evento disparador), y terminan cuando se han ejecutado todos los movimientos de datos necesarios para generar la respuesta a ese evento.

Para el caso de OOmCFP, cada 'usuario funcional humano' lleva a cabo *eventos disparadores* que ocurren en el mundo real. Este usuario funcional comienza procesos funcionales que ocurren en la capa cliente de la aplicación OO-Method. En esta capa, los procesos funcionales son representados por unidades de interacción del modelo de presentación OO-Method, que pueden ser accedidas directamente por el 'usuario funcional humano'.

En el modelo de presentación OO-Method, las funcionalidades que pueden ser accedidas directamente por el 'usuario funcional humano' son las unidades de interacción que se encuentran en el árbol jerárquico de acciones (Hierarchy Action Tree -HAT-). Las unidades de interacción que se encuentran en el HAT pueden ser un conjunto de instancias de una clase (Unidad de Interacción de Población -UIP-), la ejecución de un servicio (Unidad de Interacción de Servicio -IUS-), o unidades de interacción complejas como es un maestro detalle (Unidad de Interacción de Maestro Detalle -UIMD-). De esta manera, se ha definido la siguiente regla para identificar los procesos funcionales que ocurren en la capa cliente de las aplicaciones generadas a partir de modelos conceptuales OO-Method:

Regla 5: Identificar un proceso funcional por cada Unidad de Interacción de Población **(UIP)**, Unidad de Interacción de Servicio **(UIS)** o Unidad de Interacción de Maestro Detalle **(UIMD)** diferente que cuelga directamente del **HAT** del modelo de presentación OO-Method.

Utilizando la regla 5 se pueden identificar los procesos funcionales que puede ejecutar el 'usuario funcional humano'. Sin embargo, además de identificar la unidad de interacción es necesario identificar los elementos que componen esa unidad de interacción que son relevantes para el tamaño funcional, y que por lo tanto, también forman parte del proceso funcional. A estos elementos les llamaremos *elementos contenidos*.

Los procesos funcionales que correspondan a una UIP pueden estar compuestos por conjuntos de visualización, filtros, criterios de ordenación, acciones y navegaciones. A su vez, los filtros, las acciones y las navegaciones pueden contener unidades de interacción, que también forman parte del proceso funcional que se está analizando (vea la Figura 5). Por esta razón, los procesos funcionales UIP deben descomponerse en patrones de presentación y unidades de interacción para poder calcular su tamaño funcional. Para identificar los elementos que pueden estar contenidos en una UIP se ha definido la siguiente regla:

Regla 5.1: Para cada **UIP**, identificar los conjuntos de visualización, los filtros, los criterios de ordenación, las acciones, las navegaciones y las unidades de interacción que transitivamente la componen.

Los procesos funcionales que correspondan a UIS pueden estar compuestos por argumentos (mediante patrones de entrada, selección, agrupador de argumentos, dependencias, precarga) y navegación condicional. Del mismo modo, los argumentos y las navegaciones condicionales pueden estar compuestos por otras unidades de interacción y patrones de presentación que también forman parte del proceso funcional (vea la Figura 5). Para identificar los elementos que pueden estar contenidos en una UIS se ha definido la siguiente regla:

Regla 5.2: Para cada **UIS**, identificar las navegaciones condicionales y las unidades de interacción que transitivamente lo componen.

Los procesos funcionales que corresponden a UIMD pueden estar compuestos por maestros y detalles (vea la Figura 5). Cada uno de estos patrones de presentación a su vez puede estar compuesto de unidades de interacción de instancia (UII), unidades de interacción de población (UIP) y unidades de interacción de maestro-detalle (UIMD). Estas unidades de interacción deben ser identificadas para poder calcular el tamaño funcional de los procesos funcionales que corresponden a un UIMD. Para identificar las unidades de interacción que pueden estar contenidas en un UIMD se ha definido la siguiente regla:

Regla 5.3: Para cada **UIMD**, identificar el maestro, el detalle y las unidades de interacción que transitivamente lo componen.

Cuando un proceso funcional contenga una unidad de interacción de instancia (UII), los elementos que componen la unidad de instancia también serán parte del proceso funcional. Estos elementos pueden ser conjuntos de visualización, acciones y navegaciones (vea la Figura 5). Al igual que en las unidades de interacción de población, las acciones y las navegaciones pueden contener otras unidades de interacción. Estos elementos y unidades de interacción contenidas en una UII también deben ser identificados, por lo que se ha definido la siguiente regla:

Regla 5.4: Para cada **UII**, identificar el conjunto de visualización, las acciones, las navegaciones y las unidades de interacción que transitivamente la componen.

Para identificar completamente un proceso funcional, una vez aplicada la regla 5 se deben identificar los elementos que componen ese proceso funcional. Para esto se debe iterar en la utilización de las reglas 5.1, 5.2, 5.3, y 5.4 que contribuyen a la correcta identificación de los *elementos contenidos* en cada proceso funcional.

En el proceso de identificar las unidades de interacción que pueden estar contenidas en un proceso funcional no se deben tomar en cuenta aquellas que correspondan a otro proceso funcional, ya que dicha unidad de interacción será analizada de forma particular (debido a que también cuelga directamente del HAT). Para evitar contabilizar el tamaño funcional de unidades de interacción que corresponden a procesos funcionales como elementos contenidos en otro proceso funcional, se ha definido la siguiente regla:

Regla 6: Para cada proceso funcional, eliminar las unidades de interacción que correspondan a un proceso funcional del conjunto de elementos contenidos.

Un proceso funcional puede estar compuesto por varias unidades de interacción, a las cuales se accede mediante los patrones de presentación contenidos en el proceso funcional. Para evitar la identificación de funcionalidad replicada, se debe tener en cuenta que cada unidad de interacción contenida en un proceso funcional se debe identificar sólo una vez, aunque sea accedida desde diferentes patrones de presentación contenidos en el proceso funcional (Por ejemplo desde acciones, navegaciones, filtros, maestro, detalles, argumentos o navegación condicional). Para evitar la identificación de unidades de interacción duplicadas en un proceso funcional, se ha definido la siguiente regla:

Regla 7: Para cada proceso funcional, eliminar las unidades de interacción repetidas en el conjunto de elementos que contiene.

La Tabla 7 resume los patrones de presentación que pueden estar contenidos en las unidades de interacción y que contribuyen al tamaño funcional de la aplicación.

Tabla 7. Patrones de presentación contenidos en las unidades de interacción.

Unidad de Interacción	Patrón	Elementos contenidos
Unidad de Interacción de Instancia (UII)	Conjunto de visualización	-
	Acciones	UII, UIP, UIMD, UIS
	Navegaciones	UII, UIP, UIMD
Unidad de Interacción de Población (UIP)	Conjunto de visualización	-
	Filtro	UIP
	Criterio de Ordenación	-
	Acciones	UII, UIP, UIMD, UIS
	Navegaciones	UII, UIP, UIMD
Unidad de Interacción de Maestro Detalle (UIMD)	Maestro	UII, UIP
	Detalle	UII, UIP, UIMD
Unidad de Interacción de Servicio (UIS)	Argumentos	UIP
	Navegación Condicional	UII, UIP, UIMD, UIS

El 'usuario funcional cliente' lleva a cabo *eventos disparadores* que ocurren en las unidades de interacción del modelo de presentación del modelo conceptual OO-Method. Este usuario funcional comienza procesos funcionales que ocurren en la capa servidor de la aplicación OO-Method. Los procesos funcionales que ocurren en la capa servidor corresponden a las acciones que realiza esta capa en respuesta a los eventos disparados en las unidades de interacción de la capa cliente. Para una mayor claridad en la identificación de los procesos funcionales y su correspondencia con las unidades de interacción que llevan a cabo los eventos disparadores, los procesos funcionales identificados recibirán el nombre de los procesos funcionales que contienen dichas unidades de interacción. Para nombrar los procesos funcionales de la capa servidor se ha definido la siguiente regla:

Regla 8: Para cada proceso funcional de la capa servidor, asignar el nombre del proceso funcional que contiene las unidades de interacción que disparan los eventos.

Para identificar los elementos que contienen los procesos funcionales de la capa servidor es necesario identificar las acciones que puede realizar esta capa en respuesta a los eventos disparados por las unidades de interacción de la capa cliente.

Las unidades de interacción de instancia (UII) pueden solicitar a la capa servidor los valores de los atributos que componen el conjunto de visualización, la ejecución de un servicio y los valores por defecto asignados a los argumentos de un servicio.

Las unidades de interacción de población (UIP) pueden solicitar a la capa servidor los valores de los atributos que componen el conjunto de visualización, los valores de las variables de filtro que tengan asociado un valor por defecto, la ejecución de un servicio y los valores por defecto asignados a los argumentos de un servicio.

Dado que las unidades de interacción de maestro detalle (UIMD) se descomponen en unidades de interacción de instancia (UII) o unidades de interacción de población (UIP), estas unidades de interacción pueden solicitar a la capa servidor las funcionalidades descritas en los párrafos anteriores.

Las unidades de interacción de servicio (UIS) pueden solicitar a la capa servidor el valor por defecto de los argumentos del servicio, el valor de los atributos derivados utilizados en el servicio, la ejecución del servicio asociado a la unidad de interacción, la ejecución de las reglas de dependencia de los argumentos, la ejecución de las evaluaciones, la ejecución de la navegación condicional para los casos de éxito o error de la ejecución del servicio, la inicialización de argumentos de los servicios destino de la navegación condicional, la ejecución del filtrado navegacional según la fórmula de filtrado, la validación de las precondiciones del servicio, la revisión de las restricciones de integridad de la clase que contiene el servicio, la ejecución de los triggers activados por el servicio, y el cambio de estado del objeto según las transiciones del servicio.

El 'usuario funcional servidor' lleva a cabo *eventos disparadores* que ocurren en la capa servidor de la aplicación OO-Method. Este usuario funcional activa procesos funcionales que corresponden a las acciones que realiza la capa de base de datos, la capa cliente, o los sistemas legados, en respuesta a los eventos disparados en la capa servidor.

La capa de base de datos agrega, borra o modifica la información persistente del sistema. Para esto, la capa de base de datos recibe un evento desde la capa servidor que le indica la acción que debe realizar y los datos correspondientes. Dado que las aplicaciones OO-Method utilizan sistemas gestores de base de datos (SGBD) comerciales (que tienen funcionalidad no generada por el compilador de modelos OO-Method), OOmCFP no considera los procesos funcionales que ocurren en la capa de base de datos.

Sin embargo, dentro de los procesos funcionales de la capa servidor, OOmCFP considera la comunicación con la base de datos para escribir o leer datos mediante los servicios ejecutados por la capa servidor.

En cuanto a la capa cliente, el 'usuario funcional servidor' activa procesos funcionales que permiten desplegar información, por ejemplo en los conjuntos de visualización, en los mensajes de error, etc.

Finalmente, el 'usuario funcional servidor' activa procesos funcionales de los sistemas legados, representados por las vistas legadas en el modelo de objetos del modelo conceptual OO-Method. Este usuario funcional puede solicitar el valor de los atributos de cada clase del sistema legado y ejecutar sus servicios.

4.3.3 Grupos de Datos

Un grupo de datos corresponde a un conjunto de atributos distintos, no vacío, no ordenado y no redundante que describen un aspecto de un objeto de interés.

Los objetos de interés son cosas identificadas desde el punto de vista de los requisitos funcionales. Estos objetos de interés no corresponden a los objetos definidos en la Orientación de Objetos, sino que corresponden a objetos físicos, objetos conceptuales, o incluso partes de un objeto conceptual.

En los modelos conceptuales OO-Method, todos los *objetos de interés* representados son objetos conceptuales. De esta manera, los *grupos de datos* son las clases del modelo de objetos del modelo conceptual OO-Method que participan en un proceso funcional. Para la correcta identificación de los grupos de datos se ha definido la siguiente regla:

Regla 9: Identificar 1 grupo de datos por cada clase que no pertenezca a una jerarquía de herencia y que participa en un proceso funcional.

Sin embargo, si la clase pertenece a una jerarquía de herencia, se identificará al padre de la jerarquía como un grupo de datos y en caso que los hijos en la jerarquía tengan atributos diferentes a los del padre, se identificará un grupo de datos por cada uno de ellos. Para la correcta identificación de los grupos de datos que participan en una jerarquía de herencia se han definido las siguientes reglas:

Regla 10: Identificar 1 grupo de datos por la clase padre de una jerarquía de herencia a la que pertenece una clase que participa en un proceso funcional.

Regla 11: Identificar 1 grupo de datos por cada clase hija con atributos diferentes a los del padre de la jerarquía de herencia a la que pertenece una clase que participa en un proceso funcional.

4.3.4 Atributos

Los atributos son definidos como la pieza de información más pequeña de un grupo de datos. Por esta razón, el conjunto de atributos de cada clase del modelo de objetos OO-Method que participa en un proceso funcional corresponde a los *atributos* del grupo de datos correspondiente a dicha clase. Para identificar los atributos se presenta en la siguiente regla:

Regla 12: Identificar los atributos de las clases que participan en un proceso funcional como atributos del grupo de datos correspondiente a dicha clase.

Con respecto a la fase de medición de OOmCFP, en este paso del diseño se deben identificar los movimientos de datos que ocurren en cada proceso funcional. Por esta razón, a continuación se presenta una sección que detalla la identificación de movimientos de datos que pueden ocurrir en las aplicaciones OO-Method.

4.3.5 Movimientos de Datos

Cada proceso funcional tiene un conjunto de dos o más movimientos de datos. Cada movimiento de datos mueve un grupo de datos singular. Los movimientos de datos pueden ser movimientos de datos de entrada (E), movimientos de datos de salida (X), movimientos de datos de lectura (R), y movimientos de datos de escritura (W).

Un *movimiento de datos de entrada (E)* es un movimiento realizado por el usuario funcional que mueve un grupo de datos a través de la frontera a un proceso funcional que requiere el grupo de datos.

Un *movimiento de datos de salida (X)* es un movimiento realizado por un proceso funcional que mueve un grupo de datos a

través de la frontera a un usuario funcional que requiere el grupo de datos.

Un *movimiento de datos de lectura (R)* es un movimiento que mueve un grupo de datos desde la base de datos al proceso funcional que requiere ese grupo de datos. Se debe tener en cuenta que, dado que las aplicaciones OO-Method son generadas de acuerdo a una arquitectura de tres capas, sólo la capa servidor del software puede leer la base de datos.

Un *movimiento de datos de escritura (W)* es un movimiento que mueve un grupo de datos desde un proceso funcional a la base de datos. Al igual que los movimientos de datos de lectura, en las aplicaciones OO-Method solamente la capa servidor del software puede escribir la base de datos.

La Figura 11 esquematiza los movimientos de datos que pueden ocurrir entre los usuarios funcionales y las piezas de software que pueden tener las aplicaciones generadas utilizando OO-Method.

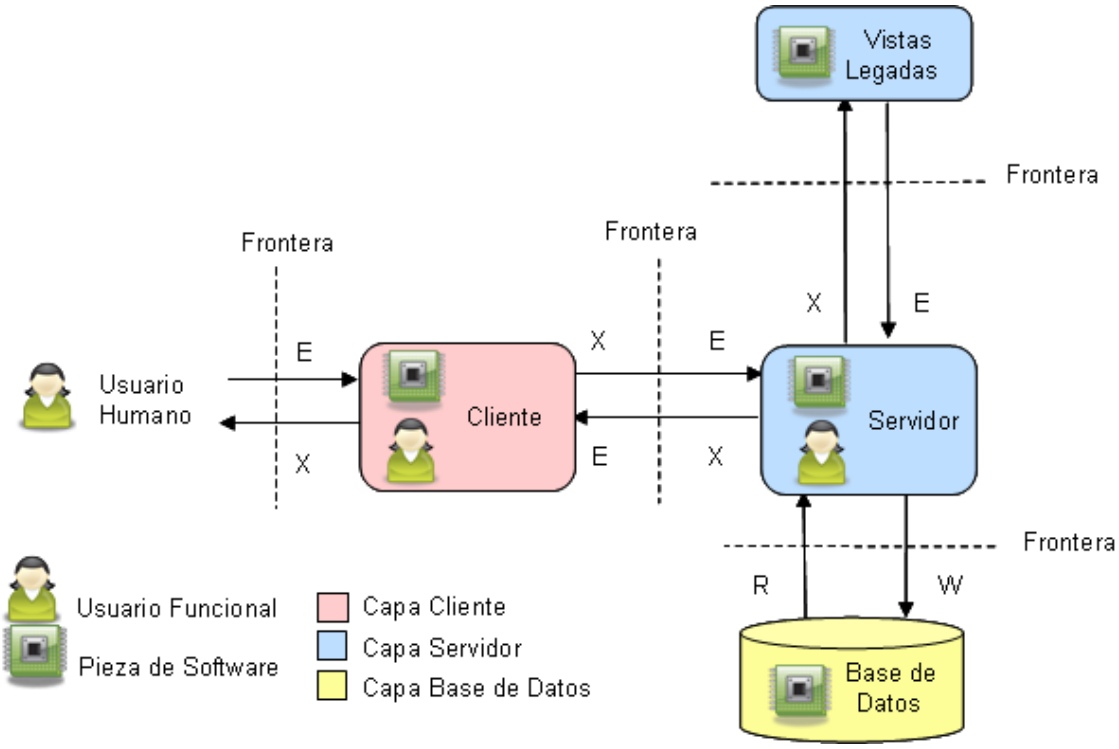


Figura 11. Movimientos de datos que podrían ocurrir en las aplicaciones OO-Method.

Teniendo en cuenta los movimientos de datos que pueden ocurrir en la aplicaciones generadas con OO-Method y las definiciones de los movimientos de datos, se han definido 50 reglas que permiten identificar los movimientos de datos. Las

reglas para identificar los movimientos de datos en cada capa de las aplicaciones OO-Method han sido definidas según la primitiva conceptual donde es posible identificar los movimientos de datos.

Para el caso de las aplicaciones OO-Method, se identifican los movimientos de datos que ocurren en los conjuntos de visualización, en los filtros y en los servicios, ya que las demás primitivas conceptuales de OO-Method no realizan movimientos de datos. De esta manera, a continuación se presentan los movimientos de datos que pueden ocurrir en los conjuntos de visualización, filtros y servicios; y las reglas definidas en OOmCFP para identificarlos.

Conjunto de Visualización

El conjunto de visualización permite desplegar información del sistema al 'usuario funcional humano'. Para realizar esto, en el modelo de presentación de una aplicación OO-Method se definen los atributos de la clase que se quieren desplegar.

Luego, cuando la aplicación OO-Method ha sido generada, si el conjunto de visualización no tiene atributos derivados, para mostrar información al usuario se realizan los siguientes movimientos:

1) La capa servidor lee desde la base de datos los valores de los atributos que serán desplegados en el conjunto de visualización.
2) La capa servidor entrega los valores recuperados a la capa cliente.
3) La capa cliente recibe los valores para el conjunto de visualización desde la capa servidor.
4) La capa cliente despliega los valores al 'usuario funcional humano'.

En caso que el conjunto de visualización tenga atributos derivados, para mostrar información al usuario se realizan los siguientes movimientos:

1) La capa servidor lee desde la base de datos los valores de los atributos que permiten calcular la condición de derivación.
2) Si la condición se cumple, la capa servidor lee desde la base de datos los valores de los atributos que permiten calcular el valor derivado.
3) La capa servidor lee desde la base de datos los valores de los demás atributos que serán desplegados en el conjunto de visualización.

4) La capa servidor entrega los valores recuperados a la capa cliente.
5) La capa cliente recibe los valores para el conjunto de visualización desde la capa servidor.
6) La capa cliente despliega los valores al 'usuario funcional humano'.

Para identificar los movimientos de datos asociados al conjunto de visualización se han definido las siguientes reglas:

Regla 13: Identificar el movimiento de datos 1X para la capa cliente por cada **conjunto de visualización** de una UIP o una UII que participa en un proceso funcional.

Regla 14: Identificar los movimientos de datos: 1E para la capa cliente, y 1X y 1R para la capa servidor, por cada **clase** diferente que contribuya con atributos al **conjunto de visualización** de una UIP o una UII que participa en un proceso funcional.

Regla 15: Identificar los movimientos de datos: 1E para la capa cliente, y 1X y 1E para la capa servidor, por cada **vista legada** diferente que contribuya con atributos al **conjunto de visualización** de una UIP o una UII que participa en un proceso funcional.

Regla 16: Identificar el movimiento de datos 1R para la capa servidor por cada **clase** diferente que sea referenciada en la **condición de la fórmula de derivación** de un atributo derivado desplegado en el **conjunto de visualización** de una UIP o una UII que participa en un proceso funcional.

Regla 17: Identificar el movimiento de datos 1R para la capa servidor por cada **clase** diferente que sea referenciada en el **efecto de la fórmula de derivación** de un atributo derivado desplegado en el **conjunto de visualización** de una UIP o una UII que participa en un proceso funcional.

Regla 18: Identificar los movimientos de datos 1X y 1E para la capa servidor por cada **vista legada** diferente que sea referenciada en la **condición de la fórmula de derivación** de un atributo derivado desplegado en el **conjunto de visualización** de una UIP o una UII que participa en un proceso funcional.

Regla 19: Identificar los movimientos de datos 1X y 1E para la capa servidor por cada **vista legada** diferente que sea referenciada en el **efecto de la fórmula de derivación** de un atributo derivado desplegado en el **conjunto de visualización** de una UIP o una UII que participa en un proceso funcional.

Filtro

Un filtro permite acotar el conjunto de instancias que es desplegado al 'usuario funcional humano' según condiciones sobre los valores de variables de filtro entregados por este usuario. Para realizar esto, en el modelo de presentación de una aplicación OO-Method se define un filtro para una clase determinada. En la definición del filtro se especifican las variables (dato valuadas u objeto valuadas), el valor por defecto para estas variables en el caso que corresponda y la condición que permite acotar el número de registros que será desplegado al 'usuario funcional humano'.

Posteriormente, cuando la aplicación OO-Method ha sido generada, para filtrar información se realizan los siguientes movimientos:

1) La capa servidor calcula el valor por defecto de las variables de filtro dato valuadas y objeto valuadas mediante valores constantes y funciones que se encuentran en esa capa, y entrega los valores calculados a la capa cliente.
2) La capa cliente recibe los valores para las variables de filtro desde la capa servidor.
3) La capa cliente despliega los valores de las variables de filtro al 'usuario funcional humano'.
4) El 'usuario funcional humano' observa los valores por defecto presentados en el filtro e ingresa a la capa cliente los valores que crea convenientes a las variables de filtro.
5) La capa cliente entrega los valores ingresados por el 'usuario funcional humano' a la capa servidor.
6) La capa servidor recibe los valores ingresados para las variables de filtro.
7) La capa servidor lee desde la base de datos la información que necesite para resolver la fórmula del filtro.
8) La capa servidor entrega la información a la capa cliente en el conjunto de visualización de la UIP que participa en un proceso funcional.

Para identificar los movimientos de datos asociados al filtro se han definido las siguientes reglas:

Regla 20: Identificar el movimiento de datos 1E y 1X para la capa cliente, y 1E para la capa servidor (representado por la clase que contiene al filtro) por el conjunto de **variables de filtro dato-valuadas** asociadas al filtro de una UIP que participa en un proceso funcional.

Regla 21: Identificar el movimiento de datos 1E y 1X para la capa cliente, y 1E para la capa servidor por cada **variable de filtro objeto-valuada** asociada al filtro de una UIP que participa en un proceso funcional.

Regla 22: Identificar el movimiento de datos 1R para la capa servidor por cada **clase** diferente que sea referenciada en la **fórmula de filtro** del filtro asociado a una UIP participa en un proceso funcional.

Regla 23: Identificar los movimientos de datos 1X y 1E para la capa servidor por cada **vista legada** diferente que sea referenciada en la **fórmula de filtro** del filtro asociado a una UIP participa en un proceso funcional.

Regla 24: Identificar el movimiento de datos 1E y 1X para la capa cliente, y 1X para la capa servidor (representado por la clase que contiene al filtro) por el conjunto de **variables de filtro dato-valuadas** que tengan **valor por defecto**, asociadas al filtro de una UIP que participa en un proceso funcional.

Regla 25: Identificar el movimiento de datos 1E y 1X para la capa cliente, y 1X para la capa servidor por cada **variable de filtro objeto-valuada** que tenga **valor por defecto**, asociada al filtro de una UIP que participa en un proceso funcional.

Regla 26: Identificar el movimiento de datos 1R para la capa servidor por cada **clase** diferente que sea referenciada en la **fórmula de filtrado navegacional** de cada **variables de filtro objeto valuadas** asociada al filtro de una UIP que participa en un proceso funcional.

Regla 27: Identificar los movimientos de datos 1X y 1E para la capa servidor por cada **vista legada** diferente que sea referenciada en la **fórmula de filtrado navegacional** de cada **variables de filtro objeto valuadas** asociada al filtro de una UIP que participa en un proceso funcional.

Servicio

Un servicio permite cambiar el estado de un objeto. Para realizar esto, en el modelo de objetos de una aplicación OO-Method se define el conjunto de servicios disponibles en cada clase. En la definición de cada servicio se especifican los argumentos del servicio (dato valuados u objeto valuados), las precondiciones del servicio, las evaluaciones del servicio cuando se trata de un evento, y la fórmula del servicio para las transacciones y operaciones. Además, en la clase que contiene el servicio se especifican las restricciones de integridad (condición que deben cumplir todos los objetos de una clase).

Posteriormente, cuando la aplicación OO-Method ha sido generada, para ejecutar un servicio se realizan los siguientes movimientos:

1) La capa servidor lee desde la base de datos los valores que necesita para resolver la precondición
2) La capa servidor calcula el valor de la precondición. Si la precondición se cumple, la ejecución continúa en el paso 5. Si la condición no se cumple, la capa servidor genera un mensaje de error y se lo pasa a la capa cliente.
3) La capa cliente recibe el mensaje de error.
4) La capa cliente despliega el mensaje de error al usuario funcional humano y se termina la ejecución del servicio.
5) La capa servidor calcula el valor por defecto de los argumentos de entrada del servicio (dato valuados y objeto valuados) mediante valores constantes y funciones que se encuentran en esa capa, y entrega los valores calculados a la capa cliente.
6) La capa cliente recibe los valores para los argumentos de entrada que tengan valor por defecto del servicio desde la capa servidor.
7) La capa cliente despliega los valores de los argumentos de entrada del servicio al 'usuario funcional humano'.
8) El 'usuario funcional humano' observa los valores por defecto presentados en el servicio e ingresa a la capa cliente los valores que crea convenientes a los argumentos del servicio.
9) La capa cliente entrega los valores ingresados por el 'usuario funcional humano' a la capa servidor.
10) La capa servidor recibe los valores ingresados para los argumentos del servicio.
11) La capa servidor lee desde la base de datos la información que necesite para resolver la fórmula del servicio.

12) Si el servicio es un evento de creación, de destrucción o es un evento que tiene evaluaciones, la capa cliente escribe en la base de datos.
13) La capa servidor lee desde la base de datos los valores que necesita para resolver las restricciones de integridad.
14) La capa servidor calcula el valor de las restricciones de integridad. Si las restricciones se cumplen, la ejecución del servicio termina satisfactoriamente. Si la restricción no se cumple, la capa servidor genera un mensaje de error y se lo pasa a la capa cliente.
15) La capa cliente recibe el mensaje de error.
16) La capa cliente despliega el mensaje de error al usuario funcional humano y se termina la ejecución del servicio.

Para identificar los movimientos de datos asociados a un servicio desde el diagrama de objetos OO-Method, se han definido las siguientes reglas:

Regla 28: Identificar el movimiento de datos 1E y 1X para la capa cliente, y 1E para la capa servidor (representado por la clase que contiene al servicio) por el conjunto de **argumentos de entrada dato-valuados** de una UIS que participa en un proceso funcional.

Regla 29: Identificar el movimiento de datos 1E y 1X para la capa cliente, y 1E para la capa servidor por cada clase o vista legada diferente que corresponda a un **argumento de entrada objeto-valuado** de una UIS que participa en un proceso funcional.

Regla 30: Identificar el movimiento de datos 1R para la capa servidor por cada **clase** diferente que sea referenciada en la **condición** de la **fórmula de evaluación** de cada servicio asociado a una UIS que participa en un proceso funcional.

Regla 31: Identificar el movimiento de datos 1R para la capa servidor por cada **clase** diferente que sea referenciada en el **efecto** de la **fórmula de evaluación** de cada servicio asociado a una UIS que participa en un proceso funcional.

Regla 32: Identificar los movimientos de datos 1x y 1E para la capa servidor por cada **vista legada** diferente que sea referenciada en la **condición** de la **fórmula de evaluación** de cada servicio asociado a una UIS que participa en un proceso funcional.

Regla 33: Identificar los movimientos de datos 1X y 1E para la capa servidor por cada **vista legada** diferente que sea referenciada en el **efecto** de la **fórmula de evaluación** de cada servicio asociado a una UIS que participa en un proceso funcional.

Regla 34: Identificar el movimiento de datos 1R para la capa servidor por cada **clase** diferente que sea referenciada en la **fórmula** de cada **transacción, operación o servicio global** asociado a una UIS que participa en un proceso funcional.

Regla 35: Identificar los movimientos de datos 1X y 1E para la capa servidor por cada **vista legada** diferente que sea referenciada en la **fórmula** de cada **transacción, operación o servicio global** asociado a una UIS que participa en un proceso funcional.

Regla 36: Identificar el movimiento de datos 1R para la capa servidor por cada **clase** diferente que sea referenciada en las **fórmulas** de las **restricciones de integridad** de la clase que contiene a cada servicio asociado a una UIS que participa en un proceso funcional.

Regla 37: Identificar los movimientos de datos 1X y 1E para la capa servidor por cada **vista legada** diferente que sea referenciada en las **fórmulas** de las **restricciones de integridad** de la clase que contiene a cada servicio asociado a una UIS que participa en un proceso funcional.

Regla 38: Identificar el movimiento de datos 1X para la capa cliente por todos los **mensajes de error** de las **restricciones de integridad** de la clase que contiene a cada servicio asociado a una UIS que participa en un proceso funcional.

Regla 39: Identificar los movimientos de datos: 1E para la capa cliente, y 1X y 1R para la capa servidor, por cada **clase** diferente que sea referenciada en las **fórmulas** de los **mensajes de error** de las **restricciones de integridad** de la clase que contiene a cada servicio asociado a una UIS que participa en un proceso funcional.

Regla 40: Identificar los movimientos de datos: 1E para la capa cliente, y 1X y 1E para la capa servidor, por cada **vista legada** diferente que sea referenciada en las **fórmulas** de los **mensajes de error** de las **restricciones de integridad** de la clase que contiene a cada servicio asociado a una UIS que participa en un proceso funcional.

Regla 41: Identificar el movimiento de datos 1R para la capa servidor por cada **clase** diferente que sea referenciada en las **fórmulas** de las **precondiciones** de cada servicio asociado a una UIS que participa en un proceso funcional.

Regla 42: Identificar los movimientos de datos 1X y 1E para la capa servidor por cada **vista legada** diferente que sea referenciada en las **fórmulas** de las **precondiciones** de cada servicio asociado a una UIS que participa en un proceso funcional.

Regla 43: Identificar el movimiento de datos 1X para la capa cliente por todos los **mensajes de error** de las **precondiciones** a cada servicio asociado a una UIS que participa en un proceso funcional.

Regla 44: Identificar los movimientos de datos: 1E para la capa cliente, y 1X y 1R para la capa servidor, por cada **clase** diferente que sea referenciada en las **fórmulas** de los **mensajes de error** de las **precondiciones** de cada servicio asociado a una UIS que participa en un proceso funcional.

Regla 45: Identificar los movimientos de datos: 1E para la capa cliente, y 1X y 1E para la capa servidor, por cada **vista legada** diferente que sea referenciada en las **fórmulas** de los **mensajes de error** de las **precondiciones** de cada servicio asociado a una UIS que participa en un proceso funcional.

Regla 46: Identificar el movimiento de datos 1R para la capa servidor por cada **clase** diferente que sea referenciada en las **fórmulas** de las **reglas de dependencias** de los argumentos de cada servicio asociado a una UIS que participa en un proceso funcional.

Regla 47: Identificar el movimiento de datos 1R para la capa servidor por cada **clase** diferente que sea referenciada

en la **condición** de las **fórmulas** de las **reglas de dependencias** de los argumentos de cada servicio asociado a una UIS que participa en un proceso funcional.

Regla 48: Identificar los movimientos de datos 1X y 1E para la capa servidor por cada **vista legada** diferente que sea referenciada en las **fórmulas** de las **reglas de dependencias** de los argumentos de cada servicio asociado a una UIS que participa en un proceso funcional.

Regla 49: Identificar los movimientos de datos 1X y 1E para la capa servidor por cada **vista legada** diferente que sea referenciada en la **condición** de las **fórmulas** de las **reglas de dependencias** de los argumentos de cada servicio asociado a una UIS que participa en un proceso funcional.

Regla 50: Identificar el movimiento de datos 1E y 1X para la capa cliente, y 1X para la capa servidor (representado por la clase que contiene al servicio) por el conjunto de **argumentos de entrada dato-valuados** que tengan **valor por defecto**, asociados a una UIS que participa en un proceso funcional.

Regla 51: Identificar el movimiento de datos 1E y 1X para la capa cliente, y 1X para la capa servidor por cada **argumento de entrada objeto-valuado** que tenga **valor por defecto**, asociados a una UIS que participa en un proceso funcional.

Regla 52: Identificar el movimiento de datos 1W para la capa servidor (representado por la clase que contiene al servicio) por cada **evento** que sea de tipo **new**, **destroy** o que tenga **evaluaciones**, asociado a una UIS que participa en un proceso funcional.

Además, en los modelos dinámicos del sistema es posible especificar restricciones a la ejecución de los servicios. De esta manera, en el diagrama de transición de estados es posible especificar condiciones que deben cumplirse para que se pueda ejecutar un servicio, y en el diagrama de interacción de objetos es posible especificar los servicios se serán ejecutados automáticamente (triggers) si se cumple una condición en la clase que contiene al servicio que se está ejecutando.

Luego, cuando la aplicación OO-Method ha sido generada, para la ejecutar un servicio se realizan los siguientes movimientos desde la perspectiva dinámica del sistema:

1) La capa servidor revisa que se cumpla la condición de control asociada al servicio que se quiere ejecutar, mediante los valores de variables que se encuentran en esa capa y valores recuperados desde la base de datos.
2) Si la condición de control se cumple, la capa servidor ejecuta el servicio, realizando todos los movimientos de datos descritos desde la perspectiva del modelo de objetos.
3) Una vez ejecutado el servicio, la capa servidor revisa las condiciones de triggers asociadas a la clase que contiene el servicio ejecutado, mediante los valores de variables que se encuentran en esa capa y valores recuperados desde la base de datos.
4) Si alguna condición de trigger se cumple, la capa servidor dispara el servicio especificado para que se ejecute cuando se cumpla esa condición, realizando todos los movimientos de datos descritos desde la perspectiva del modelo de objetos.

Para identificar los movimientos de datos asociados a un servicio, desde los modelos dinámicos de OO-Method, se han definido las siguientes reglas:

Regla 53: Identificar el movimiento de datos 1R para la capa servidor por cada **clase** diferente que sea referenciada en la **fórmula** de la **condición de control** de cada servicio asociado a una UIS que participa en un proceso funcional.

Regla 54: Identificar los movimientos de datos 1X y 1E para la capa servidor por cada **vista legada** diferente que sea referenciada en la **fórmula** de la **condición de control** de cada servicio asociado a una UIS que participa en un proceso funcional.

Regla 55: Identificar el movimiento de datos 1R para la capa servidor por cada **clase** diferente que sea referenciada en la **condición** de cada **trigger** de la clase que contiene a cada servicio asociado a una UIS que participa en un proceso funcional.

Regla 56: Identificar los movimientos de datos 1X y 1E para la capa servidor por cada **vista legada** diferente que sea referenciada en la **condición** de cada **trigger** de la

clase que contiene a cada servicio asociado a una UIS que participa en un proceso funcional.

En último lugar, en el modelo de presentación se puede especificar qué interfaz gráfica se debe mostrar cuando se ha terminado la ejecución de un servicio. Para esto, se utiliza la navegación condicional, que permite especificar las unidades de interacción que deben desplegarse al usuario cuando se cumple una determinada condición, ya sea en caso de una ejecución con éxito o de fracaso. Además, es posible especificar el valor que tendrán los argumentos de entrada de las UISs mediante la inicialización de argumentos y especificar el grupo de registros que se deben mostrar en una UII, UIP o UIMD mediante el filtrado navegacional.

Después, cuando la aplicación OO-Method ha sido generada, para ejecutar un servicio se realizan los siguientes movimientos desde la perspectiva de presentación del sistema:

1) La capa servidor revisa las condiciones de éxito o fracaso de la navegación condicional del servicio ejecutado según sea el caso. Para ver si se cumplen las condiciones de la navegación condicional, la capa servidor utiliza valores de variables que se encuentran en esa capa.
2) Si la condición de la navegación condicional se cumple, la capa servidor ejecuta la unidad de interacción asociada a esa condición.
3) Si la unidad de interacción es una UIS, la capa servidor calcula el valor de los argumentos inicializados utilizando valores de variables que se encuentran en esa capa y valores recuperados desde la base de datos.
4) Si la unidad de interacción es una UII, UIP o UIMD, la capa servidor calcula los registros que serán desplegados mediante el filtrado navegacional de los registros, para lo cual utiliza valores de variables que se encuentran en esa capa y valores recuperados desde la base de datos.

Para identificar los movimientos de datos asociados a un servicio, desde el modelo de presentación de OO-Method, se han definido las siguientes reglas:

Regla 57: Identificar el movimiento de datos 1R para la capa servidor por cada **clase** diferente que sea referenciada en la **fórmula de inicialización de argumentos** de la UIS asociada a una navegación condicional de cada servicio asociado a una UIS que participa en un proceso funcional.

Regla 58: Identificar el movimiento de datos 1R para la capa servidor por cada **clase** diferente que sea referenciada en la **condición** de la **fórmula de inicialización de argumentos** de la UIS asociada a una navegación condicional de cada servicio asociado a una UIS que participa en un proceso funcional.

Regla 59: Identificar los movimientos de datos 1X y 1E para la capa servidor por cada **vista legada** diferente que sea referenciada en la **fórmula de inicialización de argumentos** de la UIS asociada a una navegación condicional de cada servicio asociado a una UIS que participa en un proceso funcional.

Regla 60: Identificar los movimientos de datos 1X y 1E para la capa servidor por cada **vista legada** diferente que sea referenciada en la **condición** de la **fórmula de inicialización de argumentos** de la UIS asociada a una navegación condicional de cada servicio asociado a una UIS que participa en un proceso funcional.

Regla 61: Identificar el movimiento de datos 1R para la capa servidor por cada **clase** diferente que sea referenciada en la **fórmula de filtrado navegacional** de la UII, UIP o UIMD asociada a una navegación condicional de cada servicio asociado a una UIS que participa en un proceso funcional.

Regla 62: Identificar los movimientos de datos 1X y 1E para la capa servidor por cada **vista legada** diferente que sea referenciada en la **fórmula de filtrado navegacional** de la UII, UIP o UIMD asociada a una navegación condicional de cada servicio asociado a una UIS que participa en un proceso funcional.

Todas las reglas definidas para identificar los movimientos de datos que pueden ocurrir en las aplicaciones OO-Method están estructuradas con un concepto de COSMIC, una primitiva conceptual OO-Method, y la cardinalidad que permite relacionar esos conceptos. En resumen, se han definido 50 reglas para identificar los movimientos de datos que pueden ocurrir en las aplicaciones OO-Method, las cuales se han presentado anteriormente. Utilizando estas reglas sobre el modelo conceptual OO-Method se pueden identificar todos los movimientos de datos de los procesos funcionales implementados para el correcto funcionamiento de las aplicaciones OO-Method generadas a partir de ese modelo conceptual.

4.4 Definición de Reglas de Asignación Numérica

Las reglas de medición son reglas que asignan un valor numérico a los movimientos de datos que ocurren entre los usuarios funcionales y los procesos funcionales de las aplicaciones OO-Method. Estos valores numéricos representan el tamaño funcional de las aplicaciones generadas automáticamente en entornos MDA.

Como se puede observar en la Figura 7, en este paso del diseño del procedimiento de medición OOmCFP se concluye la fase de medición del procedimiento mediante la aplicación de la función de medición y la agregación de los resultados.

4.4.1 Función de Medición

De acuerdo al método de medición de tamaño funcional COSMIC, la función de medición es una función matemática que asigna una unidad de tamaño funcional referida como 1 CFP (Cosmic Function Point) a cada movimiento de datos.

En el procedimiento de medición OOmCFP, la función de medición es idéntica a la definida en el método medición COSMIC. Es decir, asigna 1 CFP a cada movimiento de datos.

4.4.2 Agregación de Resultados

Para medir el tamaño funcional de un proceso funcional de una aplicación OO-Method, todos los movimientos de datos que ocurren en ese proceso funcional son agregados. Para esto se ha definido la siguiente regla de medición:

Regla 63: Contar **1CFP** por cada **movimiento de datos** de la **capa cliente** de un proceso funcional, y agregar los valores para obtener el tamaño funcional de ese **proceso funcional** en la capa cliente.

Regla 64: Contar **1CFP** por cada **movimiento de datos** de la **capa servidor** de un proceso funcional, y agregar los valores para obtener el tamaño funcional de ese **proceso funcional** en la capa servidor.

Una vez que todos los procesos funcionales han sido medidos utilizando las reglas 63 y 64, se puede obtener el tamaño funcional de las capas de software agregando el tamaño funcional de los procesos que ocurren en cada capa. Para obtener el tamaño

funcional de las capas de software de las aplicaciones OO-Method se han definido las siguientes reglas:

Regla 65: Agregar los **tamaños funcionales** obtenidos para cada **proceso funcional** en la capa cliente para obtener el tamaño funcional de la **capa cliente**.

Regla 66: Agregar los **tamaños funcionales** obtenidos para cada **proceso funcional** en la capa servidor para obtener el tamaño funcional de la **capa servidor**.

En último lugar, para calcular el tamaño funcional de una aplicación OO-Method es necesario agregar el tamaño funcional de cada capa de software que compone dicha aplicación. Para esto se ha definido la siguiente regla:

Regla 67: Agregar los **tamaños funcionales** obtenidos para cada **capa** de la **aplicación OO-Method** para obtener el tamaño funcional de dicha aplicación.

Finalmente, con la ayuda de las reglas presentadas anteriormente es posible cuantificar el tamaño funcional de las aplicaciones de software que son generadas a partir de modelos conceptuales OO-Method en un entorno MDA.

4.5 Conclusiones

En este capítulo se ha presentado el diseño del procedimiento de medición OOmCFP, el cual permite medir el tamaño funcional de aplicaciones generadas en entornos MDA a partir de sus modelos conceptuales. El procedimiento de medición OOmCFP ha sido definido siguiendo rigurosamente el modelo de procesos para la definición de métodos de medición propuesto por Jacquet y Abran [Jacquet97] [Abran99].

El procedimiento OOmCFP ha sido diseñado de acuerdo al método estándar de medición COSMIC, debido a que este método permite medir correctamente la funcionalidad de aplicaciones generadas en varias capas con varias piezas de software, como es el caso de la mayoría de las aplicaciones generadas en entornos MDA. Además, este procedimiento ha sido validado por expertos del método COSMIC de acuerdo a su conformidad con el método de medición COSMIC versión 3.0.

En este capítulo se han presentado 62 reglas que permiten identificar los conceptos de COSMIC en las primitivas conceptuales de los modelos OO-Method. Además, se han presentado 4 reglas que ayudan a identificar correctamente los elementos que pueden estar contenidos en cada proceso funcional. Finalmente, se han presentado 5 reglas que permiten calcular el tamaño funcional de los procesos funcionales de la aplicación, de cada capa de la aplicación y de la aplicación completa. Aplicando estas reglas, el procedimiento OOmCFP puede obtener de manera exacta el tamaño funcional de aplicaciones generadas a partir de modelos conceptuales.

A pesar de que OOmCFP se ha diseñado para ser utilizado específicamente en el contexto de OO-Method, varios elementos conceptuales del modelo conceptual de OO-Method pueden encontrarse en otros métodos orientados a objetos (por ejemplo las clases, los atributos, los servicios, los argumentos, etc.). Por esta razón, este procedimiento de medición puede ser generalizado para ser aplicado a otros métodos de desarrollo orientados a objeto, que por ejemplo utilicen UML para realizar los modelos conceptuales.

Capítulo 5
Aplicación de OOmCFP

Para aplicar el procedimiento de medición OOmCFP (OO-Method COSMIC Function Points) se han seguido los pasos definidos en el modelo de procesos para los procedimientos de medición presentado en la sección 3.3 para el paso de aplicación del procedimiento de medición definido. De esta manera, en este capítulo se presenta la aplicación de OOmCFP y un caso de estudio que ejemplifica esta aplicación.

5.1 Aplicación de un Procedimiento de Medición de Tamaño Funcional

Según el modelo de procesos para los procedimientos de medición de software propuesto por Jacquet y Abran [16] [4], la aplicación de un procedimiento de medición consiste en los siguientes pasos: recolección de la documentación del software, construcción del modelo, y asignación de reglas de medición.

5.1.1 Recolección de Documentación del Software

La documentación que debe ser recolectada para aplicar OOmCFP a una aplicación OO-Method son el modelo de objetos, el modelo dinámico, el modelo funcional, y el modelo de presentación que especifican dicha aplicación. Dado que el modelo conceptual OO-Method corresponde a la conjunción de los modelos antes mencionados, la documentación que debe ser recolectada para aplicar OOmCFP a una aplicación OO-Method es dicho modelo. De esta manera, en el modelo conceptual OO-Method (documentación) se tienen todos los detalles necesarios para generar completamente la aplicación que funciona correctamente sin incluir posteriores cambios manuales.

5.1.2 Construcción del Modelo de Software

La construcción del modelo OOmCFP debe ser realizada mediante las correspondencias entre COSMIC y OO-Method detalladas en la sección 4.3 del diseño de OOmCFP. Para evitar confusión en el orden en que deben ser aplicadas las reglas

definidas en el diseño de OOmCFP se ha definido un proceso que consta de los siguientes pasos:

1. Especificar el propósito de la medición
2. Especificar el alcance de la medición
3. Identificar los usuarios funcionales y las fronteras
4. Identificar los procesos funcionales
5. Eliminar duplicidad de procesos funcionales
6. Identificar los grupos de datos
7. Identificar los atributos (opcional)
8. Identificar los movimientos de datos

En la actividad *especificar el propósito de la medición* es necesario especificar por qué se realiza la medición y para qué será utilizado el resultado de la medición.

En la actividad *especificar el alcance de la medición* se determina el alcance dependiendo del propósito definido. Como se explicó en la sección 4.1.2 del capítulo de diseño de OOmCFP, el alcance es el modelo conceptual OO-Method, a partir del cual se puede medir la aplicación OO-Method completa, la capa cliente de la aplicación OO-Method, o la capa servidor de la aplicación OO-Method.

En la actividad *identificar los usuarios funcionales y las fronteras*, se identifican los usuarios funcionales de acuerdo a las capas de software que serán incluidas en la medición. Utilizando las *reglas 1, 2,* y *3* especificadas en la sección 4.3.1 se identifican los usuarios funcionales y utilizando la *regla 4* se identifican las fronteras de las piezas de software que serán medidas.

En la actividad *identificar los procesos funcionales* se identifican los procesos funcionales aplicando la *regla 5* y *8* definidas en la sección del diseño del procedimiento de medición OOmCFP. Además, para la correcta identificación de los elementos que pueden estar contenidos en los procesos funcionales se aplican las *reglas 5.1, 5.2, 5.3* y *5.4*.

En la actividad *eliminar duplicidad en los procesos funcionales* se eliminan los procesos que hayan sido identificados más de una vez utilizando las *reglas 6* y *7*.

En la actividad *identificar los grupos de datos* se identifican las clases del modelo de objetos de OO-Method que participan en un proceso funcional. Es decir, se identifican los grupos de datos utilizando las *reglas 9, 10* y *11*.

La actividad *identificar los atributos de datos* es una actividad opcional que permite identificar los atributos de los grupos de datos. Para esto se utiliza la *regla 12*.

La actividad *identificar los movimientos de datos* es la actividad principal en la aplicación del procedimiento de medición OOmCFP. Para identificar los movimientos de datos se deben aplicar las *reglas 13 al 62* definidas en el diseño de OOmCFP.

5.1.3 Asignación de Reglas de Medición

En este paso se deben asignar al modelo de software construido para OOmCFP las reglas de medición detalladas en la sección 4.4 del diseño de OOmCFP. Para evitar confusión en el orden en que deben ser aplicadas las reglas de asignación numérica definidas en el diseño de OOmCFP se ha definido un proceso que consta de los siguientes pasos:

1. Calcular el tamaño funcional de cada proceso funcional
2. Calcular el tamaño funcional de cada capa de software
3. Calcular el tamaño funcional de la aplicación

La actividad *calcular el tamaño funcional de cada proceso funcional* se realiza aplicando las *reglas 63* y *64* definidas en la sección 4.4.2 del diseño de OOmCFP.

La actividad *calcular el tamaño funcional de cada capa de software* permite obtener el tamaño funcional de la capa cliente y de la capa servidor de una aplicación OO-Method. Para esto se aplican las *reglas 65* y *66*.

Finalmente, la actividad *calcular el tamaño funcional de la aplicación* permite obtener el tamaño funcional de la aplicación OO-Method utilizando la *regla 67*.

Los pasos de la asignación de reglas de medición deben realizarse inmediatamente después de los pasos definidos en la construcción del modelo de software. De esta manera, el proceso completo de aplicación del procedimiento OOmCFP queda compuesto de once pasos. La Figura 12 ilustra el proceso de aplicación del procedimiento OOmCFP.

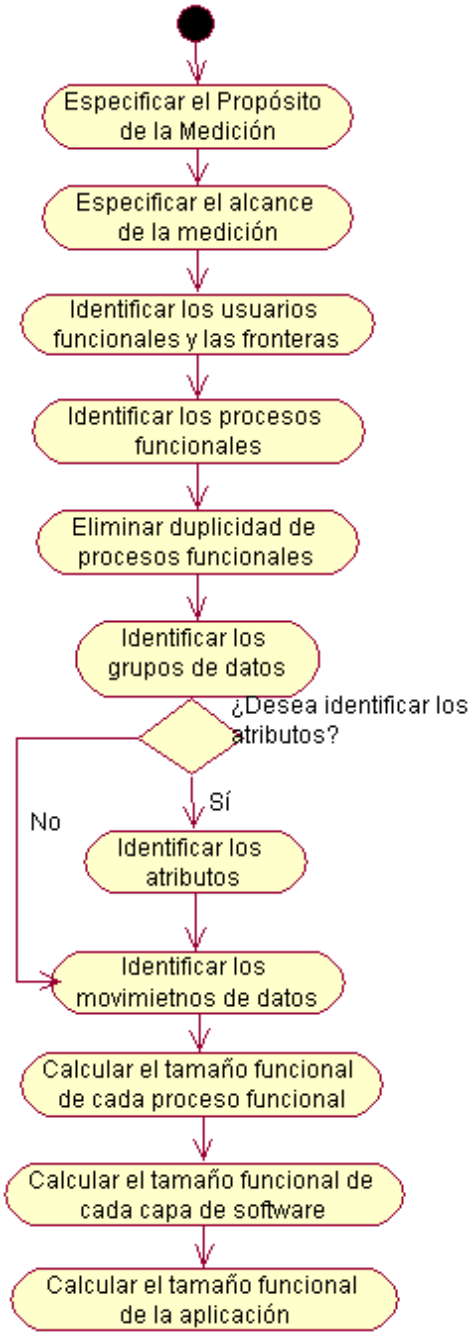


Figura 12. Proceso de aplicación de OOmCFP.

5.2 Aplicación de OOmCFP

Utilizando el proceso de aplicación de OOmCFP definido en la sección 5.1, el procedimiento de medición OOmCFP fue aplicado a seis modelos conceptuales OO-Method que representan la funcionalidad de las siguientes aplicaciones: Facturación, Alquiler de coches, Camping, Editorial, Agencia fotográfica, y Nota de gastos.

Las aplicaciones generadas a partir de los modelos conceptuales de Facturación, Alquiler de coches y Camping fueron completamente medidas utilizando el procedimiento de medición OOmCFP. Los tamaños funcionales obtenidos fueron los siguientes: para la aplicación de Facturación se obtuvo un tamaño funcional de 108 CFP, para la aplicación de Alquiler de coches se obtuvo un tamaño funcional de 96 CFP, y para la aplicación de Camping se obtuvo un tamaño funcional de 456 CFP.

Las aplicaciones generadas a partir de los modelos conceptuales de Editorial, Agencia fotográfica, y Nota de gastos fueron utilizados para medir algunos procesos funcionales. Por ejemplo: para la aplicación de Editorial se midieron el proceso funcional de administración de usuarios, obteniendo un tamaño funcional de 32 CFP, y el proceso funcional de publicación de un libro, obteniendo un tamaño funcional de 9 CFP; para la aplicación de Agencia fotográfica se midieron el proceso funcional de asignación de solicitud de fotógrafo, obteniendo un tamaño funcional de 20 CFP, y el proceso funcional de creación de un reportaje fotográfico, obteniendo un tamaño funcional de 18 CFP; y finalmente, para la aplicación de Nota de gastos se midieron el proceso funcional de administración de gastos, obteniendo un tamaño funcional de 72 CFP, y el proceso funcional de pago de la nota de gastos, obteniendo un tamaño funcional de 48 CFP.

Las siguientes secciones ejemplifican la aplicación del procedimiento de medición de tamaño funcional OOmCFP a la aplicación de Alquiler de coches.

5.2.1 Recolección de Documentación del Software

La documentación que debe ser recolectada para aplicar OOmCFP a la aplicación de alquiler de coches OO-Method es el modelo conceptual OO-Method para dicha aplicación.

La descripción del caso de estudio del sistema de alquiler de coches es la siguiente: "Se desea diseñar un sistema para administrar la información los alquileres de una empresa dedicada al alquiler de automóviles, teniendo en cuenta que:

- Un determinado cliente puede tener varios alquileres.
- De cada cliente se desea almacenar su DNI, nombre, dirección y teléfono. Además, los clientes se diferencian por un código único.
- Un alquiler lo realiza un único cliente y está asociado a un vehículo.

- Es importante registrar la fecha de inicio y final del alquiler, el precio del alquiler, y los litros de gasolina en el depósito en el momento de realizar el alquiler.
- El precio del alquiler es calculado con un monto fijado por la empresa, más una tasa que depende del tamaño del vehículo.
- Cada vehículo tiene un grupo que representa el tamaño del vehículo, y que es utilizado para calcular el precio del alquiler.
- De cada vehiculo se requiere la matrícula, el modelo, el color y la marca.
- Los vehículos disponibles para alquilar pueden ser turismos o minibús. De los turismos se necesita conocer el número de puertas y si consume diesel o gasolina. Todos los minibuses consumen diesel, y de ellos sólo se necesita conocer el número de asientos.
- Cada alquiler se realiza en una determinada oficina de la empresa de alquiler de coches".

La Figura 13 muestra el modelo de objetos del modelo conceptual OO-Method que especifica el sistema de alquiler de coches según la descripción presentada anteriormente. Además, en este modelo se ha creado una clase agente denominada *Administrador* para ejecutar los servicios del sistema (recuerde que las clases agentes pueden ejecutar los servicios que tienen asociados con una línea punteada).

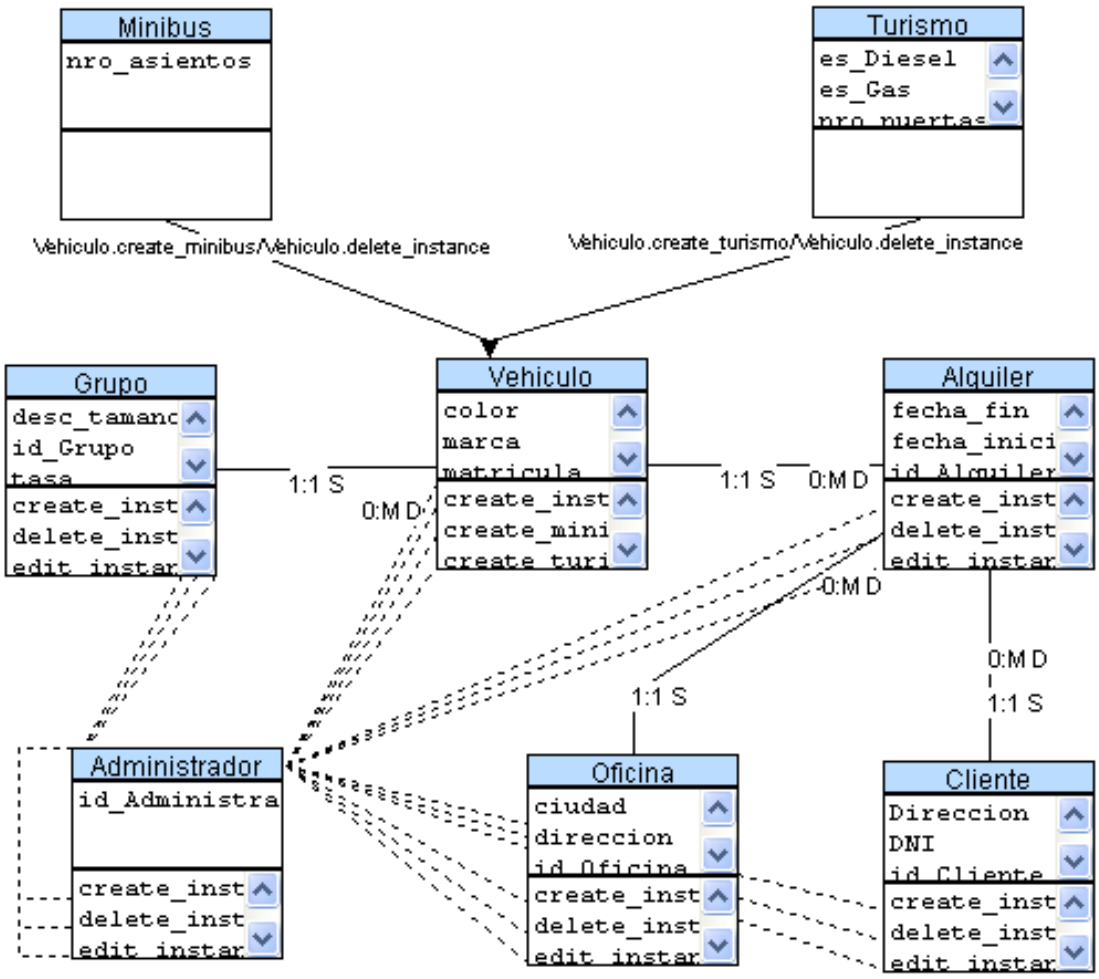


Figura 13. Modelo de objetos del sistema de Alquiler de coches.

El modelo funcional del modelo conceptual OO-Method para el sistema de Alquiler de coches captura la semántica asociada a los cambios de estado dada la ejecución de eventos. Por ejemplo, la siguiente figura muestra una parte del modelo funcional del sistema Alquiler de coches, en donde se muestra que el evento *edit_instance* de la clase *Vehiculo* asigna el valor p_color al atributo color.

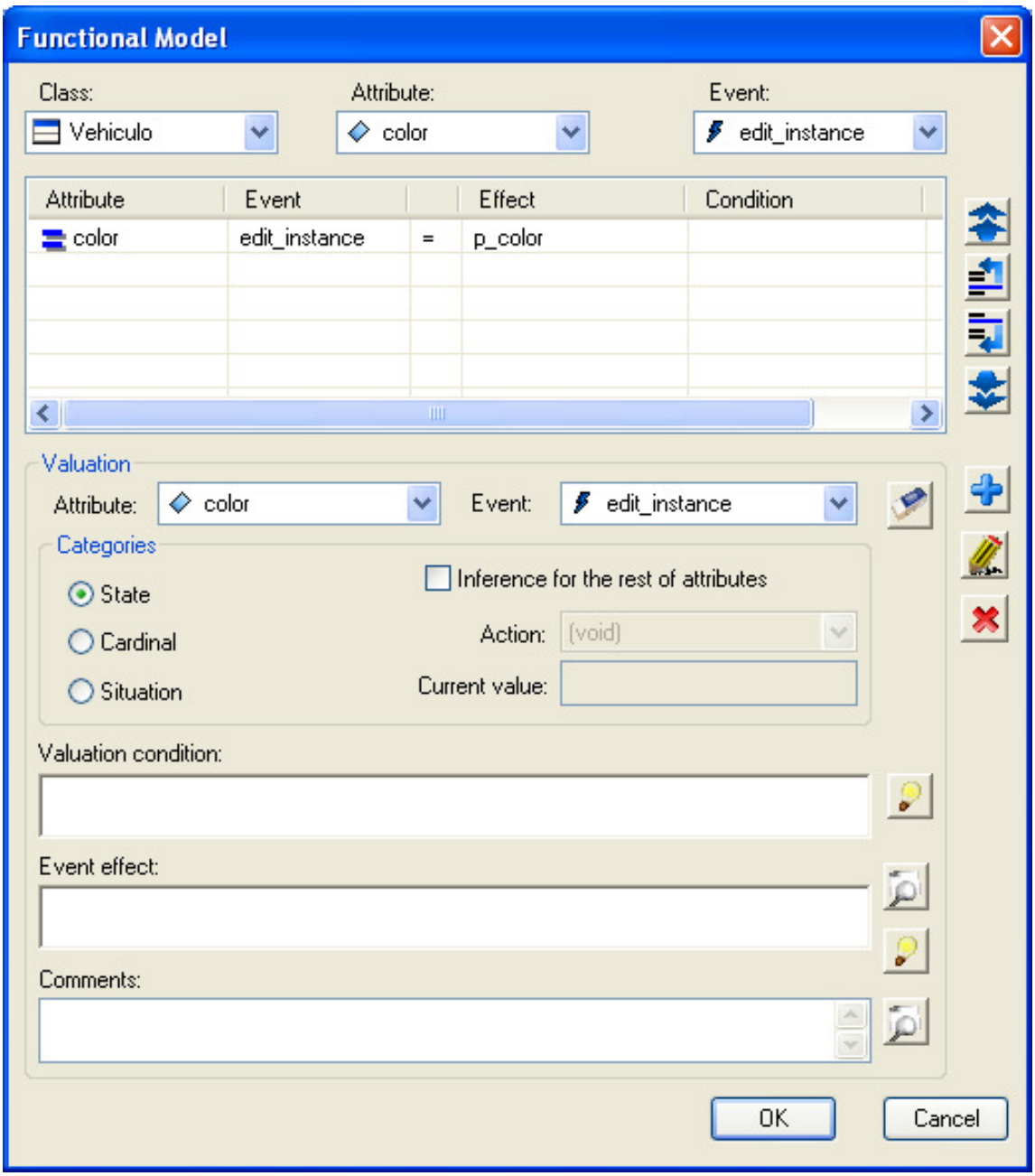


Figura 14. Modelo funcional del sistema de Alquiler de coches.

El modelo dinámico del modelo conceptual OO-Method para el sistema de Alquiler de coches representa las vidas válidas de los objetos de las clases mostradas en la Figura 13. El modelo dinámico especifica los posibles estados que pueden alcanzar los objetos de una clase y los servicios que cambian el estado de los objetos. Este modelo es creado automáticamente por la herramienta Olivanova Modeler [CARE] a partir del modelo de objetos. La Figura 15 muestra el modelo dinámico de la clase *Cliente* del sistema de Alquiler de coches.

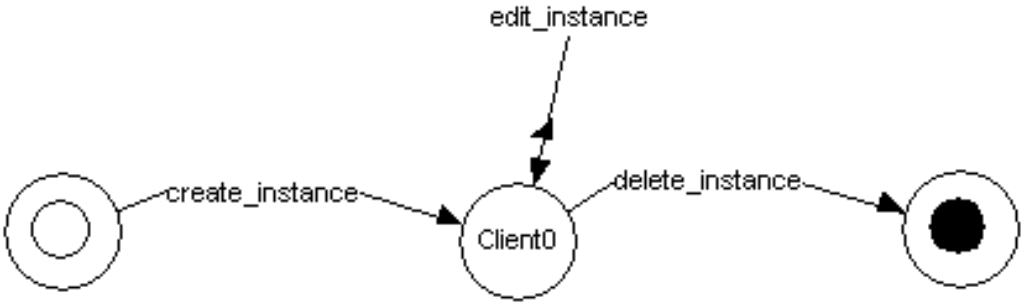


Figura 15. Modelo dinámico del sistema de Alquiler de coches.

La información del sistema de Alquiler de coches es presentada al usuario de la aplicación mediante un menú de opciones que son modeladas a través del árbol jerárquico de acciones (HAT) en el modelo de presentación. Para el sistema de Alquiler de coches, las opciones de menú son tres grupos de registros denominados Unidades de Interacción de Población (UIP), los cuales muestran información de conjuntos de instancias de las clases del modelo de objetos. Para cada UIP definida en el menú del sistema de Alquiler de coches, también se especifican los atributos que serán mostrados al usuario, las acciones que podrá realizar, las navegaciones que podrá llevar a cabo y los filtros que tiene disponibles para seleccionar la información desplegada en la unidad de interacción. La Figura 16 muestra el HAT del sistema de alquiler de coches.

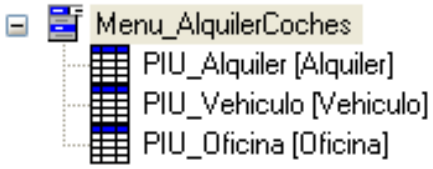


Figura 16. Menú especificado en el modelo de presentación del sistema de Alquiler de coches.

Cuando todos los modelos del modelo conceptual de OO-Method han sido especificados, es posible verificar el modelo conceptual y generar la aplicación final a partir de ese modelo mediante la utilización del compilador de modelos OO-Method [CARE]. De esta manera, el modelo conceptual del sistema de Alquiler de coches es la documentación necesaria para aplicar el procedimiento de medición OOmCFP.

5.2.2 Construcción del Modelo de Software

La construcción del modelo de software para OOmCFP ha sido realizada siguiendo el proceso de aplicación de OOmCFP (vea la

Figura 12). En esta sección se presentan los elementos que contribuyen al tamaño funcional del sistema de Alquiler de coches.

1) Especificar el Propósito de la Medición

El propósito de aplicar OOmCFP al sistema de Alquiler de coches es medir el tamaño funcional del sistema de Alquiler de coches generado por el compilador de modelos OO-Method desarrollado por la empresa CARE Technologies [CARE]. En términos de la plantilla GQM [Basili88], el propósito es el siguiente:

Aplicar el procedimiento de medición OOmCFP
con el propósito de medir el sistema de Alquiler de coches
con respecto a su tamaño funcional
desde el punto de vista del desarrollador
en el contexto del Modelo Conceptual OO-Method.

2) Especificar el Alcance de la Medición

El alcance de la medición es el modelo conceptual OO-Method del sistema de Alquiler de coches. El nivel de granularidad es bajo, ya que todos los detalles del modelo conceptual son necesarios para la correcta generación del sistema de Alquiler de coches.

Las piezas de software del sistema de Alquiler de coches son el componente Cliente, el componente Servidor, y el componente Base de Datos. Cada componente de una aplicación OO-Method es construida para un entorno de software específico, por ejemplo, el componente cliente puede ser construido para C#, ASP o JSP, el componente servidor puede ser construido para C#, EJB, o VB, y el componente base de datos puede ser construido para SQL, Oracle, o DB2.

Con respecto a las capas de software, dado que el sistema de Alquiler de coches está generado en una arquitectura de tres capas que se relacionan entre sí mediante una jerarquía, cada capa de la arquitectura es considerada una capa de software por el procedimiento de medición OOmCFP.

3) Identificar los Usuarios Funcionales y la Frontera

Los usuarios funcionales del sistema de Alquiler de coches son los usuarios de los procesos funcionales especificados en cada pieza de software del sistema.

La clase *Administrador* de la Figura 13 es un usuario funcional debido a que las instancias de esa clase pueden ejecutar los servicios del sistema de Alquiler de coches.

Además, el componente cliente del sistema de Alquiler de coches es un usuario funcional ya que es un usuario del componente servidor del sistema. Este usuario funcional será denominado 'usuario funcional cliente'.

Asimismo, el componente servidor del sistema de Alquiler de coches es un usuario funcional del componente cliente y del componente de base de datos del sistema. Este usuario funcional será denominado 'usuario funcional servidor'.

Entre cada usuario funcional del sistema de Alquiler de coches y sus componentes existe una frontera. Esto es, existe una frontera entre el usuario administrador y el componente cliente del sistema de Alquiler de coches, existe una frontera entre el 'usuario funcional cliente' y el componente servidor del sistema de Alquiler de coches, y existe una frontera entre el 'usuario funcional servidor' y los componentes cliente y base de datos del sistema de Alquiler de coches.

La siguiente figura presenta los usuarios y fronteras del sistema de Alquiler de coches obtenido de la instanciación de la Figura 10:

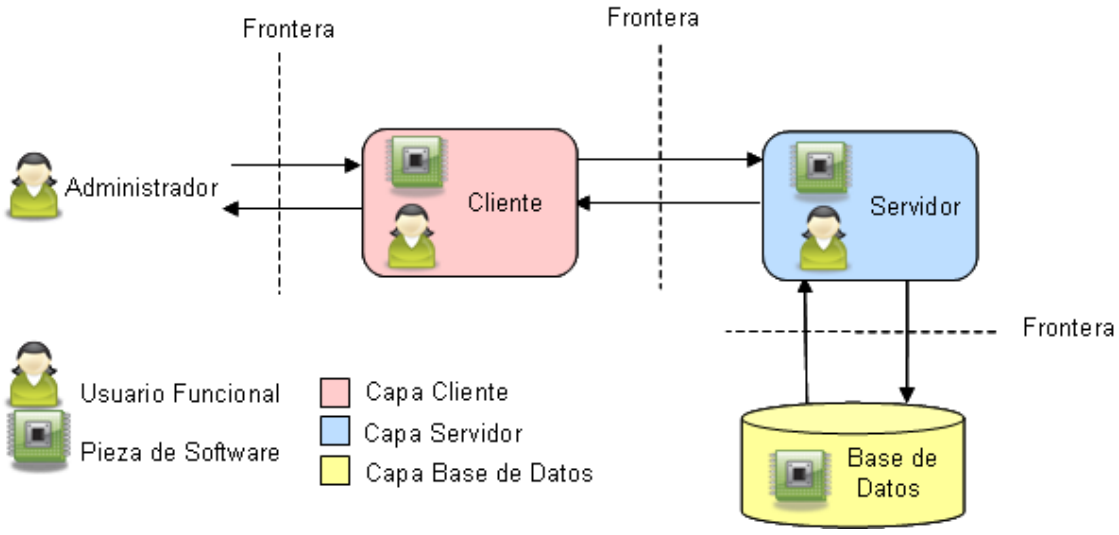


Figura 17. Usuarios funcionales, fronteras, piezas y capas de software del sistema de Alquiler de coches.

4) Identificar los Procesos Funcionales

El usuario administrador realiza *eventos disparadores* que ocurren en el mundo real, por ejemplo registrar los vehículos pertenecientes a la empresa, registrar las oficinas y registrar los alquileres de coches. Para realizar cada una de estas tareas en el sistema, el administrador comienza procesos funcionales.

Para identificar los procesos funcionales, se ha utilizado la *regla 5* definida en la sección 4.3, que especifica que los hijos directos del árbol jerárquico de acciones (HAT) del modelo de presentación que sean UIP, UIS o UIMD son procesos funcionales del sistema. De esta manera, los procesos funcionales del sistema de Alquiler de coches son: PIU_Alquiler, PIU_Vehiculo, y PIU_Oficina (vea la Figura 16).

El 'usuario funcional cliente' del sistema de Alquiler de coches comienza procesos funcionales de la capa servidor del sistema, los cuales son acciones que realiza la capa servidor en respuesta a las peticiones del 'usuario funcional cliente'. Por ejemplo, estas acciones serían: leer valores desde la base de datos para mostrárselos al usuario administrador en una unidad de interacción de población (UIP), la ejecución de una fórmula de un servicio que es ejecutado mediante las acciones de una unidad de interacción de población (UIP), la evaluación de las precondiciones de un servicio que es ejecutado en la fórmula de valor por defecto de una variable de filtro de una unidad de interacción de población (UIP), etc. Por simplicidad, el conjunto de acciones que la capa servidora realiza en respuesta a las peticiones del 'usuario funcional cliente' recibirán el mismo nombre de los procesos funcionales que ocurren en la capa cliente y que necesitan su ejecución. De esta manera, aplicando la *regla 8*, los procesos funcionales de la capa servidor del sistema de Alquiler de coches son: PIU_Alquiler, PIU_Vehiculo, y PIU_Oficina.

Para identificar los elementos contenidos en cada proceso funcional se utilizan las *reglas 5.1* y *5.2*. Para el proceso funcional que permite registrar los alquileres (PIU_Alquiler), aplicando la *regla 5.1* se ha identificado el conjunto de visualización DS_Alquiler, y las acciones A_Alquiler. Nuevamente aplicando la *regla 5.1* se han identificado las unidades de interacción de servicio (UIS) que pueden ser ejecutadas a partir de las acciones A_Alquiler, es decir, las unidades de interacción que transitivamente componen a PIU_Alquiler. Estas unidades de interacción son: SIU_create_instance, SIU_delete_instance, y SIU_edit_instance. Luego, aplicando la *regla 5.2* se identifican las navegaciones condicionales y las unidades de interacción que

transitivamente componen cada UIS. Dado que las unidades de interacción de servicio SIU_create_instance, SIU_delete_instance, y SIU_edit_instance no presentan navegación condicional ni otras unidades de interacción asociadas, se termina con la identificación de elementos contenidos en el proceso funcional PIU_Alquiler (vea la Tabla 8).

Tabla 8. Elementos contenidos en el proceso funcional PIU_Alquiler.

Proceso funcional	**Elementos contenidos**	
PIU_Alquiler	DS_Alquiler	
	A_Alquiler	SIU_create_instance
		SIU_delete_instance
		SIU_edit_instance

Para el proceso funcional que permite registrar los vehículos que posee la empresa (PIU_Vehiculo), aplicando la *regla 5.1* se ha identificado el conjunto de visualización DS_Vehiculo, las acciones A_Vehiculo, y las navegaciones N_Vehiculo. Nuevamente aplicando la *regla 5.1* se han identificado las unidades de interacción de servicio (UIS) que pueden ser ejecutadas a partir de las acciones A_Vehiculo: SIU_create_instance, SIU_delete_instance, y SIU_edit_instance. También, aplicando la regla 5.1 se han identificado las unidades de interacción que estan contenidas en la navegación N_Vehiculo: PIU_Minibus y PIU_Turismo. Aplicando la regla 5.2, no se han identificado navegaciones condicionales ni otras unidades de interacción asociadas a las unidades de interacción de servicio SIU_create_instance, SIU_delete_instance, y SIU_edit_instance, por lo que se termina con la identificación de elementos contenidos en las acciones A_Vehiculo.

En cuanto a los elementos contenidos en las navegaciones N_Vehiculo, para identificar los elementos contenidos en la unidad de interacción de población PIU_Minibus se aplica la *regla 5.1*. Así, para PIU_Minibus se ha identificado el conjunto de visualización DS_Minibus y las acciones A_Minubus. Iterando en la aplicación de la regla 5.1, se identifica la unidad de interacción de servicio SIU_create_minibus contenida en las acciones A_Minibus. Posteriormente, aplicando la regla 5.2, no se identifican elementos contenidos en SIU_create_minibus, por lo que se termina con la identificación de elementos contenidos en la unidad de interacción PIU_Minibus. Por otro lado, para identificar los elementos contenidos en PIU_Turismo también se aplican las *reglas 5.1* y *5.2*. De esta manera, se identifica el conjunto de visualización DS_Turismo, las acciones A_Turismo y la unidad de interacción de servicio SIU_create_turismo, y se termina con la identificación de elementos contenidos en PIU_Turismo. De esta manera, se

concluye con la identificación de elementos contenidos en el proceso funcional PIU_Vehiculo (vea la Tabla 9).

Tabla 9. Elementos contenidos en el proceso funcional PIU_Vehiculo.

Proceso funcional	**Elementos contenidos**			
PIU_Vehiculo	DS_Vehiculo			
	A_Vehiculo	SIU_create_instance		
		SIU_delete_instance		
		SIU_edit_instance		
	N_Vehiculo	PIU_Minibus	DS_Minubus	
			A_Minibus	SIU_create_minibus
		PIU_Turismo	DS_Turismo	
			A_Turismo	SIU_create_turismo

Para el proceso funcional que permite registrar las oficinas de la empresa (PIU_Oficina), aplicando la *regla 5.1* se ha identificado el conjunto de visualización DS_Oficina, y las acciones A_Oficina. Luego, aplicando nuevamente la *regla 5.1* se han identificado las unidades de interacción de servicio SIU_create_instance, SIU_edit_instance y SIU_delete_instance. A continuación, aplicando la *regla 5.2* no se han identificado elementos contenidos en las unidades de interacción de servicio, por lo que concluye la identificación de elementos contenidos en el proceso funcional PIU_Oficina (vea la Tabla 10).

Tabla 10. Elementos contenidos en el proceso funcional PIU_Oficina.

Proceso funcional	**Elementos contenidos**	
PIU_Oficina	DS_Oficina	
	A_Oficina	SIU_create_instance
		SIU_delete_instance
		SIU_edit_instance

5) Eliminar Duplicidad de Procesos Funcionales

Para evitar la duplicidad en el conteo del tamaño funcional se aplican las *reglas 6* y *7* definidas en la sección 4.3 del capítulo de diseño de OOmCFP.

La *regla 6* permite eliminar los procesos funcionales que estén contenidos en otro proceso funcional. Dado que en el sistema de Alquiler de coches no se presenta esta situación, no es necesario eliminar procesos funcionales contenidos dentro de otro proceso funcional.

La *regla 7* permite eliminar unidades de interacción que han sido identificadas varias veces dentro de un mismo proceso funcional. Dado que en el sistema de Alquiler de coches no se presenta esta situación, no es necesario eliminar unidades de interacción duplicadas en un proceso funcional.

6) Identificar los Grupos de Datos

Los grupos de datos del sistema de Alquiler de coches han sido identificados aplicando las *reglas 9, 10* y *11* definidas en la sección 4.3 del capítulo de diseño de OOmCFP.

La *regla 9* especifica que un grupo de datos de un proceso funcional es cada una de las clases que participan en ese proceso funcional y que no pertenecen a una jerarquía de herencia. Con esta regla se han identificado los grupos de datos: Cliente, Alquiler, Grupo, Oficina y Administrador.

La *regla 10* especifica que se debe reconocer un grupo de datos por la clase padre de una jerarquía de herencia que participa en un proceso funcional. Con esta regla se identifica el grupo de datos Vehiculo.

Finalmente, aplicando la *regla 11* que especifica que se debe reconocer un grupo de datos por cada clase hija en una jerarquía de herencia que tenga atributos diferentes a los del padre y que participa en un proceso funcional, se han identificado los grupos de datos: Minibus y Turismo.

7) Identificar los Atributos

Para identificar los atributos de los grupos de datos se ha aplicado la *regla 12* del diseño de OOmCFP, que determina que los atributos de las clases identificadas como grupo de datos son los atributos de los grupos de datos correspondientes.

A modo de resumen, la Tabla 11 muestra los procesos funcionales, los grupos de datos que participan en los procesos funcionales, y los atributos de esos grupos de datos.

Tabla 11. Procesos funcionales, grupos de datos, y atributos de los grupos de datos del sistema de Alquiler de coches.

Proceso funcional	Grupos de datos	Atributos
PIU_Alquiler	Alquiler	id_Alquiler, fecha_inicio, fecha_fin, litros_gas, tasa_fija, precio
	Cliente	id_Client, DNI, Nombre, Direccion, Telefono
	Vehiculo	matricula, modelo, color, marca
	Grupo	id_Grupo, desc_tamano, tasa
	Oficina	id_Oficina, telefono, direccion, ciudad, pais
	Administrador	id_Administrador
PIU_Vehiculo	Vehiculo	matricula, modelo, color, marca
	Grupo	id_Grupo, desc_tamano, tasa
	Minibus	nro_asientos
	Turismo	nro_puertas, es_Diesel, es_Gas
	Administrador	id_Administrador
PIU_Oficina	Oficina	id_Oficina, telefono, direccion, ciudad, pais
	Administrador	id_Administrador

8) Identificar los Movimientos de Datos

La Figura 18 presenta los movimientos de datos que pueden ocurrir en el sistema de Alquiler de coches.

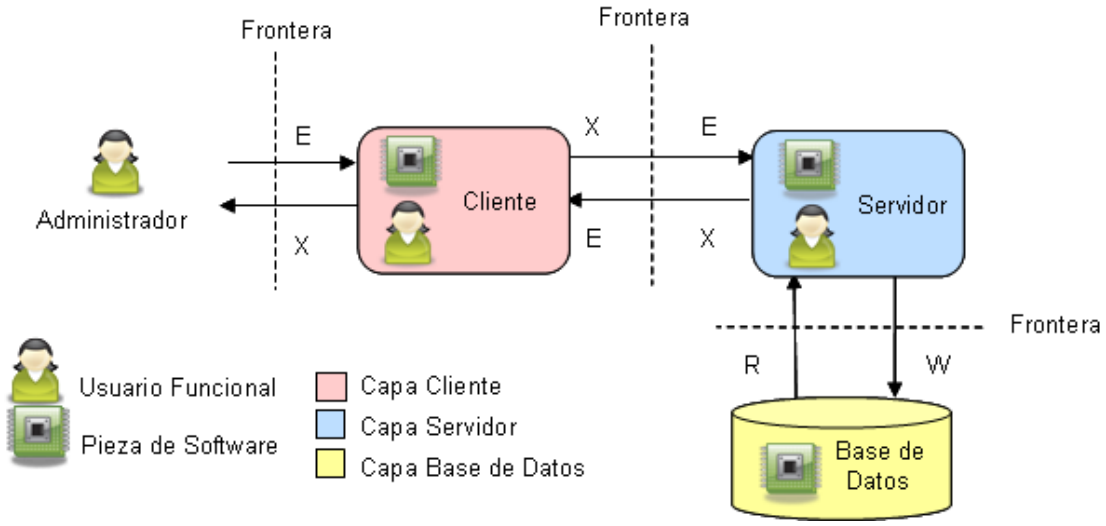


Figura 18. Movimientos de datos que pueden ocurrir en el sistema de Alquiler de coches.

Para identificar los movimientos de datos del sistema de Alquiler de coches se ha aplicado un conjunto de reglas que comprende desde la *regla 13 a la regla 62* del diseño de OOmCFP.

Para el proceso funcional PIU_Alquiler se aplican las reglas a los elementos contenidos en ese proceso funcional (vea la Tabla 8). Aplicando la *regla 13* al conjunto de visualización DS_Alquiler se identifica un movimiento de datos de salida (X) para la capa cliente. Luego, aplicando la *regla 14* a DS_Alquiler se identifican tres movimientos de datos entrada (E) para la capa cliente, tres movimientos de datos de salida (X) para la capa servidor, y tres movimientos de lectura (R) para la capa servidor por las clases Alquiler, Vehiculo y Cliente que contribuyen al conjunto de visualización DS_Alquiler. Aplicando la *regla 17* a DS_Alquiler se identifica un movimiento de lectura (R) para la capa servidor por la clase Grupo.

Posteriormente se analizan los servicios asociados a A_Alquiler. Aplicando la *regla 28* a SIU_create_instance se identifica un movimiento de entrada (E) y un movimiento de salida (X) para la capa cliente, y un movimiento de entrada (E) para la capa servidor por el conjunto de argumentos dato-valuados del servicio, representados por la clase Alquiler. Aplicando la *regla 29* se identifican tres movimientos de entrada (E) y tres movimientos de salida (X) para la capa cliente, y tres movimientos de entrada (E) para la capa servidor por los argumentos objeto-valuados que corresponden a las clases Cliente, Vehiculo y Oficina. Luego, aplicando la *regla 52* se identifica un movimiento de escritura para la capa servidor debido a que SIU_create_instance es un evento de tipo *new*.

Aplicando la *regla 29* a SIU_delete_instance se identifica un movimiento de entrada (E) y un movimiento de salida (X) para la capa cliente, y un movimiento de entrada (E) para la capa servidor por el argumento objeto-valuado correspondiente a la clase Alquiler. Además, aplicando la *regla 52* se identifica un movimiento de escritura para la capa servidor debido a que SIU_delete_instance es un evento de tipo *destroy*.

En cuanto a SIU_edit_instace, aplicando la *regla 28* se identifica un movimiento de entrada (E) y un movimiento de salida (X) para la capa cliente, y un movimiento de entrada (E) para la capa servidor por el conjunto de argumentos dato-valuados del servicio, representados por la clase Alquiler. Luego, se aplica la *regla 29* para identificar los argumentos objeto-valuados. Este servicio tiene un argumento objeto valuado que corresponde a la clase Alquiler. Dado que esta clase ya ha sido identificada mediante la *regla 28*,

no se identifican nuevos movimientos de datos. Finalmente, aplicando la *regla 52* se identifica un movimiento de escritura para la capa servidor debido a que SIU_edit_instance es un evento que tiene evaluaciones.

La siguiente tabla resume los movimientos de datos que ocurren cuando se ejecuta el proceso funcional PIU_Alquiler.

Tabla 12. Movimientos de datos del proceso funcional PIU_Alquiler.

Proceso funcional	**Elementos contenidos**		**Cliente**		**Servidor**			
			E	**X**	**E**	**X**	**R**	**W**
PIU_Alquiler	DS_Alquiler		3	1	0	3	4	0
	A_Alquiler	SIU_create_instance	4	4	4	0	0	1
		SIU_delete_instance	1	1	1	0	0	1
		SIU_edit_instance	1	1	1	0	0	1

Para el proceso funcional PIU_Vehiculo se aplican las reglas a los elementos contenidos en dicho proceso funcional (vea la Tabla 9). Aplicando la *regla 13* al conjunto de visualización DS_Vehiculo se identifica un movimiento de datos de salida (X) para la capa cliente. Luego, aplicando la *regla 14* a DS_Vehiculo se identifican dos movimientos de datos entrada (E) para la capa cliente, dos movimientos de datos de salida (X) para la capa servidor, y dos movimientos de lectura (R) para la capa servidor por las clases Vehiculo y Grupo que contribuyen al conjunto de visualización DS_Vehiculo.

Posteriormente se analizan los servicios asociados a A_Vehiculo. Aplicando la *regla 28* a SIU_create_instance se identifica un movimiento de entrada (E) y un movimiento de salida (X) para la capa cliente, y un movimiento de entrada (E) para la capa servidor por el conjunto de argumentos dato-valuados del servicio, representados por la clase Vehiculo. Aplicando la *regla 29* se identifican un movimiento de entrada (E) y un movimiento de salida (X) para la capa cliente, y un movimiento de entrada (E) para la capa servidor por el argumento objeto-valuado que corresponde a la clase Grupo. Luego, aplicando la *regla 52* se identifica un movimiento de escritura para la capa servidor debido a que SIU_create_instance es un evento de tipo *new*.

Aplicando la *regla 29* a SIU_delete_instance se identifica un movimiento de entrada (E) y un movimiento de salida (X) para la capa cliente, y un movimiento de entrada (E) para la capa servidor por el argumento objeto-valuado correspondiente a la clase Vehiculo. Además, aplicando la *regla 52* se identifica un

movimiento de escritura para la capa servidor debido a que SIU_delete_instance es un evento de tipo *destroy*.

En cuanto a SIU_edit_instace, aplicando la *regla 28* se identifica un movimiento de entrada (E) y un movimiento de salida (X) para la capa cliente, y un movimiento de entrada (E) para la capa servidor por el conjunto de argumentos dato-valuados del servicio, representados por la clase Vehiculo. Luego, aplicando la *regla 52* se identifica un movimiento de escritura para la capa servidor debido a que SIU_edit_instance es un evento que tiene evaluaciones.

Luego, se analizan las unidades de interacción contenidas en N_Vehiculo. Aplicando la *regla 13* al conjunto de visualización DS_Minibus de PIU_Minibus se identifica un movimiento de datos de salida (X) para la capa cliente. Luego, aplicando la *regla 14* a DS_Minibus se identifican dos movimientos de datos entrada (E) para la capa cliente, dos movimientos de datos de salida (X) para la capa servidor, y dos movimientos de lectura (R) para la capa servidor por las clases Vehiculo y Minibus que contribuyen al conjunto de visualización DS_Minibus. En cuanto a SIU_create_minibus de las acciones A_Minibus, aplicando la *regla 28* se identifica un movimiento de entrada (E) y un movimiento de salida (X) para la capa cliente, y un movimiento de entrada (E) para la capa servidor por el conjunto de argumentos dato-valuados del servicio, representados por la clase Minibus. Aplicando la *regla 29* se identifican un movimiento de entrada (E) y un movimiento de salida (X) para la capa cliente, y un movimiento de entrada (E) para la capa servidor por el argumento objeto-valuado que corresponde a la clase Vehiculo.

Aplicando la *regla 13* al conjunto de visualización DS_Turismo de PIU_Turismo se identifica un movimiento de datos de salida (X) para la capa cliente. Luego, aplicando la *regla 14* a DS_Turismo se identifican dos movimientos de datos entrada (E) para la capa cliente, dos movimientos de datos de salida (X) para la capa servidor, y dos movimientos de lectura (R) para la capa servidor por las clases Vehiculo y Turismo que contribuyen con sus atributos al conjunto de visualización DS_Turismo. En cuanto a SIU_create_turismo de las acciones A_Turismo, aplicando la *regla 28* se identifica un movimiento de entrada (E) y un movimiento de salida (X) para la capa cliente, y un movimiento de entrada (E) para la capa servidor por el conjunto de argumentos dato-valuados del servicio, representados por la clase Turismo. Aplicando la *regla 29* se identifican un movimiento de entrada (E) y un movimiento de salida (X) para la capa cliente, y un movimiento de entrada (E)

para la capa servidor por el argumento objeto-valuado que corresponde a la clase Vehiculo.

La siguiente tabla resume los movimientos de datos que ocurren cuando se ejecuta el proceso funcional PIU_Vehiculo.

Tabla 13. Movimientos de datos del proceso funcional PIU_Vehiculo.

Proceso funcional	**Elementos contenidos**				**Cliente**		**Servidor**			
					E	**X**	**E**	**X**	**R**	**W**
PIU_Vehiculo	DS_Vehiculo				2	1	0	2	2	0
	A_Vehiculo	SIU_create_instance			2	2	2	0	0	1
		SIU_delete_instance			1	1	1	0	0	1
		SIU_edit_instance			1	1	1	0	0	1
	N_Vehiculo	PIU_Minibus	DS_Minibus		2	1	0	2	2	0
			A_Minibus	SIU_create_minibus	2	2	2	0	0	0
		PIU_Turismo	DS_Turismo		2	1	0	2	2	0
			A_Turismo	SIU_create_turismo	2	2	2	0	0	0

Finalmente se identifican los movimientos de datos para el proceso funcional PIU_Oficina aplicando las reglas a los elementos contenidos en dicho proceso funcional (vea la Tabla 10). Aplicando la *regla 13* al conjunto de visualización DS_Oficina se identifica un movimiento de datos de salida (X) para la capa cliente. Luego, aplicando la *regla 14* a DS_Oficina se identifica un movimiento de datos de entrada (E) para la capa cliente, un movimiento de datos de salida (X) para la capa servidor, y un movimiento de datos de lectura (R) para la capa servidor por la clase Oficina.

Posteriormente se analizan los servicios asociados a las acciones A_Oficina. Aplicando la *regla 28* a SIU_create_instance se identifica un movimiento de entrada (E) y un movimiento de salida (X) para la capa cliente, y un movimiento de entrada (E) para la capa servidor por el conjunto de argumentos dato-valuados del servicio, representados por la clase Oficina. Aplicando la *regla 52* se identifica un movimiento de escritura para la capa servidor debido a que SIU_create_instance es un evento de tipo *new*.

Aplicando la *regla 29* a SIU_delete_instance se identifica un movimiento de entrada (E) y un movimiento de salida (X) para la capa cliente, y un movimiento de entrada (E) para la capa servidor por el argumento objeto-valuado correspondiente a la clase Oficina. Además, aplicando la *regla 52* se identifica un movimiento

de escritura para la capa servidor debido a que SIU_delete_instance es un evento de tipo *destroy*.

En cuanto a SIU_edit_instace, aplicando la *regla 28* se identifica un movimiento de entrada (E) y un movimiento de salida (X) para la capa cliente, y un movimiento de entrada (E) para la capa servidor por el conjunto de argumentos dato-valuados del servicio, representados por la clase Oficina. Luego, aplicando la *regla 52* se identifica un movimiento de escritura para la capa servidor debido a que SIU_edit_instance es un evento que tiene evaluaciones.

En resumen, la siguiente tabla muestra los movimientos de datos que ocurren cuando se ejecuta el proceso funcional PIU_Oficina.

Tabla 14. Movimientos de datos del proceso funcional PIU_Oficina.

Proceso funcional	**Elementos contenidos**		**Cliente**		**Servidor**			
			E	**X**	**E**	**X**	**R**	**W**
PIU_Oficina	DS_Oficina		1	1	0	1	1	0
	A_Oficina	SIU_create_instance	1	1	1	0	0	1
		SIU_delete_instance	1	1	1	0	0	1
		SIU_edit_instance	1	1	1	0	0	1

Una vez identificados todos los movimientos de datos de los procesos funcionales del sistema de Alquiler de coches, se concluye con la construcción del modelo de software.

5.2.3 Asignación de Reglas de Medición

La asignación de reglas de medición ha sido realizada siguiendo el proceso de aplicación de OOmCFP (vea la Figura 12). De esta manera, en esta sección se realiza la medición de los elementos que contribuyen al tamaño funcional del sistema de Alquiler de coches.

1) Calcular el Tamaño Funcional de cada Proceso Funcional

Según se ha especificado en la sección 4.4 del diseño de OOmCFP, a cada movimiento de datos se le asigna una unidad de tamaño funcional referida como 1 CFP (Cosmic Funtion Point).

Utilizando las *reglas 63* y *64*, que especifican que el tamaño funcional de un proceso funcional corresponde a la suma de los movimientos de datos que ocurren en él, se han obtenido los

tamaños funcionales para los procesos funcionales del sistema de Alquiler de coches (vea la Tabla 15).

Tabla 15. Tamaño funcional de los procesos funcionales del sistema de Alquiler de coches.

Proceso Funcional	Capa Cliente	Capa Servidor
PIU_Alquiler	16 CFP	16 CFP
PIU_Vehiculo	25 CFP	23 CFP
PIU_Oficina	8 CFP	8 CFP

2) Calcular el Tamaño Funcional de cada Capa de Software

El tamaño funcional de la capa cliente y de la capa servidora del sistema de Alquiler de coches se ha calculado aplicando las *reglas 65* y *66* respectivamente. Estas reglas especifican que el tamaño funcional de una capa de software es igual a la agregación de los tamaños funcionales de los procesos funcionales que ocurren en esa capa de software. Así, para la capa cliente del sistema de Alquiler de coches se ha obtenido un tamaño funcional de 49 CFP, y para la capa servidor del sistema de Alquiler de coches se ha obtenido un tamaño funcional de 47 CFP.

3) Calcular el Tamaño Funcional de la Aplicación

Finalmente, aplicando la *regla 67* detallada en el diseño de OOmCFP, que especifica que el tamaño funcional de una aplicación OO-Method es igual a la suma de los tamaños funcionales de las capas de software que la componen, se ha obtenido el valor de 96 CFP como el tamaño funcional del sistema de Alquiler de coches.

5.3 Conclusiones

En este capítulo se ha presentado la aplicación del procedimiento de medición OOmCFP siguiendo rigurosamente el modelo de procesos para la definición de métodos de medición propuesto por Jacquet y Abran [Jacquet97] [Abran99].

Para aplicar correctamente el procedimiento OOmCFP, en este capítulo se ha definido un proceso compuesto de 11 pasos secuenciales. Este proceso de aplicación puede ser generalizado para aplicar otros procedimientos de medición basados en COSMIC. La aplicación del procedimiento ha sido validada en un estudio piloto [Marin08].

En este capítulo se han descrito los tamaños funcionales obtenidos en la aplicación manual de OOmCFP a casos de estudio, y se ha ilustrado el proceso de aplicación de OOmCFP paso a paso en la medición del tamaño funcional del caso de estudio de un sistema de Alquiler de coches.

Capítulo 6
Automatización de OOmCFP

La medición manual del tamaño funcional generalmente toma mucho tiempo y puede involucrar errores de precisión dada la naturaleza humana. Esta situación ocurre tanto en la medición de casos de estudio, como en la medición de aplicaciones reales. Sin embargo, esta situación es crítica cuando se trata de la medición de aplicaciones reales, que pueden ser tan grandes que la persona que realiza la medición olvide la medición de algunas funcionalidades, cuente las funcionalidades más de una vez, confunda los valores obtenidos, etc. Por este motivo, es necesario automatizar la medición del tamaño funcional, para poder obtener el tamaño funcional de las aplicaciones utilizando recursos mínimos y disminuyendo los errores de precisión en el cálculo del tamaño funcional.

Para el caso de la medición de tamaño funcional a partir de modelos conceptuales, la realización de una medición manual utilizando OOmCFP en un modelo conceptual correspondiente a un sistema de Facturación pequeño ha tomado 70 minutos. Teniendo en cuenta que ese modelo conceptual tenía sólo 4 clases, y los modelos conceptuales OO-Method de aplicaciones reales pueden llegar a tener 100 o más clases, sería necesario utilizar más de 116 horas para medir sistemas reales. Por esta razón, es muy importante automatizar el procedimiento de medición OOmCFP para poder medir eficientemente el tamaño funcional de aplicaciones reales, disminuyendo el tiempo necesario para realizar la medición.

Además, la automatización de OOmCFP permite evitar la posibilidad de incorporar errores humanos en la medición, debido a que la medición se realiza automáticamente a partir del modelo conceptual de la aplicación. La automatización de OOmCFP también permite reducir los costos incurridos cuando se lleva a cabo una medición, debido a que las mediciones se realizan en menos tiempo y sin errores. La automatización de OOmCFP también repercute en el aumento de la productividad de las personas que realizan las mediciones de tamaño funcional. Finalmente, la automatización de un procedimiento de medición facilita la utilización del tamaño funcional en otros modelos de calidad (modelos de defectos, modelos de presupuesto, etc.), por lo que también presenta ventajas en esos modelos. En resumen, la

automatización de un procedimiento de medición presenta una serie de beneficios que contribuyen a que el procedimiento de medición sea utilizado industrialmente.

En este capítulo se presenta un resumen de herramientas que automatizan procedimientos de medición basados en COSMIC, la herramienta que automatiza el procedimiento de medición OOmCFP, cómo se utiliza la herramienta, y los resultados de 5 casos reales de la empresa CARE Technologies [CARE] medidos con la herramienta OOmCFP.

6.1 Herramientas que Automatizan COSMIC

Dadas las ventajas que presenta la automatización de un procedimiento de medición de tamaño funcional, hoy en día existen varias herramientas que han automatizado la medición de procedimientos basados en COSMIC.

Mendes et al. [Mendes96] presenta un Framework general que permite clasificar las herramientas que automatizan la medición del tamaño funcional. Este framework permite clasificar las herramientas según la manera en que realizan el conteo de puntos de función y según la posterior utilización de estos puntos de función. Para realizar esto, el framework definido por Mendes et al. se compone de cuatro dimensiones y diez categorías que se muestran en la siguiente tabla:

Tabla 16. Dimensiones y categorías del Framework para clasificar herramientas que automatizan la medición del tamaño funcional (adaptado de [Mendes96]).

Dimension	**Categoría**
Soporte a la Medición	Documentación Entrenamiento
Medición	Recolección y cálculo de datos Sistema experto Medición automática Estimación a alto nivel
Almacenamiento	Repositorio de mediciones de software
Utilización	Estimación y predicción Gestión de proyectos Evaluación (por ejemplo benchmarking)

La dimensión *soporte a la medición* permite clasificar las herramientas según el soporte que presentan respecto del método o procedimiento de medición automatizado. Para esto, esta

dimensión presenta dos categorías: documentación y entrenamiento. La categoría *documentación* corresponde a manuales del método de medición (o procedimiento) automatizado por la herramienta y casos de estudio. La categoría *entrenamiento* corresponde a guías de medición, ejemplos, tutoriales, etc.

La dimensión *medición* permite clasificar las herramientas según la forma en que realizan la medición. Esta dimensión se compone de cuatro categorías que no son excluyentes entre sí: recolección y cálculo de datos, sistema experto para medir, medición automática y estimación a alto nivel. La categoría *recolección y cálculo de datos* está presente en herramientas que permiten ingresar datos del sistema que se quiere medir. La categoría *sistema experto* está presente en las herramientas que tiene un sistema experto que ayuda a realizar la medición. La categoría *medición automática* está presente en aquellas herramientas que permiten identificar lo que se va a medir de manera automática y realizan la obtención del tamaño funcional de un sistema de manera completamente automática. La categoría *estimación a alto nivel* está presente en herramientas que realizan una exploración sobre antiguos proyectos para obtener una estimación del tamaño funcional.

La dimensión de *almacenamiento* permite clasificar las herramientas según las opciones de persistencia de las medidas obtenidas por la herramienta. Esta dimensión sólo tiene una categoría: *repositorio de mediciones de software*. Esta categoría esta presente en aquellas herramientas que almacenan los valores obtenidos, ya sea en una base de datos, en un archivo, etc.

Finalmente, la dimensión de *utilización* permite clasificar las herramientas según la utilización del tamaño funcional obtenido. Esta dimensión se compone de tres categorías: estimación y predicción, gestión de proyectos, y evaluación. La categoría de *estimación y predicción* está presente en aquellas herramientas que utilizan el tamaño funcional para estimar esfuerzo, riesgo, producción, duración, defectos, personas, etc. La categoría de *gestión de proyectos* está presente en aquellas herramientas que utilizan el tamaño funcional en la planificación de proyectos, el control de proyectos, pronósticos en proyectos, etc. La categoría *evaluación* está presente en aquellas herramientas que utilizan el tamaño funcional de un proyecto para compararlo con el tamaño funcional de otros proyectos (benchmarking), para fijar líneas base para proyectos que tengan características similares, etc.

En base a las dimensiones y categorías presentadas anteriormente es posible comparar las herramientas existentes

tanto en el mercado como en la comunidad académica. En 2006, Stambollian et al. [Stambollian06] presenta un resumen de herramientas disponibles en el mercado utilizando el Framework de Mendes et al. [Mendes96]. Los resultados obtenidos se presentan en la Tabla 17, en donde se ha marcado con una **X** las celdas en que cada herramienta satisface parcial o completamente cada categoría.

Tabla 17. Clasificación de herramientas industriales que automatizan COSMIC (adaptado de [Stambollian06]).

Dimensión	Categoría	COSMIC Xpert	ISBG	MeterIT suite	Experience Pro	Know-ledge Plan	SIESTA
Soporte a la Medición	Documentación	**X**		**X**			
	Entrenamiento	**X**		**X**			
Medición	Recolección y cálculo de datos			**X**	**X**	**X**	**X**
	Sistema experto	**X**					
	Medición automática						
	Estimación a alto nivel			**X**	**X**	**X**	**X**
Almacena-miento	Repositorio de mediciones de software		**X**	**X**	**X**	**X**	
Utilización	Estimación y predicción			**X**	**X**	**X**	**X**
	Gestión de proyectos			**X**	**X**	**X**	**X**
	Evaluación (por ejemplo benchmarking)		**X**	**X**	**X**	**X**	

En la Tabla 17 se puede observar que sólo dos de las seis herramientas disponibles en el mercado dan soporte a la medición, es decir presentan documentación del método automatizado y ejemplos de mediciones.

En cuanto a la medición, en la Tabla 17 se puede observar que la herramienta ISBG no contempla la medición del tamaño funcional. La herramienta COSMIC Xpert utiliza un sistema experto para entregar una solución en base a la interpretación de conceptos introducidos por el usuario. Las restantes herramientas permiten ingresar datos del sistema que será medido (por ejemplo los usuarios funcionales, los procesos funcionales, los movimientos de datos, etc.) y con estos datos realizan estimaciones del tamaño funcional según tamaños funcionales almacenados en un repositorio. Es importante destacar que ninguna de las herramientas disponibles en le mercado permiten realizar una

medición automática del tamaño funcional, es decir, no existen herramientas que identifiquen automáticamente los movimientos de datos de un sistema para obtener el tamaño funcional. Dadas las ventajas que provee la automatización, es de gran importancia para la industria que esta característica sea abordada por las herramientas.

En cuanto al repositorio de las medidas obtenidas, sólo cuatro de las herramientas tienen persistencia, facilitando la posterior utilización del tamaño funcional obtenido.

Con respecto a la utilización del tamaño funcional obtenido, la mayoría de las herramientas satisfacen alguna categoría de esta dimensión. Solamente la herramienta COSMIC Xpert no satisface ninguna de las categorías. La herramienta ISBG utiliza el tamaño funcional para la evaluación de proyectos. La herramienta SIESTA utiliza el tamaño funcional para la estimación y predicción, y gestión de proyectos. Las herramientas restantes utilizan el tamaño funcional en las tres categorías de esta dimensión.

Por otro lado, en el capítulo 2 se han descrito tres propuestas académicas que presentan herramientas para automatizar los procedimientos de medición basados en COSMIC: la propuesta de Azzouz et al. [Azzouz04], la propuesta de Grau et al. [Grau07], y la propuesta de Diab et al. [Diab05]. La Tabla 18 presenta la clasificación de estas herramientas utilizando el framework de Mendes et al. [Mendes96].

Tabla 18. Clasificación de herramientas académicas que automatizan COSMIC.

Dimensión	Categoría	Azzouz	Diab	Grau
Soporte a la Medición	Documentación	**X**	**X**	**X**
	Entrenamiento			
Medición	Recolección y cálculo de datos	**X**	**X**	**X**
	Sistema experto			
	Medición automática		**X**	
	Estimación a alto nivel	**X**		**X**
Almacenamiento	Repositorio de mediciones de software	**X**		
Utilización	Estimación y predicción			
	Gestión de proyectos			
	Evaluación (por ejemplo benchmarking)			

En la Tabla 18 se puede observar que las tres herramientas analizadas presentan soporte a la medición. Este soporte esta dado por la documentación de las correspondencias definidas entre los conceptos de COSMIC y los conceptos de los modelos conceptuales que miden.

En cuanto a la medición, las herramientas de Azzouz y Grau permiten ingresar datos a partir de las especificaciones de requisitos y estimar el tamaño funcional. Sólo la herramienta de Diab permite realizar una medición automática del tamaño funcional de acuerdo a las especificaciones de los modelos RRRT.

Con respecto al almacenamiento, sólo la propuesta de Azzouz presenta un repositorio persistente de los tamaños funcionales obtenidos. Si bien todas las herramientas plantean que en un futuro pueden ser utilizadas para la estimación y la gestión de proyectos, no hemos encontrado evidencias de futuras versiones de las herramientas que efectivamente utilicen el tamaño funcional obtenido.

En resumen, es posible observar que las herramientas del mercado cubren casi todas las categorías de las dimensiones del framework. Sin embargo, ninguna de estas herramientas presenta la categoría de medición automática. Por otro lado, ninguna de las herramientas académicas utiliza el tamaño funcional obtenido, pero existe una herramienta que satisface la categoría de medición automática para sistemas de tiempo real. Dado el desarrollo masivo de sistemas de información de gestión, y las ventajas que presenta el método de medición COSMIC, se pone de manifiesto la necesidad de contar con una herramienta que permita medir automáticamente el tamaño funcional exacto de aplicaciones de gestión utilizando un procedimiento de medición basado en COSMIC.

6.2 Herramienta OOmCFP

Para el procedimiento de medición OOmCFP, la fase de estrategia de medición puede ser automatizada completamente dado que el propósito, el alcance y el nivel de granularidad son valores que pueden ser parametrizados en las aplicaciones OO-Method (que posteriormente pueden ser seleccionados por la persona que realice la medición) y que los usuarios funcionales con sus fronteras son identificados desde el modelo conceptual OO-Method (suponiendo que el modelo es correcto, completo y sin defectos).

La fase de correspondencias también puede ser completamente automatizada debido a que los procesos funcionales, los grupos de datos y los atributos de datos en OOmCFP se identifican en el modelo conceptual OO-Method.

Finalmente, la fase de medición de OOmCFP puede automatizarse completamente ya que los movimientos de datos son identificados en el modelo conceptual OO-Method y las reglas de medición sólo agregan los valores obtenidos.

De esta manera, el procedimiento de medición OOmCFP puede ser automatizado completamente, ya que todas sus fases pueden ser automatizadas completamente. La herramienta que automatiza OOmCFP ha sido desarrollada utilizando el lenguaje C# de Visual Studio .Net 2003.

La herramienta que automatiza OOmCFP ha sido desarrollada con el objetivo de medir de manera completa los modelos conceptuales OO-Method utilizados industrialmente. Con respecto al Framework de Mendes et al. [Mendes96] para la clasificación de herramientas, la herramienta OOmCFP inicialmente ha sido desarrollada teniendo en cuenta las dos primeras dimensiones: soporte a la medición y medición. El soporte a la medición está dado por el manual de usuario de la herramienta y la guía de medición con OOmCFP que presenta todas las reglas del procedimiento. En cuanto a la medición misma, la medición se realiza automáticamente a partir de un modelo conceptual OO-Method. Para medir modelos conceptuales utilizados industrialmente, la herramienta OOmCFP debe tener una arquitectura flexible que permita adaptarse a la evolución de los modelos conceptuales OO-Method, y debe ser ágil en el proceso de conteo de las aplicaciones generadas.

6.2.1 Arquitectura Flexible

Para proporcionar una *arquitectura flexible*, la herramienta OOmCFP ha sido desarrollada en un conjunto de pasos secuenciales perfectamente delimitados (vea la Figura 19), que facilita la incorporación de nuevas reglas de medición o modificaciones a las reglas existentes.

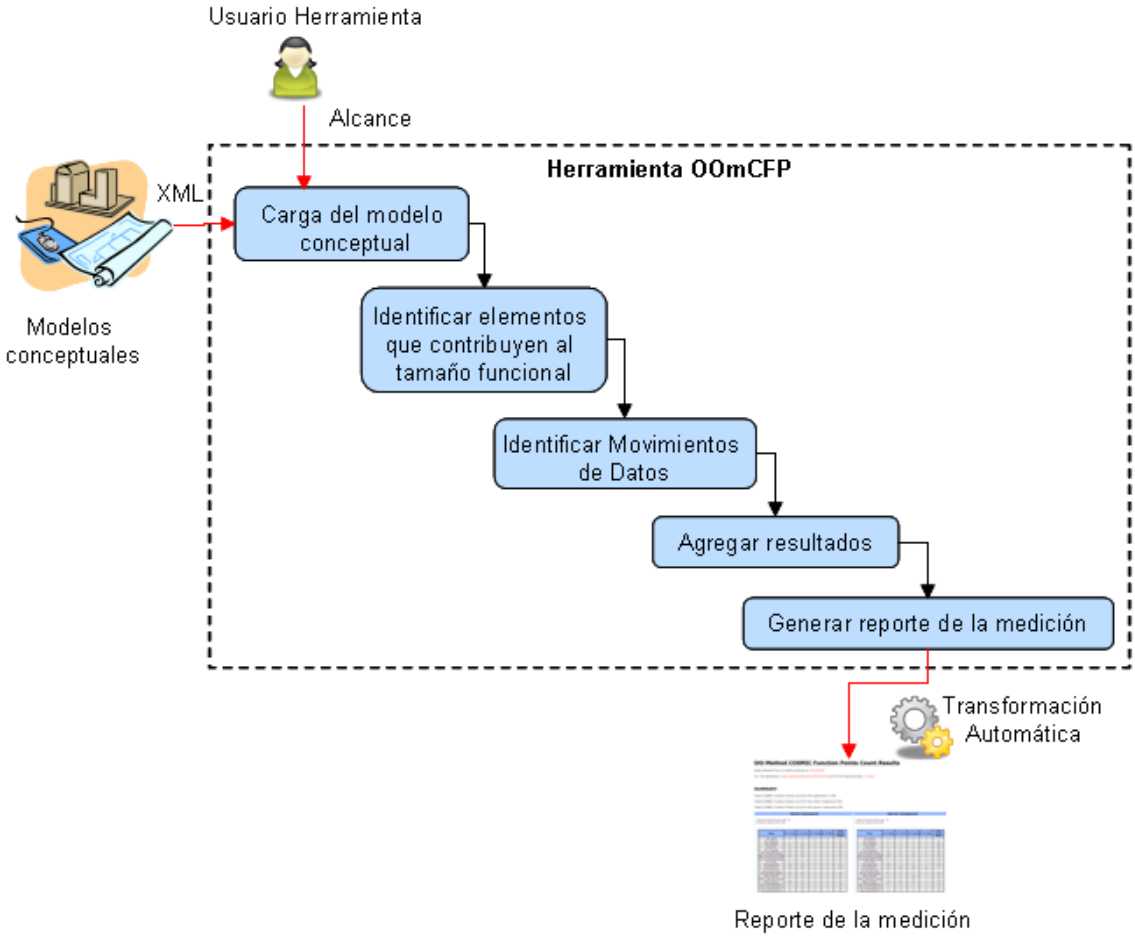


Figura 19. Proceso de medición de la herramienta OOmCFP.

En el primer paso de la herramienta se realiza la carga del modelo conceptual OO-Method a partir de un archivo XML, que es generado utilizando las facilidades de la herramienta OlivaNova Modeler [CARE]. Para realizar la carga del modelo, inicialmente se identifican los usuarios funcionales según el alcance seleccionado por el usuario de la herramienta. Teniendo en cuenta los usuarios funcionales identificados, se identifican las unidades de interacción presentes en el HAT del modelo de presentación, es decir, se identifican los procesos funcionales. Tanto los usuarios funcionales como los procesos funcionales son almacenados en memoria temporal.

Una vez identificados los procesos funcionales, en el segundo paso de la herramienta se identifican y cargan en memoria temporal todos los elementos contenidos en cada unidad de interacción que contribuyen al tamaño funcional de la medición. En otras palabras, por cada unidad de interacción almacenada como proceso funcional se identifican sus conjuntos de visualización, sus filtros, sus servicios, y las unidades de interacción a las cuales acceden. Luego, las unidades de interacción del proceso funcional

se analizan recursivamente hasta terminar con la identificación de los elementos contenidos en cada proceso funcional. En este paso también se aplican las reglas de duplicidad, para impedir la identificación de procesos funcionales y unidades de interacción que ya han sido identificadas.

En el tercer paso de la herramienta OOmCFP se realiza la identificación de los movimientos de datos que pueden ocurrir en esos procesos funcionales. En este paso están implementadas todas las reglas definidas para identificar los movimientos de datos (desde la *Regla 13* hasta la *Regla 62*). Para disminuir el acoplamiento de la identificación de los movimientos de datos, cada regla está implementada por primitiva del modelo conceptual. El resultado del análisis de los movimientos de datos es almacenado en cada elemento identificado.

En el cuarto paso de la herramienta OOmCFP se agregan los movimientos de datos identificados en cada proceso funcional de acuerdo al alcance seleccionado como parámetro y a las reglas de medición (desde la *Regla 63* a la *Regla 67*).

En el último paso de la herramienta OOmCFP se genera un reporte final de la medición en un archivo XML. Este archivo puede ser transformado en otros formatos utilizando XSLT. Por defecto, en este paso la herramienta OOmCFP transforma el archivo XML generado en páginas HTML.

6.2.2 Agilidad en el Conteo

En cuanto a la *agilidad en el conteo*, se ha detectado que la mayor carga de procesamiento y tiempo de ejecución ocurre en el segundo paso, donde se identifican los elementos contenidos en los procesos funcionales, y en el tercer paso, donde se identifican los movimientos de datos.

Para optimizar estos pasos se ha implementado un *mecanismo de caché*, que se centra en reducir la gran cantidad de tiempo procesamiento que se requiere para analizar elementos que han sido analizados previamente. Un ejemplo de esta situación es el análisis de servicios, ya que los servicios de tipo transacción pueden invocar a otras transacciones o invocarse a sí mismas recursivamente. El mecanismo de caché almacena los resultados obtenidos en el análisis de cada elemento que forma parte de un proceso funcional con un identificador único. De esta manera, al analizar un nuevo elemento contenido en un proceso funcional, la herramienta primero verifica si dicho elemento ya existe en el

caché, y en ese caso los movimientos de datos se recuperan directamente, evitando volver a identificar los movimientos de datos de ese elemento.

También se deben evitar los *desbordamientos de pila* al momento de ejecutar la medición. Debido a la naturaleza iterativa del análisis de puntos de función, se ha detectado que en modelos de gran tamaño se pueden producir desbordamientos durante la ejecución de la medición. Estos desbordamientos se producen por falta de memoria para el manejo de las iteraciones que se llevan a cabo para identificar los movimientos de datos. Particularmente, en caso del framework .Net versión 1.1, la pila para el almacenamiento posee un tamaño de 512KB, que se ve sobrepasada en el análisis de modelos de gran tamaño (por ejemplo, que tienen más de 80 clases).

Para evitar el desbordamiento de pila, los elementos relacionados se almacenan en un arreglo auxiliar. De esta manera, una vez acabado el análisis del primer elemento, se continúa analizando de manera secuencial los elementos almacenados en el arreglo auxiliar. Si los elementos relacionados a su vez referencian a otros elementos, éstos se van agregando al final del arreglo, eliminando el anidamiento de iteraciones y evitando el desbordamiento de pila (vea la Figura 20).

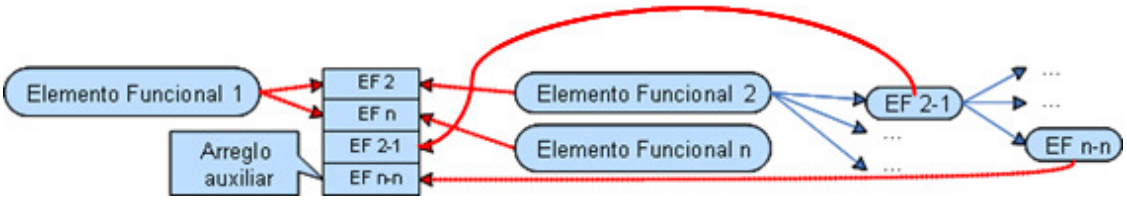


Figura 20. Esquema para evitar el desbordamiento de pila.

6.3 Utilizando la Herramienta OOmCFP

La herramienta OOmCFP ha sido desarrollada para que sea fácil de utilizar. Por esta razón, la herramienta OOmCFP sólo cuenta con 3 interfaces gráficas que permiten al usuario obtener el tamaño funcional de la aplicación modelada con OO-Method.

En la primera interfaz gráfica, el usuario debe indicar la ruta donde se encuentra el archivo XML que contiene toda la especificación del modelo conceptual OO-Method, la ruta donde el reporte HTML de la medición realizada será guardado, y además, el usuario debe seleccionar el alcance que tendrá la medición, que

para las aplicaciones OO-Method puede ser la pieza cliente, la pieza servidor o toda la aplicación. La Figura 21 muestra la ventana inicial de la herramienta OOmCFP.

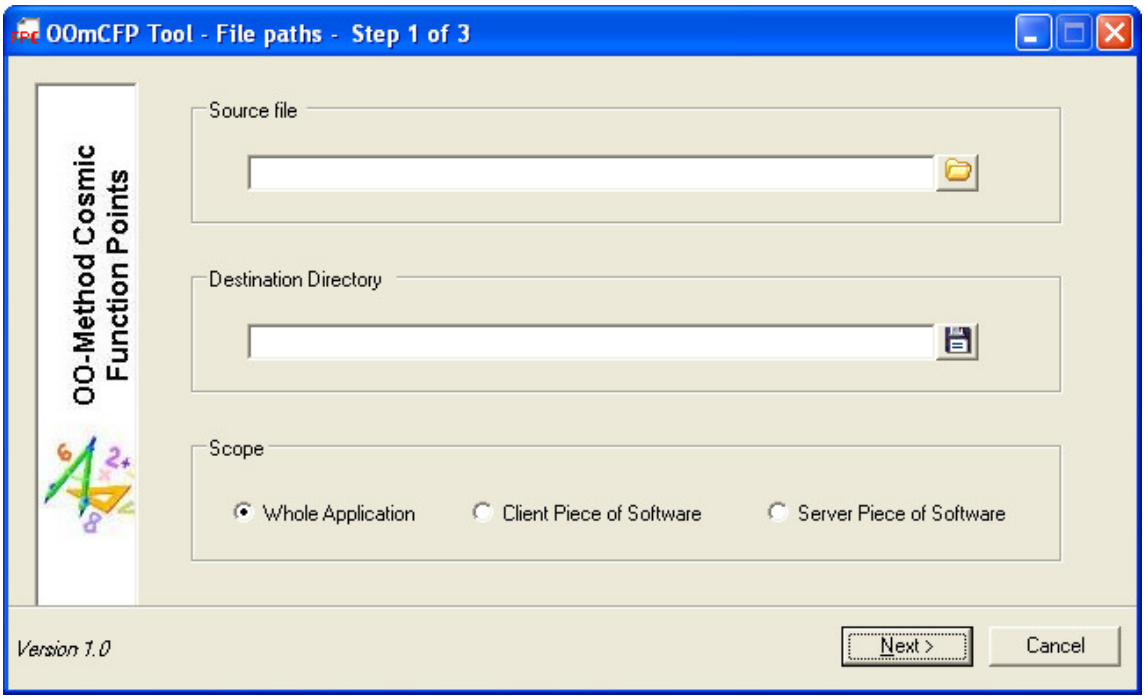


Figura 21. Ventana inicial de herramienta OOmCFP.

Con esta información, la herramienta valida que el archivo que contiene el modelo conceptual OO-Method sea un XML que corresponda al DTD versión 3.8 de la suite Olivanova. En caso que el archivo no corresponda a un archivo XML, o que no corresponda a la versión del DTD, la herramienta entrega un mensaje de error al usuario indicándole que debe seleccionar un archivo con las características mencionadas anteriormente.

Si el archivo indicado como modelo conceptual OO-Method es correcto, la herramienta verifica si el directorio donde se almacenará el reporte existe. En caso de que no exista, crea el directorio, y en caso de que exista, verifica si se tienen permisos de escritura para grabar posteriormente el reporte. Para cada una de las validaciones que realiza la herramienta, en caso de error la herramienta entrega un mensaje al usuario.

Una vez que el archivo fuente y el directorio destino han sido validados, la herramienta almacena la ruta del modelo conceptual, el archivo del modelo conceptual, la ruta del directorio destino y el alcance de la medición.

La segunda interfaz gráfica que presenta la herramienta es meramente informativa. En esta interfaz, la herramienta despliega al usuario un resumen con el nombre del modelo que será medido, la ruta del modelo, la ruta donde será guardado el reporte y el alcance de la medición (Figura 22).

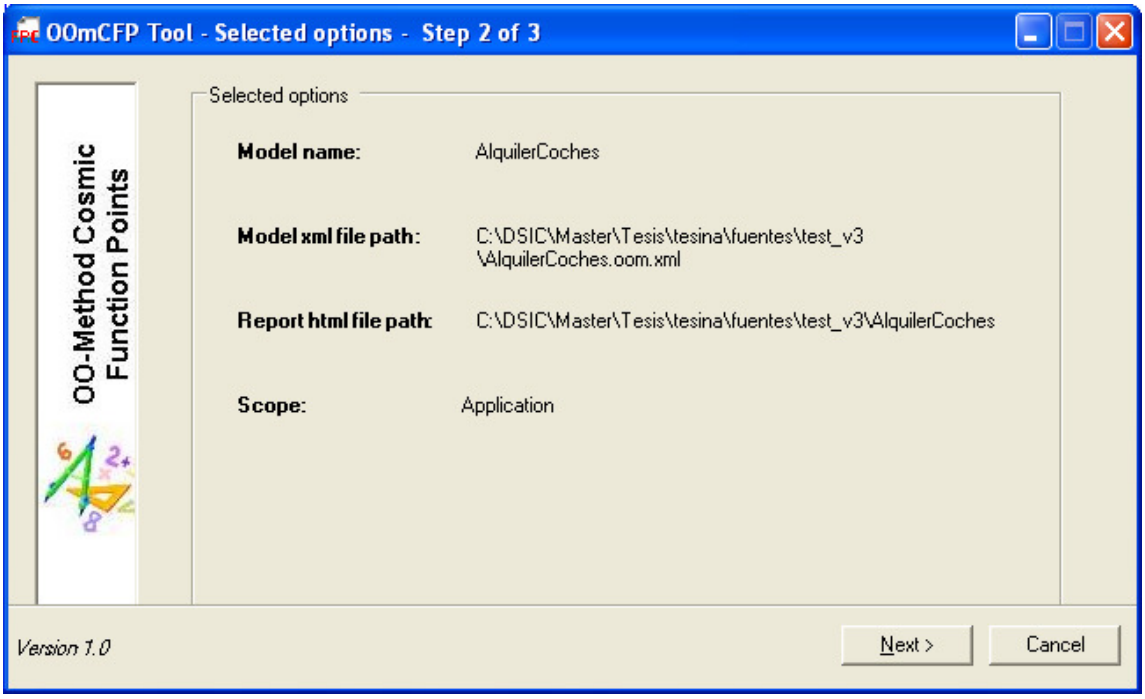


Figura 22. Ventana con resumen de opciones seleccionadas en la herramienta OOmCFP.

Una vez que el usuario presiona el botón *Next*, se identifican desde el archivo XML que contiene las especificaciones del modelo los elementos que contribuyen al tamaño funcional de la aplicación. Posteriormente, cuando todos los elementos han sido identificados y almacenados, se contabilizan los movimientos de datos utilizando las reglas definidas en OOmCFP. Finalmente, cuando todos los movimientos de datos que pueden ocurrir en la aplicación correspondiente al modelo que se está midiendo han sido identificados y almacenados, los resultados son agregados de acuerdo al alcance de la medición seleccionado.

En la tercera interfaz gráfica, la herramienta muestra el número de procesos funcionales que han sido medidos y el tamaño funcional obtenido según el alcance indicado (Figura 23). Cuando el usuario presiona el botón *Done*, la herramienta genera un archivo XML y un reporte HTML con todos los resultados de la medición, y luego almacena estos archivos en la ruta indicada por el usuario en la primera interfaz gráfica.

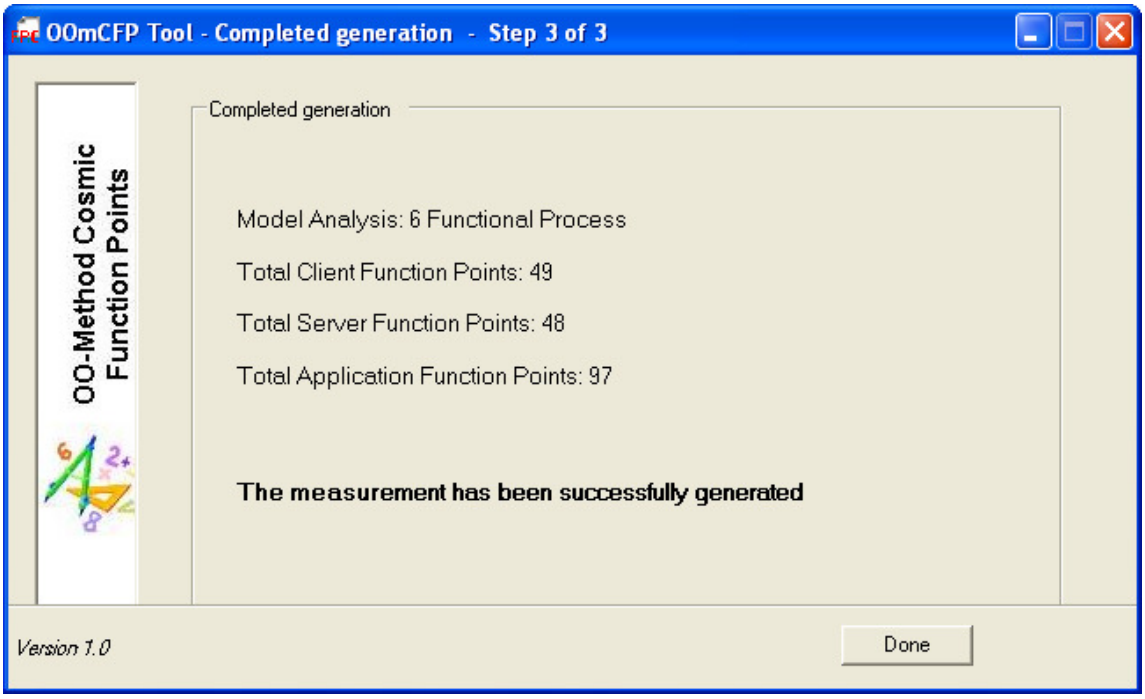
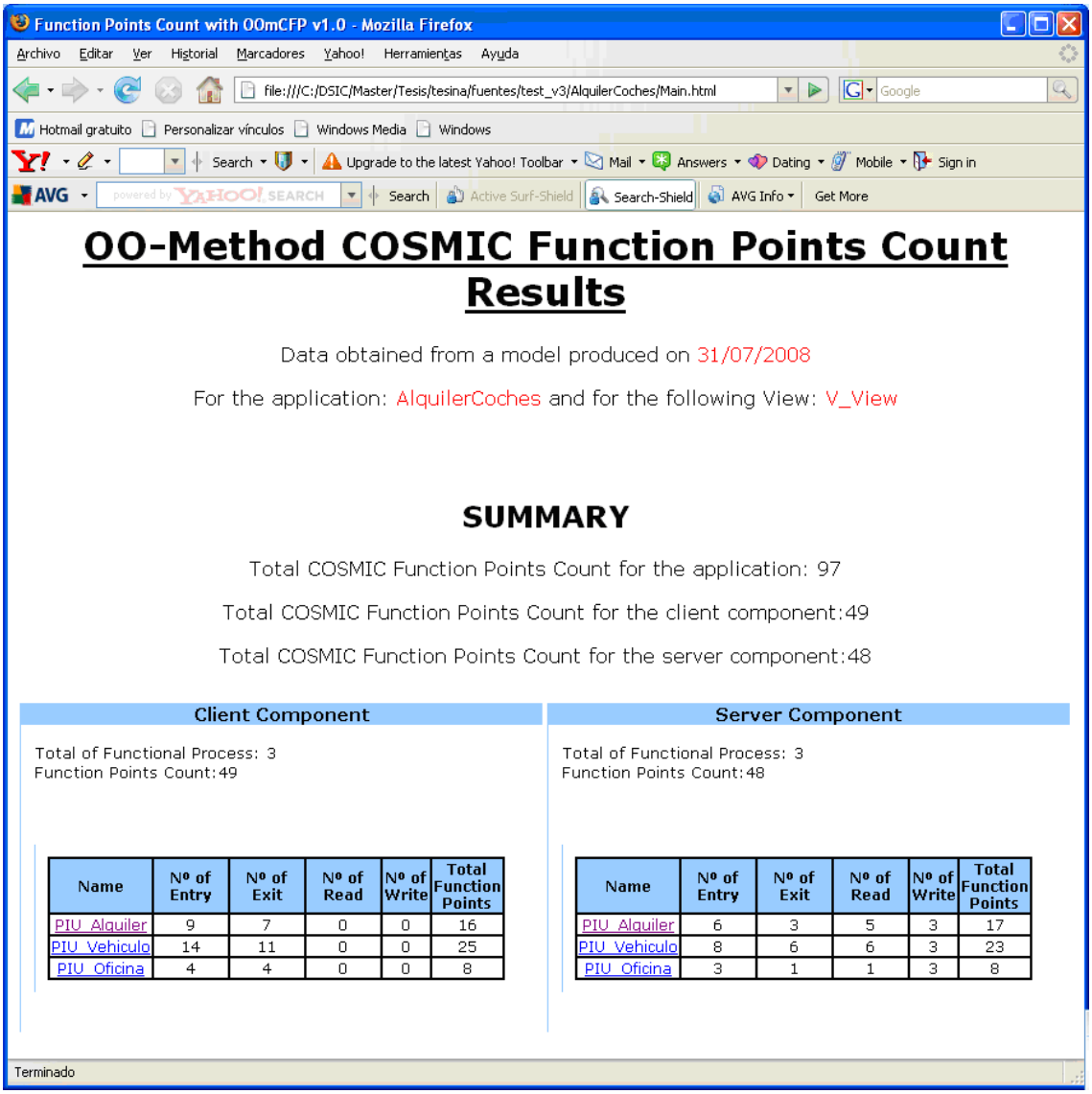


Figura 23. Procesos funcionales medidos y tamaño funcional obtenido en la herramienta OOmCFP.

La Figura 24 muestra la página inicial del reporte generado por la herramienta OOmCFP para la medición del modelo conceptual OO-Method utilizado en el capítulo 5 para ilustrar la aplicación de OOmCFP. Es decir, muestra el reporte generado en la medición del sistema de Alquiler de coches.



Name	Nº of Entry	Nº of Exit	Nº of Read	Nº of Write	Total Function Points
PIU_Alquiler	9	7	0	0	16
PIU_Vehiculo	14	11	0	0	25
PIU_Oficina	4	4	0	0	8

Name	Nº of Entry	Nº of Exit	Nº of Read	Nº of Write	Total Function Points
PIU_Alquiler	6	3	5	3	17
PIU_Vehiculo	8	6	6	3	23
PIU_Oficina	3	1	1	3	8

Figura 24. Página principal del reporte generado por la herramienta OOmCFP.

Desde la página inicial del reporte es posible navegar a los elementos contenidos en cada proceso funcional (Figura 25), y por cada elemento es posible navegar a los grupos de datos que corresponden a los movimientos de datos del elemento (Figura 26).

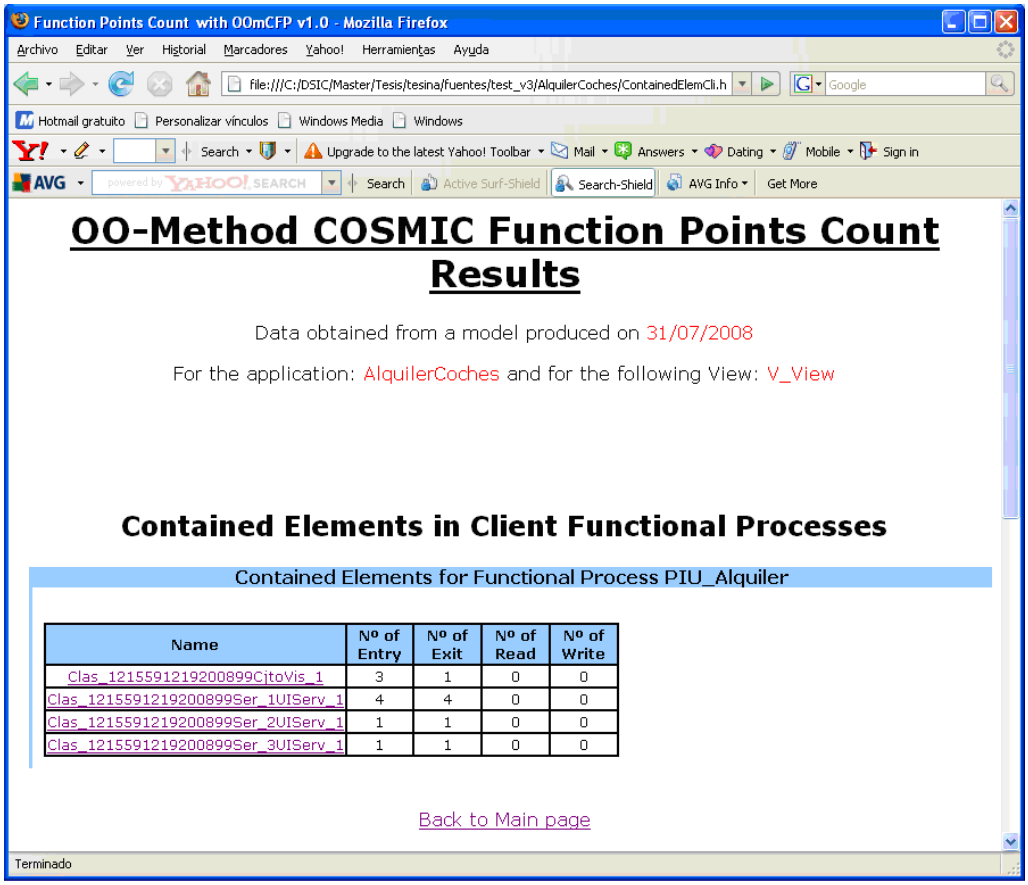


Figura 25. Elementos contenidos en los procesos funcionales.

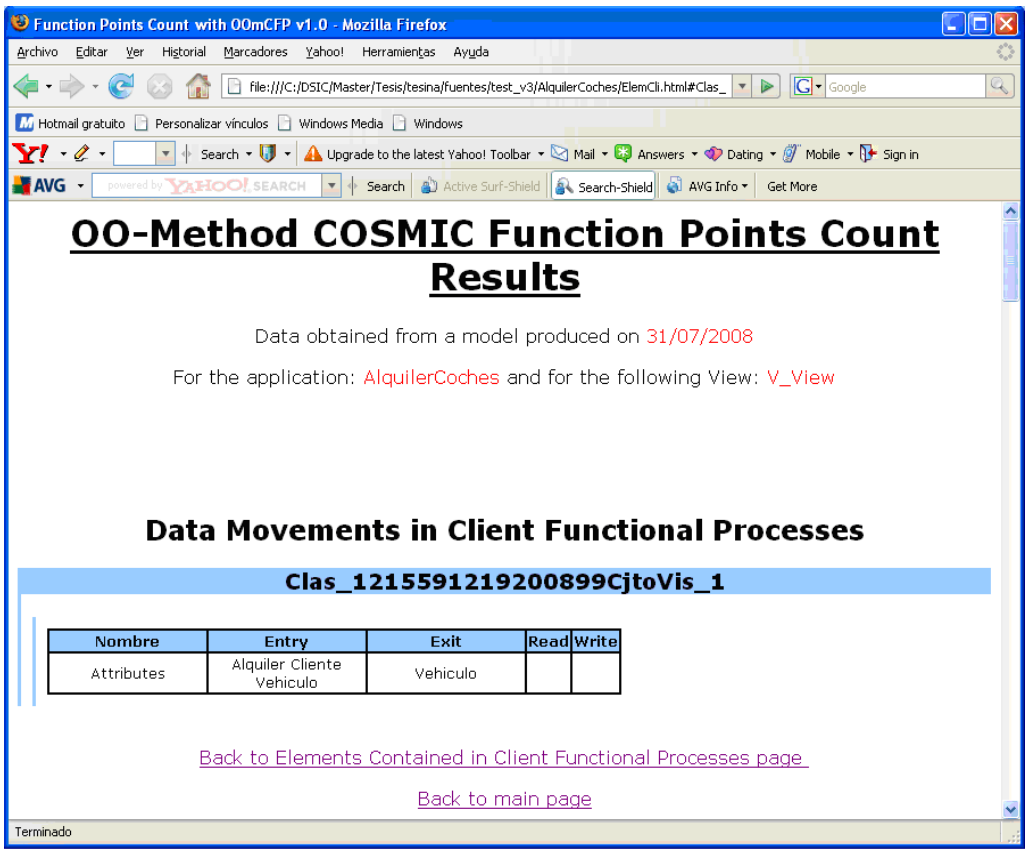


Figura 26. Movimientos de datos de los elementos contenidos en los procesos funcionales.

6.3 Medición con la Herramienta OOmCFP

La herramienta OOmCFP está siendo utilizada para medir casos reales de la empresa CARE Technologies [CARE] que tiene implementado un compilador de modelos OO-Method. Todos estos sistemas de información corresponden al dominio de sistemas de información de gestión.

En cuanto a la verificación de los resultados obtenidos por la herramienta, inicialmente la herramienta OOmCFP fue verificada mediante pruebas unitarias para comprobar la correcta implementación de cada regla de correspondencia, y de cada regla de medición. Una vez que las pruebas unitarias fueron superadas por la herramienta OOmCFP, se verificaron los resultados obtenidos por la herramienta con los resultados obtenidos en 6 casos de estudio medidos manualmente. Estos casos de estudio corresponden a los sistemas: Facturación, Alquiler de coches, Camping, Editorial, Agencia fotográfica, y Nota de gastos

Para ilustrar la utilización de la herramienta OOmCFP en casos reales, la Tabla 19 presenta los resultados obtenidos en la medición de 5 casos reales. Por motivos de privacidad, no es posible especificar las empresas a las que corresponden estos sistemas de información.

Tabla 19. Medición de casos reales.

Modelo	Número Procesos Funcionales	Tamaño Funcional Pieza Cliente	Tamaño Funcional Pieza Servidor	Tamaño Funcional Aplicación	Tiempo Medición
Modelo_1	30	201 CFP	174 CFP	375 CFP	2 seg.
Modelo_2	18	489 CFP	481 CFP	970 CFP	4 seg.
Modelo_3	192	1491 CFP	1395 CFP	2886 CFP	38 seg.
Modelo_4	108	1838 CFP	1759 CFP	3597 CFP	31 seg.
Modelo_5	258	4326 CFP	5851 CFP	10177 CFP	98 seg.

Según los resultados mostrados en la Tabla 19, se puede observar que algunas aplicaciones reales son muy grandes (como los modelos 3, 4 y 5), por lo que resultaría muy costoso en tiempo y recursos realizar la medición del tamaño funcional de manera manual. Gracias a la herramienta OOmCFP es posible realizar la medición en pocos minutos, y sin introducir errores debido a la gran cantidad de procesos funcionales y movimientos de datos que deben ser identificados.

Además, la herramienta OOmCFP permite obtener el tamaño funcional de cada pieza de software que compone la aplicación. Teniendo en cuenta que algunas aplicaciones son modeladas para generar automáticamente la interfaz gráfica de la aplicación (pieza cliente) para facilitar el trabajo con ERPs (piezas servidoras), contar con el tamaño funcional por pieza de software permite presupuestar correctamente la aplicación generada.

6.4 Conclusiones

En este capítulo se ha presentado la herramienta que automatiza completamente el procedimiento de medición OOmCFP. Para realizar la medición, el usuario de la herramienta debe especificar el modelo que será medido, la ruta donde se almacenará el reporte de la medición y el alcance que tendrá la medición.

La herramienta que implementa OOmCFP ha sido desarrollada teniendo en cuenta mecanismos de caché que permiten que el proceso de medición de modelos conceptuales OO-Method sea eficiente. Además, esta herramienta ha sido desarrollada teniendo en cuenta mecanismos para evitar el desbordamiento de memoria cuando se están midiendo modelos conceptuales de aplicaciones muy grandes. De esta manera, la herramienta OOmCFP realiza las mediciones de modelos conceptuales reales en pocos minutos, permitiendo agilizar el proceso de estimación del costo de la aplicación generada.

Finalmente, el uso de la herramienta que implementa OOmCFP elimina la posibilidad de incluir errores que pueden ocurrir en los procesos de medición manual, disminuye el costo de entrenamiento del personal que realiza la medición, disminuye el tiempo incurrido en realizar la medición, y asegura la repetibilidad de las medidas obtenidas.

Capítulo 7
Conclusiones y Trabajos Futuros

En este capítulo se presentan las conclusiones del trabajo presentado en esta tesis. En primer lugar se listan las contribuciones de esta tesis en el área de medición de tamaño funcional para sistemas generados a partir de modelos conceptuales en entornos MDA. Luego, se explica el trabajo que actualmente está siendo llevado a cabo y los trabajos futuros. Finalmente, se citan las publicaciones obtenidas durante el desarrollo de esta tesis.

7.1 Contribuciones

El trabajo presentado en esta tesis está relacionado principalmente a dos métodos: el método de medición COSMIC y el método de desarrollo de software OO-Method. Esta tesis corresponde a un procedimiento de medición que permite aplicar el método COSMIC a modelos conceptuales orientados a objeto para obtener el tamaño funcional exacto de las aplicaciones generadas en el entorno MDA OO-Method.

En este contexto, las principales contribuciones obtenidas de la realización de esta tesis se presentan a continuación:

1. Se ha realizado una revisión sistemática de procedimientos de medición que permiten obtener el tamaño funcional de aplicaciones a partir de sus modelos conceptuales. La revisión sistemática ha comprendido procedimientos de medición basados en cualquiera de las versiones de los métodos de medición estándares: IFPUG FPA, NESMA FPA, MK II FPA y COSMIC.

2. Se ha diseñado un procedimiento de medición de tamaño funcional para medir el tamaño funcional de aplicaciones generadas con OO-Method a partir de sus modelos conceptuales. Este procedimiento ha sido definido de acuerdo al método de medición COSMIC, ya que en contraste a otros FSM como IFPUG FPA, NESMA FPA o MK II FPA, COSMIC permite la medición del tamaño funcional de aplicaciones multi-capas (como las aplicaciones

modeladas con OO-Method). Se ha seleccionado la versión 3.0 del método COSMIC debido a que presenta mejoras sobre las versiones anteriores e incorpora una fase para definir la estrategia de la medición. El diseño de este procedimiento de medición incluye las reglas que permiten identificar los conceptos considerados en COSMIC para la medición del tamaño funcional del software en modelos conceptuales orientados a objetos; las reglas que evitan la contabilización duplicada de los elementos del modelo conceptual y las reglas que permiten obtener el tamaño funcional de los procesos funcionales, de las piezas de software que componen la aplicación OO-Method, y de la aplicación OO-Method completa.

3. Se ha definido una secuencia de pasos para facilitar la aplicación del procedimiento de medición al momento de realizar la medición manual de modelos conceptuales OO-Method. Este proceso de aplicación del procedimiento de medición OOmCFP considera las actividades definidas en cada fase de la versión 3.0 del método COSMIC (estrategia, correspondencias, y medición) y actividades para eliminar duplicidades y agregar los resultados para obtener el tamaño funcional de cada proceso funcional, de cada pieza de software que compone la aplicación, y de la aplicación completa.

4. Se ha implementado una herramienta que automatiza el procedimiento de medición OOmCFP. Esta herramienta realiza una medición completamente automática del tamaño funcional de aplicaciones generadas a partir de los modelos conceptuales OO-Method. La herramienta ha sido desarrollada teniendo en cuenta mecanismos para que la medición se realice de manera eficiente y se minimicen los costos a la hora de realizar modificaciones en la herramienta. Gracias a esta herramienta, el procedimiento de medición OOmCFP se puede utilizar de manera industrial.

7.2 Trabajos Actuales y Futuros

Esta tesis no es un trabajo cerrado. Aún es necesario seguir investigando en el área de la medición de tamaño funcional para aplicaciones generadas en entornos MDA.

Algunas actividades que se están realizando actualmente son:

- Evaluación de la precisión de las medidas obtenidas por el procedimiento de medición OOmCFP. Durante el último año se ha desarrollado un método para evaluar la precisión de las medidas de software teniendo en cuenta su repetibilidad y su reproducibilidad. Este método se ha aplicado a OOmCFP en un estudio piloto con resultados satisfactorios [Marin08]. Actualmente, se está diseñando un experimento controlado que permita evaluar la precisión de las medidas obtenidas por OOmCFP.

- Desarrollo de un marco de validación para procedimientos de medición basados en COSMIC. Si bien existen diversas formas de validar un procedimiento de medición, la utilización de cada una de ellas dependerá del contexto donde será utilizado el procedimiento. Actualmente se está desarrollando un marco de validación para procedimientos de medición basados en COSMIC que pueda ser utilizado por personas que posteriormente desarrollen procedimientos de medición basados en este estándar.

- Desarrollo de una nueva versión de la herramienta OOmCFP. En esta nueva versión se está incorporando un repositorio para almacenar los resultados de los proyectos medidos. Además, la herramienta se está modificando para aceptar archivos XMI como especificaciones de modelos conceptuales. De esta manera, la herramienta podría medir modelos conceptuales realizados en otras herramientas (como por ejemplo Poseidon, Rational, Eclipse UML2, etc), que correspondan a modelos OO-Method según un el perfil UML de OO-Method que se está desarrollando actualmente dentro del centro de investigación PROS.

Los trabajos futuros que se pretenden llevar a cabo inmediatamente son:

- Desarrollo de un modelo de predicción de costo basado en puntos de función para entornos MDA. Teniendo en cuenta que el tamaño funcional es utilizado para estimar el costo de las aplicaciones, en entornos MDA (donde las aplicaciones se generan automáticamente a partir de un modelo conceptual), los modelos de predicción de costo

deben ajustarse a las particularidades de las aplicaciones generadas en entornos MDA.

- Desarrollo de una herramienta que permita automatizar la predicción del costo de las aplicaciones generadas en entornos MDA y que reciba como entrada el archivo XML obtenido como resultado de la medición del tamaño funcional por la herramienta OOmCFP.

- Desarrollo de un modelo de predicción de defectos basado en puntos de función para entornos MDA. El tamaño funcional obtenido con el método de medición COSMIC está siendo utilizado para detectar defectos en los modelos de requisitos. Dado que las aplicaciones generadas en entornos MDA se generan automáticamente a partir de las especificaciones del modelo conceptual, es posible desarrollar un modelo para predecir defectos de las aplicaciones generadas a partir del tamaño funcional obtenido en sus modelos conceptuales.

- Desarrollo de una herramienta que permita automatizar la predicción de defectos a partir del archivo de XML generado como resultado por la herramienta OOmCFP.

7.3 Publicaciones Relacionadas

Para llevar a cabo el trabajo presentado en esta tesis ha sido necesario investigar en diferentes áreas de la ingeniería del software, entre las cuales se pueden destacar: métodos de modelado conceptual, métricas de calidad para modelos conceptuales, estándares de calidad de software, métodos de medición de software, métodos y estándares de validación de medidas, métodos de medición de tamaño funcional y herramientas de medición de tamaño funcional. Fruto de estas investigaciones se tienen 9 publicaciones. En este sentido, las publicaciones que respaldan el trabajo presentado en esta tesis son:

- **Beatriz Marín**, Giovanni Giachetti, Oscar Pastor. *Intercambio de Modelos UML y OO-Method*. X Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes Software (**IDEAS 2007**). Isla Margarita, Venezuela, Mayo 07-11, 2007. pp. 283–296.

- Giovanni Giachetti, **Beatriz Marín**, Nelly Condori-Fernández, Juan Carlos Molina. *Updating OO-Method Function Points*. 6th IEEE International Conference on the Quality of Information and Communications Technology (**QUATIC 2007**). Lisboa, Portugal, Septiembre 12-14, 2007. IEEE Computer Society Press, pp. 55-64.

- **Beatriz Marín**, Nelly Condori-Fernández, Oscar Pastor. *Calidad en Modelos Conceptuales: Un Análisis Multidimensional de Modelos Cuantitativos basados en la ISO 9126*. VII Conferencia Anual de la Asociación Española de Métricas de Sistemas Informáticos (**AEMES 2007**). Madrid, España, Octubre 01-02, 2007. Revista de Procesos y Métricas de las Tecnologías de la Información vol. 4, pp. 153-167.

- **Beatriz Marín**, Giovanni Giachetti, Oscar Pastor. *Una Herramienta Industrial para la Medición del Tamaño Funcional de Aplicaciones Desarrolladas en Entornos MDA*. XI Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes de Software (**IDEAS 2008**). Recife, Brasil, Febrero 11-15, 2008. pp. 357-362.

- **Beatriz Marín**, Nelly Condori-Fernández, Oscar Pastor, Alain Abran. *Measuring the Functional Size of Conceptual Models in an MDA Environment*. The 20th International Conference on Advanced Information Systems Engineering Forum (**CAiSE Forum 2008**). Montpellier, Francia, Junio 18-20, 2008. pp. 33-36.

- **Beatriz Marín**, Oscar Pastor, Giovanni Giachetti. *Automating the Measurement of Functional Size of Conceptual Models in a MDA Environment*. The 9th International Conference on Product Focused Software Process Improvement (**PROFES 2008**). Roma, Italia, Junio 23-25, 2008. LNCS 5089, Springer 2008, pp. 215-229.

- **Beatriz Marín**, Nelly Condori-Fernández, Oscar Pastor. *Towards a Method for Evaluating the Precision of Software Measures*. The 8th International Conference on Quality Software (**QSIC 2008**). Oxford, Gran Bretaña, Agosto 12-13, 2008. IEEE Computer Society Press, pp. 305-310.

- **Beatriz Marín**, Nelly Condori-Fernández, Oscar Pastor. *Design of a Functional Size Measurement Procedure for a Model-Driven Software Development Method*. Accepted in the 3rd Workshop on Quality in Modeling (**MODELS Workshops 2008**). Toulouse, Francia, Septiembre 28-30, 2008.

- **Beatriz Marín**, Giovanni Giachetti, Oscar Pastor. *Measurement of Functional Size in Conceptual Models: A Survey of Measurement Procedures based on COSMIC*. Accepted in the 3rd International Conference on Software Process and Product Measurement (**MENSURA 2008**). Munich, Alemania, Noviembre 17-19, 2008. LNCS, Springer 2008.

Referencias

[Abrahao01] Abrahao, S.; Pastor, O., *Estimating the Applications Functional Size from Object-Oriented Conceptual Models.* In: International Function Point User Group Annual Conference (IFPUG'01), Las Vegas, USA, 2001.

[Abrahao02] Abrahao, S.; Olsina, L.; Pastor, O., *A Methodology for Evaluating Quality and Functional Size of Operative WebApps*. In: 2nd International Workshop on Web Oriented Software Technology (IWWOST 2002), Malaga, Spain, June 10, 2002.

[Abrahao03] Abrahao, S.; Pastor, O., *Measuring the functional size of web applications.* In: International Journal of Web Engineering and Technology (1:1), 2003, pp. 5-16.

[Abrahao04] Abrahao, S., PhD Thesis: *On the Functional Size Measurement of Object-Oriented Conceptual Schemas: Design and Evaluation Issues*, Universidad Politécnica de Valencia, España, Julio 2004.

[Abrahao04a] Abrahao, S.; Poels, G.; Pastor, O., *Assessing the reproducibility and accuracy of functional size measurement methods through experimentation.* In: Proceedings of the International Symposium on Empirical Software Engineering (ISESE 2004), August 19-20, 2004, pp. 189-198.

[Abrahao04b] Abrahao, S.; Poels, G.; Pastor, O., *Evaluating a functional size measurement method for Web applications: an empirical analysis*. In: 10th International Symposium on Software Metrics (METRICS 2004), September 14-16, 2004, pp.358-369.

[Abrahao06] Abrahao, S.; Poels, G., *Further Analysis on the Evaluation of a Size Measure for Web Applications*. In: 4th Latin American Web Congress (LA-Web 2006), October 2006, pp. 230-240.

[Abrahao06a] Abrahao, S.; Poels, G.; Pastor, O., *A Functional Size Measurement Method for Object-Oriented Conceptual Schemas: Design and Evaluation Issues*. In: Journal of Software and System Modeling, Vol(5) Nro(1), 2006.

[Abrahao07] Abrahao, S.; Poels, G., *Experimental Evaluation of an Object-Oriented Function Point Measurement Procedure*. In:

Journal of Information and Software Technology, Vol(49) Nro(4), 2007, pp. 366-380.

[Abrahao07a] Abrahao, S.; Mendes, E.; Gomez, J.; Insfrán, E., *A Model-Driven Measurement Procedure for Sizing Web Applications: Design, Automation and Validation*. In: ACM/IEEE 10th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2007), Nashville, USA, 2007, pp. 467-481.

[Abran94] Abran, A.; Pierre, N., *Function Points: A Study of Their Measurement Processes and Scale Transformations*, Journal Systems and Software 25(2), 1994, pp. 171-184.

[Abran99] Abran, A.; Jacquet, J.P., *A Structured Analysis of the New ISO Standard of Functional Size Measurement – Definition of Concepts.* In: 4th IEEE International Symposium and Forum of Software Engineering Standards, 1999, pp. 230-241.

[Abran99a] Abran, A.; Desharnais, J.; Oligny, S.; St-Pierre, D.; Symons, C., *COSMIC-FFP Measurement Manual, version 2.0.* In: Software Engineering Management, Research Laboratory, Université du Québec à Montréal, Montreal, Canada, 1999.

[Abran00] Abran, A.; Oligny, S.; Symons, C.; Fagg, P.; Desharnais, J.; St-Pierre, D.; Tran-Cao, D., *Caso de estudio Rice Cooker.* In: http://www.gelog.etsmtl.ca/cosmic-ffp/casestudies/version2.1ricecookercasestudy.pdf

[Abran01] Abran, A.; Desharnais, J.; Oligny, S.; St-Pierre, D.; Symons, C., *COSMIC-FFP Measurement Manual, Version 2.1.* In: The Common Software Measurement International Consortium web site, May*o,* 2001.

[Abran03] Abran, A.; Desharnais, J.M.; Oligny, S.; St-Pierre, D.; Symons, C., *COSMIC-FFP Measurement Manual – Version 2.2, The COSMIC Implementation Guide for ISO/IEC 19761:2003*. In: The Common Software Measurement International Consortium web site, 2003.

[Abran07] Abran, A.; Desharnais, J.; Lesterhuis, A.; Londeix, B.; Meli, R.; Morris, P.; Oligny, S.; O'Neil, M.; Rollo, T.; Rule,G.; Santillo, L.; Symons, C.; Toivonen, H., *The COSMIC Functional Size Measurement Method, version 3.0.* In GELOG web site www.gelog.etsmtl.ca, 2007.

[Albrecht79] Albrecht, A.J., *Measuring Application Development*

Productivity. In: Proceedings IBM Applications Development Symposium*,* Monterey, California, October 14-17, 1979.

[Antoniol03] Antoniol, G.; Fiutem, R.; Lokan, C., *Object-Oriented Function Points: An Empirical Validation*. In: Journal of Empirical Software Engineering, Vol(8) Nro(3), 2003, pp. 225-254.

[Azzouz04] Azzouz, S.; Abran, A., *A Proposed Measurement Role in the Rational Unified Process and its Implementation with ISO 19761: COSMIC FFP*. In: Software Measurement European Forum (SMEF 2004), Rome, Italy, 2004.

[Basili88] Basili, V. and Rombach, H.; *The TAME Project: Towards Improvement Oriented Software Environments*, IEEE Transactions on Software Engineering, 1988, pp. 758-773.

[Berg05] van den Berg, K.G.; Dekkers, T.; Oudshoorn, R., *Functional Size Measurement applied to UML-based user requirements.* In: Proceedings of the 2nd Software Measurement European Forum (SMEF2005), Rome, Italy, March 16-18, 2005, pp. 69-80.

[Bertolami03] Bertolami, M.; Oliveros, A., *Análisis de Puntos Función en la Elicitación de Requerimientos*. In: Proceedings of the 6th Workshop on Requirements Engineering (WER 2003), 2003, pp. 32-47.

[Bertolami05] Bertolami, M.; Oliveros, A., *Estimate of the Functional Size in the Requirements Elicitation*. In: Journal in Computer Science & Technology, 2005.

[Bertolami06] Bertolami, M.; Oliveros, A., *SFP: Un Procedimento de Estimación de Puntos de Función de Escenarios*. In: Proceedings of the 9th Workshop on Requirements Engineering (WER 2006), Rio de Janeiro, Brasil, July 13-14, 2006, pp. 101-108.

[Bevo99] Bévo, V.; Lévesque, G.; Abran, A., *Application de la méthode FFP à partir d'une spécification selon la notation UML: compte rendu des premiers essais d'application et questions*. In: Proceedings of the 9th International Workshop Software Measurement, Lac Supérieur, Canada, 1999, pp. 230-242.

[Caldiera98] Caldiera, G.; Antoniol, G.; Fiutem, R.; Lokan, C., *Definition and Experimental Evaluation of Function Points for Object-Oriented Systems*. In: 5th International Symposium on

Software Metrics (METRICS 1998), November 20-21, 1998, pp. 167-178.

[Cantone04] Cantone, G.; Pace, D.; Calavaro, G., *Applying function point to unified modeling language: conversion model and pilot study*. In: 10th International Symposium on Software Metrics (METRICS 2004), September 14-16, 2004, pp. 280-291.

[CARE] Sitio Web de CARE Technologies, www.care-t.com

[Condori04] Condori-Fernandez, N.; Abrahao, S.; Pastor, O., *Towards a Functional Size Measure for Object-Oriented Systems from Requirements Specifications*. In: Proceedings of 4th International Conference on Quality Software (QSIC 2004), IEEE Computer Society, Germany, 2004, pp. 94-101.

[Condori06] Condori-Fernández, N.; Pastor, O., *Verifying the Construction of a Software Model from a Requirements Model*. In: Proceedings of the 9th Workshop on Requirements Engineering (WER 2006), Rio de Janeiro, Brasil, July 13-14, 2006.

[Condori06a] Condori-Fernández, N.; Pastor, O., *Evaluating the Productivity and Reproducibility of a Measurement Procedure*. In: Proceedings of the ER Workshops (ER Wrokshops 2006), pp. 352-361.

[Condori06b] Condori-Fernández, N.; Pastor, O., *An Empirical Study on the Likelihood of Adoption in Practice of a Size Measurement Procedure for Requirements Specification*. In: Proceedings of 4th International Conference on Quality Software (QSIC 2006), IEEE Computer Society, 2006, pp. 133-140.

[Condori06c] Condori-Fernández, N.; Pastor, O., *Re-Assessing the Intention to Use a Measurement Procedure based on COSMIC-FFP*. In: 16th International Workshop on Software Measurement (IWSM 2006), Berlin, Germany, November 2-4, 2006, pp. 63-71.

[Condori07] Condori-Fernández, N., *Un procedimiento de medición de tamaño funcional a partir de especificaciones de requisitos*, Doctoral thesis, Univ. Politécnica de Valencia, España, 2007.

[Condori07a]Condori-Fernández, N.; Abrahão, S.; Pastor, O., *On the Estimation of Software Functional Size from Requirements Specifications*. In: Journal of Computer Science and Technology (JCST), Springer, Vol(22) Nro(3), May 2007, pp. 358-370.

[Conte86] Conte, S.D., Dunsmore, H.E., and Shen, V.Y., Software Engineering Metrics and Models, 1986.

[Dedene94] Dedene, G.; Snoeck, M., *M.E.R.O.DE.: A Model-driven Entity-Relationship Object-oriented Development Method*. In: ACM SIGSOFT Software Engineering Notes, 1994, 19(3): 51-61.

[DeMarco82] DeMarco, T.; *Controlling Software Projects*; Ed. Prentice Hall; 1982.

[Diab01] Diab, H.; Frappier, M.; St-Denis, R., *Formalizing COSMIC-FFP Using ROOM*. In: ACS/IEEE International Conference on Computer Systems and Applications (AICCSA 2001), June 25-29, 2001, pp. 312-318.

[Diab01a] Diab, H.; Frappier, M.; St-Denis, R, *A formal definition of COSMICFFP for automated measurement of ROOM specifications*. In: Proceeding of the 4th European Conference on Software Measurement and ICT Control (FESMA 2001), Heidelberg, Germany, 2001, pp. 185–196.

[Diab05] Diab, H.; Koukane, F.; Frappier, M.; St-Denis, R., *µcROSE: Automated Measurement of COS-MIC-FFP for Rational Rose Real Time*. In: Journal of Information and Software Technology, Vol(47) Nro(3), 2005, pp. 151-166.

[Diaz05] Diaz, I.; Sanchez, J.; Pastor, O., *Metamorfosis: Un marco para el análisis de requisitos funcionales.* In: Proceedings of the Workshop on Requirements Engineering (WER 2005), Porto, Portugal, 2005, pp. 233-244.

[Fenton97] Fenton, N.; Pfleeger, S., *Software Metrics. A Rigorous & Practical Approach*, 1997.

[Fetcke97] Fetcke, T.; Abran, A.; Nguyen, T., *Mapping the OO-Jacobson Approach into Function Point Analysis*. In: Proceedings of 23th Technology of Object-Oriented Languages and Systems (TOOLS 1997), July 28-August 01, 1997, pp. 192-202.

[Fons03] Fons, J.; Pelechano, V.; Albert, M.; Pastor, O., *Development of Web Applications from Web Enhanced Conceptual Schemas*. In: Proceedings of the ER conference (ER 2003), 2003, pp. 232-245.

[Fraternali06] Fraternali, P.; Tisi, M.; Bongio, A., *Automating Function Point Analysis with Model Driven Development*. In:

Proceedings of the 2006 conference of the Center for Advanced Studies on Collaborative research (CASCON 2006), October 2006, ACM.

[Giachetti07] Giachetti, G.; Marín, B.; Condori-Fernández, N.; Molina, J.C., *Updating OO-Method Function Points*. In: 6th IEEE International Conference on the Quality of Information and Communications Technology (QUATIC 2007), Lisboa, Portugal, September 12-14, 2007, pp. 55-64.

[Gómez98] Gómez, J.; Insfrán, E.; Pelechano, V.; Pastor, O., *The Execution Model: a component-based architecture to generate software components from conceptual models*. In: Workshop on Component-based Information Systems Engineering, Pisa, Italia, June 1998.

[Gramantieri97] Gramantieri, F.; Lamma, E.; Riguzzi, F.; Mello, P., A System for Measuring Function Points from Specifications, 1997.

[Grau07a] Grau, G.; Franch, X., *Using the PRiM method to Evaluate Requirements Model with COSMIC-FFP.* In: International Conference on Software Process and Product Measurement (IWSM-MENSURA 2007), Mallorca, Spain, November 2007.

[Grau07b] Grau, G., Franch, X.: *ReeF: Defining a Customizable Reengineering Framework*. In: Proceedings of the 19th International Conference on Advanced Information Systems Engineering(CAiSE 2007), Springer, LNCS 4495, 2007, pp. 485-500.

[Habela05] Habela, P.; Glowacki, E.; Serafinski, T.; Subieta, K., *Adapting Use Case Model for COSMIC-FFP Based Measurement*. In: Proceedings of the 15th International Workshop on Software Measurement (IWSM 2005), Montreal, Canada, 2005, pp. 195-207.

[Harput05] Harput, V.; Kaindl, H.; Kramer, S., Extending Function Point Analysis to Object-Oriented Requirements Specifications. In: 11th International Symposium on Software Metrics (METRICS 2005), Vienna, Austria, September 19-22, 2005.

[Hericko06] Hericko, M.; Rozman, I.; Zivkovic, A., *A Formal Representation on Functional Size Measurement Methods*. In: Journal of Systems and Software, Vol(79) Nro(9), 2006, pp.1341-1358.

[IFPUG] International Function Point Users Group Web Site, www.ifpug.org

[IFPUG99] IFPUG, *Function Point Counting Practices Manual Release 4.1*, Westerville, Ohio, USA, 1999.

[Insfran02] Insfrán, E.; Pastor, O.; Wieringa, R., *Requirements Engineering-Based Conceptual Modelling*. In: Journal Requirements Engineering (RE), Springer, 2002, pp. 61-72.

[ISO98] ISO/IEC 14143-1, *Information Technology – Software Measurement – Functional Size Measurement – Part 1: Definition of Concepts*, 1998.

[ISO02] ISO/IEC 20968, *Software Engineering* – Mk II Function Point Analysis – Counting Practices Manual, 2002.

[ISO02a] ISO/IEC 14143-4, *Information Technology – Software Measurement – Functional Size Measurement – Part 4: Reference Model*, 2002.

[ISO03] ISO/IEC 19761, *Software Engineering – CFF – A Functional Size Measurement Method*, 2003.

[ISO03a] ISO/IEC 20926, *Software Engineering – IFPUG 4.1 Unadjusted Functional Size Measurement Method – Counting Practices Manual*, 2003.

[ISO04] ISO/IEC 24570, *Software Engineering* – NESMA Functional Size Measurement Method version 2.1 – Definitions and Counting Guidelines for the application of Function Point Analysis, 2004.

[ISO04a] ISO, *International vocabulary of basic and general terms in metrology (VIM)*, 2004.

[Jacobson92] Jacobson, Ivar; et al. Object-Oriented Software Engineering. Addison-Wesley & ACM, 1992.

[Jacquet97] Jacquet, J.P.; Abran, A., *From Software Metrics to Software Measurement Methods: A Process Model.* In: 3rd International Standard Symposium and Forum on Software Engineering Standards, ISESS 1997, Walnut Creek, USA, 1997.

[Jenner02] Jenner, M.S., *Automation of Counting of Functional Size Using COSMIC-FFP in UML*. In: Proceedings of the 12th

International Workshop Software Measurement (IWSM 2002), 2002, pp. 43-51.

[Khan01] Khan, K. S.; Ter, R. G.; Glanville, J.; Sowden, A. J.; Kleijnen, Jo, *Undertaking Systematic Review of Research on Effectiveness. CRD's Guidance for those Carrying Out or Commissioning Reviews.* CRD Report Number 4 (2nd Edition), NHS Centre for Reviews and Dissemination, University of York, ISBN 1 900640 20 1, 2001.

[Kitchenham97] Kitchenham, B., *Counterpoint: The Problem with Function Points*, IEEE Software Status Report 14(2), 1997, pp. 29-31.

[Kruchten00] Kruchten, P.; *The Rational Unified Process, an Introduction*, Addison Wesley, March 2000.

[Kusumoto00] Kusumoto, S.; Inoue, K.; Kasimoto, T.; Suzuki, A.; Yuura, K.; Tsuda, M., *Function Point Measurement for Object-Oriented Requirements Specification*. In: 24th International Computer Software and Applications Conference (COMPSAC 2000), October 25-27, 2000, pp.543-548.

[Lehne97] Lehne, A., *Experience Report: Function Points Counting of Object-Oriented Analysis and Design based on the OOram method.* In: Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'97), Atlanta, Georgia, October 1997.

[Levesque08] Levesque, G.; Bevo, V.; Tran Cao, D., *Estimating software size with UML models*. In: Proceedings of the C^3S^2E Conference (C3S2E 2008), Montreal, Quebec, Canada, May 12-13, 2008, ACM, pp. 81-87.

[Marin07] Marín, B; Giachetti, G.; Pastor, O., *Intercambio de Modelos UML y OO-Method*. In: X Workshop Iberoamericano de Ingenieria de Requisitos y Ambientes Software (IDEAS), Isla Margarita, Venezuela, 2007, pp. 283 – 296.

[Marin08] Marín, B.; Condori-Fernández, N.; Pastor, O., *Towards a Method for Evaluating the Precision of Software Measures*. In: The 8th International Conference on Quality Software (QSIC 2008). Oxford, Gran Bretaña, Agosto 12-13, 2008. IEEE Computer Society Press, pp. 305-310.

[Meli00] Meli, R.; Abran, A.; Ho Vinh, T.; Oligny, S., *On the Applicability of COSMIC-FFP for Measuring Software Throughout its Life Cycle*. In: Proceedings of the 11th European Software Control and Metrics Conference, Munich, 2000.

[Mellor02] Mellor, S.; Balcer, J., *Executable UML: A Foundation for Model-Driven Architecture*, Addison Wesley, 2002.

[Mendes96] Mendes, O.; Abran, A.; Bourque, P., *FP Tool Classification Framework and Market Survey*. In: SEMRL, IFPUG Fall Conference, Dallas, 1996.

[Miller03] Miller, J.; Mukerji, J., *MDA Guide Version 1.0.1*, 2003.

[Molina03] Molina, P., *Especificación de interfaz de usuario: De los requisitos a la generación automática*. Doctoral thesis, Universidad Politécnica de Valencia, Valencia, España, 2003.

[Nagano03] Nagano, S.; Ajisaka, T., *Functional Metrics using COSMIC FFP for Object-Oriented Real-Time Systems*. In: Proceedings of the 13th International Workshop on Software Measurement (IWSM 2003), Montreal, Canada, September 23-25, 2003.

[Pastor92] Pastor, O.; Hayes, F.; Bear, S., *OASIS: An Object-Oriented Specification Language*. In: 4th International Conference on Advanced Information Systems Engineering (CAiSE 1992), Manchester, UK, 1992, pp. 348-363.

[Pastor01] Pastor, O.; Gómez, J.; Insfrán E.; Pelechano, V., *The OO-Method Approach for Information Systems Modelling: From Object-Oriented Conceptual Modeling to Automated Programming.* In: Information Systems, vol.26, 2001, pp. 507-534.

[Pastor04] Pastor, O.; Molina, J.C.; Iborra, E., *Automated production of fully functional applications with OlivaNova Model Execution*. In: ERCIM News No.57, Abril 2004.

[Pastor07] Pastor, O.; Molina, J.C., *Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling*, Springer, 2007.

[Poels00] Poels, G.; Dedene, G., *Distance-based software measurement: necessary and sufficient properties for software measures*. In: Information and Software Technology, Vol(42) Nro (1), 2000, pp. 35-46.

[Poels02] Poels, G., *A Functional Size Measurement Method for Event-Based Object Oriented Enterprise Models*. In: ICEIS 2002, pp. 667-675.

[Poels03a] Poels G., *Definition and Validation of a COSMIC-FFP Functional Size Measure for Object-Oriented Systems*. In: Proceedings of the 7th International ECOOP Workshop QAOOSE (QAOOSE 2003). Darmstadt, 2003, pp. 1-6.

[Poels03b] Poels, G., *Functional Size Measurement of Multi-Layer Object-Oriented Conceptual Models*. In: Proceedings of 9th International Object-Oriented Information Systems Conference, Geneva, Switzerland, 2003, pp. 334-345.

[Ram00] Ram, D.J.; Raju, S., *Object Oriented Design Function Points*. In: 1st Asia-Pacific Conference on Quality Software, 2000, IEEE Press.

[RETO] Web Site RETO Requirements Tool, http://reto.dsic.upv.es/reto/

[Saito95] Saito, M.; Onari, N.; Yuura, I.; Kameda, T., *Visualizing Tool for Required Specifications*. In: The Hitachi Hyoron, Vol(77) Nro(12), 1995, pp. 15-18.

[Schwabe95] Schwabe, D.; Rossi, G., *The Object-Oriented Hypermedia Design Model*. In: Communications of the ACM Vol(38) Nro(8), 1995, pp. 45–46.

[Schmidt06] Schmidt, D., *Model Driven Engineering*. In: IEEE Computer Society, Vol(39) Nro(2), 2006, pp. 25-31.

[Selic94] Selic, B.; Gullekson, G.; Ward, P.T., *Real-time Object Oriented Modelling*, Wiley, 1994.

[Selic07] Selic, B., *A Systematic Approach to Domain-Specific Language Design Using UML*. In: Proceedings of the 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC), 2007, pp. 2–9.

[Shlaer92] Shlaer, S.; Mellor, S., Object Lifecycles, Modelling the World in States, Yourdon Press, Prentice-Hall, 1992.

[Shoval88] Shoval, P., *ADISSA: Architectural Design of Information Systems based on Structured Analysis*. In: Information Systems, Vol(13), 1988, pp. 193–210.

[Shoval97] Shoval, P.; Feldman, O., A Combination of the Mk-II Function Points Software Estimation Method with the ADISSA Methodology for Systems Analysis and Design. In: Journal of Information and Software Technology, Vol(39) Nro(13), 1997, pp. 855-865.

[Tavares02] Tavares, H.; Carvalho, A.; Castro, J., *Medicao de Pontos por Funcao a partir da Especificacao de Requisitos*. In: Workshop on Requirements Engieneering, Universidad Politécnica de Valencia, Spain, November 2002, pp. 278-298.

[TranCao02] Tran-Cao, D.; Levesque, G.; Abran, A., *Measuring Software Functional Size: Towards an Effective Measurement of Complexity*. In: Proceedings of the International Conference on Software Maintenance, 2002, IEEE Computer Society, pp. 370-376.

[Uemura99] Uemura, T.; Kusumoto, S.; Inoue, K., *Function Point Measurement Tool for UML Design Specification*. In: 5th International Symposium on Software Metrics (METRICS 1999), Florida, USA, November 4-6, 1999, pp. 62-69.

[Uemura01] Uemura, T.; Kusumoto, S.; Inoue, K., *Function Point Analysis using Design Specifications based on the Unified Modelling Language*. In: Journal of Software Maintenance and Evolution: Research and Practice, Vol(13) Nro(4), 2001, pp. 223-243.

[UKSMA98] United Kingdom Software Metrics Association (UKSMA), *MKII Function Point Analysis Counting Practices Manual. Version 1.3.1*, 1998.

[Zivkovic05] Zivkovic, A.; Rozman, I.; Hericko, M., *Automated Software Size Estimation based on Function Points using UML Models*. In: Journal of Information & Software Technology, Vol(47), 2005, pp. 881–890.

Anexo A
Formulario Extracción de Datos

1) Información de la Búsqueda
    - Fecha de extracción de datos
    - Términos utilizados en la búsqueda
    - Base de datos donde se encontró

2) Información General
    - Título
    - Autores
    - Año publicación
    - Lugar de publicación

3) Información Específica
    - Resumen
    - Método de medición y versión
    - Artefacto de software
    - Diseño del procedimiento
    - Validación del procedimiento
    - Herramienta

4) Notas sobre la Calidad del Estudio

5) Otras Notas

Anexo B
Estudios Seleccionados

IEEE Xplore

1. Abrahao, S.; Poels, G.; Pastor, O., *Assessing the reproducibility and accuracy of functional size measurement methods through experimentation.* In: Proceedings of the International Symposium on Empirical Software Engineering (ISESE 2004), August 19-20, 2004, pp. 189-198.

2. Abrahao, S.; Poels, G.; Pastor, O., *Evaluating a functional size measurement method for Web applications: an empirical analysis*. In: 10th International Symposium on Software Metrics (METRICS 2004), September 14-16, 2004, pp.358-369.

3. Abrahao, S.; Poels, G., *Further Analysis on the Evaluation of a Size Measure for Web Applications*. In: 4th Latin American Web Congress (LA-Web 2006), October 2006, pp. 230-240.

4. Caldiera, G.; Antoniol, G.; Fiutem, R.; Lokan, C., *Definition and Experimental Evaluation of Function Points for Object-Oriented Systems*. In: 5th International Symposium on Software Metrics (METRICS 1998), November 20-21, 1998, pp. 167-178.

5. Cantone, G.; Pace, D.; Calavaro, G., *Applying function point to unified modeling language: conversion model and pilot study*. In: 10th International Symposium on Software Metrics (METRICS 2004), September 14-16, 2004, pp. 280-291.

6. Condori-Fernandez, N.; Abrahao, S.; Pastor, O., *Towards a Functional Size Measure for Object-Oriented Systems from Requirements Specifications*. In: 4th International Conference on Quality Software (QSIC 2004), IEEE Computer Society, Germany, 2004, pp. 94-101.

7. Diab, H.; Frappier, M.; St-Denis, R., *Formalizing COSMIC-FFP Using ROOM*. In: ACS/IEEE International Conference on Computer Systems and Applications (AICCSA 2001), June 25-29, 2001, pp. 312-318.

8. Fetcke, T.; Abran, A.; Nguyen, T., *Mapping the OO-Jacobson*

Approach into Function Point Analysis. In: 23th Technology of Object-Oriented Languages and Systems (TOOLS 1997), July 28-August 01, 1997, pp. 192-202.

9. Giachetti, G.; Marín, B.; Condori-Fernández, N.; Molina, J.C., *Updating OO-Method Function Points*. In: 6th IEEE International Conference on the Quality of Information and Communications Technology (QUATIC 2007), Lisboa, Portugal, September 12-14, 2007, pp. 55-64.

10. Kusumoto, S.; Inoue, K.; Kasimoto, T.; Suzuki, A.; Yuura, K.; Tsuda, M., *Function Point Measurement for Object-Oriented Requirements Specification*. In: 24th International Computer Software and Applications Conference (COMPSAC 2000), October 25-27, 2000, pp.543-548.

11. Uemura, T.; Kusumoto, S.; Inoue, K., *Function Point Measurement Tool for UML Design Specification*. In: 6th International Symposium on Software Metrics (METRICS 1999), Florida, USA, November 4-6, 1999, pp. 62-69.

ACM Digital Library

1. Fraternali, P.; Tisi, M.; Bongio, A., *Automating Function Point Analysis with Model Driven Development*. In: Proceedings of the 2006 conference of the Center for Advanced Studies on Collaborative research (CASCON 2006), October 2006, ACM.

2. Levesque, G.; Bevo, V.; Tran Cao, D., *Estimating software size with UML models*. In: Proceedings of the C^3S^2E Conference (C3S2E 2008), Montreal, Quebec, Canada, May 12-13, 2008, ACM, pp. 81-87.

Springer Link

1. Abrahao, S.; Poels, G.; Pastor, O., *A Functional Size Measurement Method for Object-Oriented Conceptual Schemas: Design and Evaluation Issues*. In: Journal of Software and System Modeling, Vol(5) Nro(1), 2006.

2. Abrahao, S.; Mendes, E.; Gomez, J.; Insfrán, E., *A Model-Driven Measurement Procedure for Sizing Web Applications: Design, Automation and Validation*. In: ACM/IEEE 10th International

Conference on Model Driven Engineering Languages and Systems (MoDELS 2007), Nashville, USA, 2007, pp. 467-481.

3. Condori-Fernández, N.; Pastor, O., *Evaluating the Productivity and Reproducibility of a Measurement Procedure*. In: Proceedings of the ER Workshops (ER Wrokshops 2006), pp. 352-361.

4. Grau, G.; Franch, X., *Using the PRiM method to Evaluate Requirements Model with COSMIC-FFP.* In: Proceedings of the International Conference on Software Process and Product Measurement (IWSM-MENSURA 2007), Mallorca, Spain, November 2007.

Science Direct

1. Abrahao, S.; Poels, G., *Experimental Evaluation of an Object-Oriented Function Point Measurement Procedure*. In: Journal of Information and Software Technology, Vol(49) Nro(4), 2007, pp. 366-380.

2. Hericko, M.; Rozman, I.; Zivkovic, A., *A Formal Representation on Functional Size Measurement Methods*. In: Journal of Systems and Software, Vol(79) Nro(9), 2006, pp.1341-1358.

Google Scholar

1. Abrahao, S.; Olsina, L.; Pastor, O., *A Methodology for Evaluating Quality and Functional Size of Operative WebApps*. In: 2nd International Workshop on Web Oriented Software Technology (IWWOST 2002), Malaga, Spain, June 10, 2002.

2. Antoniol, G.; Fiutem, R.; Lokan, C., *Object-Oriented Function Points: An Empirical Validation*. In: Journal of Empirical Software Engineering, Vol(8) Nro(3), 2003, pp. 225-254.

3. Azzouz, S.; Abran, A., *A Proposed Measurement Role in the Rational Unified Process and its Implementation with ISO 19761: COSMIC FFP*. In: Software Measurement European Forum (SMEF 2004), Rome, Italy, 2004.

4. Bertolami, M.; Oliveros, A., *Análisis de Puntos Función en la Elicitación de Requerimientos*. In: 6th Workshop on Requirements Engineering (WER 2003), 2003, pp. 32-47.

5. Bertolami, M.; Oliveros, A., *Estimate of the Functional Size in the Requirements Elicitation*. In: Journal in Computer Science & Technology, 2005.

6. Bertolami, M.; Oliveros, A., *SFP: Un Procedimento de Estimación de Puntos de Función de Escenarios*. In: 9th Workshop on Requirements Engineering (WER 2006), Rio de Janeiro, Brasil, July 13-14, 2006, pp. 101-108.

7. Bévo, V.; Lévesque, G.; Abran, A., *Application de la méthode FFP à partir d'une spécification selon la notation UML: compte rendu des premiers essais d'application et questions*. In: 9th International Workshop Software Measurement (IWSM 1999), Lac Supérieur, Canada, 1999, pp. 230-242.

8. Condori-Fernández, N.; Pastor, O., *Verifying the Construction of a Software Model from a Requirements Model*. In: 9th Workshop on Requirements Engineering (WER 2006), Rio de Janeiro, Brasil, July 13-14, 2006.

9. Condori-Fernández, N.; Pastor, O., *Re-Assessing the Intention to Use a Measurement Procedure based on COSMIC-FFP*. In: 16th International Workshop on Software Measurement (IWSM 2006), Berlin, Germany, November 2-4, 2006, pp. 63-71.

10. Diab, H.; Frappier, M.; St-Denis, R, *A formal definition of COSMICFFP for automated measurement of ROOM specifications*. In: 4th European Conference on Software Measurement and ICT Control (FESMA 2001), Heidelberg, Germany, 2001, pp. 185–196.

11. Diab, H.; Koukane, F.; Frappier, M.; St-Denis, R., *µcROSE: Automated Measurement of COS-MIC-FFP for Rational Rose Real Time*. In: Journal of Information and Software Technology, Vol(47) Nro(3), 2005, pp. 151-166.

12. Gramantieri, F.; Lamma, E.; Riguzzi, F.; Mello, P., A System for Measuring Function Points from Specifications, 1997.

13. Habela, P.; Glowacki, E.; Serafinski, T.; Subieta, K., *Adapting Use Case Model for COSMIC-FFP Based Measurement*. In: 15th International Workshop on Software Measurement (IWSM 2005), Montreal, Canada, 2005, pp. 195-207.

14. Harput, V.; Kaindl, H.; Kramer, S., Extending Function Point Analysis to Object-Oriented Requirements Specifications. In: 11th

International Symposium on Software Metrics (METRICS 2005), Vienna, Austria, September 19-22, 2005.

15. Nagano, S.; Ajisaka, T., *Functional Metrics using COSMIC FFP for Object-Oriented Real-Time Systems*. In: 13th International Workshop on Software Measurement (IWSM 2003), Montreal, Canada, September 23-25, 2003.

16. Poels G., *Definition and Validation of a COSMIC-FFP Functional Size Measure for Object-Oriented Systems*. In: 7th International ECOOP Workshop QAOOSE (QAOOSE 2003). Darmstadt, 2003, pp. 1-6.

17. Poels, G., *Functional Size Measurement of Multi-Layer Object-Oriented Conceptual Models*. In: Proceedings of 9th International Object-Oriented Information Systems Conference, Geneva, Switzerland, 2003, pp. 334-345.

18. Ram, D.J.; Raju, S., *Object Oriented Design Function Points*. In: 1st Asia-Pacific Conference on Quality Software, 2000, IEEE Press.

19. Shoval, P.; Feldman, O., *A Combination of the Mk-II Function Points Software Estimation Method with the ADISSA Methodology for Systems Analysis and Design*. In: Journal of Information and Software Technology, Vol(39) Nro(13), 1997, pp. 855-865.

20. Tavares, H.; Carvalho, A.; Castro, J., *Medicao de Pontos por Funcao a partir da Especificacao de Requisitos*. In: Workshop on Requirements Engieneering, Universidad Politécnica de Valencia, Spain, November 2002, pp. 278-298.

21. Uemura, T.; Kusumoto, S.; Inoue, K., *Function Point Analysis using Design Specifications based on the Unified Modelling Language*. In: Journal of Software Maintenance and Evolution: Research and Practice, Vol(13) Nro(4), 2001, pp. 223-243.

22. van den Berg, K.G.; Dekkers, T.; Oudshoorn, R., *Functional Size Measurement applied to UML-based user requirements.* In: 2nd Software Measurement European Forum (SMEF2005), Rome, Italy, March 16-18, 2005, pp. 69-80.

23. Zivkovic, A.; Rozman, I.; Hericko, M., *Automated Software Size Estimation based on Function Points using UML Models*. In: Journal of Information & Software Technology, Vol(47), 2005, pp. 881–890.


30
23
13
12
11
10