Agradezco a mi familia, que me apoyo en todo momento,
y a mis amigos que me acompañaron durante la carrera.

# Índice

# Tabla de figuras

# 1 Introducción

El desarrollo de software no es y no ha sido nunca una tarea fácil. Un claro indicio de esto es la numerosa cantidad de propuestas metodológicas para su desarrollo, la cantidad de proyectos retrasados o sin poder concretarse. Cada día las exigencias al software son más, y si se quiere empezar a empezar a construir los cimientos para el futuro, es preciso y necesario que se construya software de calidad. Este es el objetivo de fondo de este trabajo: aumentar la calidad en el software, generando mayor valor en el y por otro lado facilitar la tarea del ingeniero de software a la hora de hacer seguimiento de proyecto o evaluación de calidad de productos.

La generación de software es un proceso complejo, que involucra muchos participantes, metodologías, técnicas y herramientas de software. La ingeniería de software esta constantemente intentando dar con mejores maneras de desarrollar software en un contexto muy cambiante. Al igual que en otras ramas ingenieriles, las propuestas metodológicas son fundamentales a la hora de manejar proyectos muy grandes. La metodología en si puede ser tan importante como el soporte que tenga mediante alguna herramienta que facilite su implementación. Las herramientas que dan soporte a las actividades a realizar por un ingeniero de software son llamadas "herramientas CASE" (Computer Aided Software Engineering), y tienen los siguientes objetivos:

1. Mejorar la productividad en el desarrollo y mantenimiento del software.
2. Aumentar la calidad del software.
3. Mejorar el tiempo y coste de desarrollo y mantenimiento de los sistemas informáticos.
4. Mejorar la planificación de un proyecto
5. Aumentar la biblioteca de conocimiento informático de una empresa ayudando a la búsqueda de soluciones para los requisitos.
6. Automatizar, desarrollo del software, documentación, generación de código, pruebas de errores y gestión del proyecto.
7. Ayuda a la reutilización del software, portabilidad y estandarización de la documentación
8. Gestión global en todas las fases de desarrollo de software con una misma herramienta.
9. Facilitar el uso de las distintas metodologías propias de la ingeniería del software.

Para clasificarlas, es necesario determinar las plataformas que soportan, las fases del ciclo de vida del desarrollo de sistemas que cubren, la arquitectura de las aplicaciones que producen y su funcionalidad.

El trabajo de esta tesina de grado consistió en desarrollar una herramienta CASE que básicamente permite asistir al ingeniero de software:

- Realizar rápidamente una evaluación de un producto de software de cualquier tamaño, pero con particular interés en los productos medianos y grandes.
- Tener control y poder hacer seguimiento de múltiples proyectos de software a medida que van evolucionando con el tiempo.
- Definir métricas personalizadas para evaluar la calidad de un producto dado.
- Generar información valiosa para utilizar en proyectos futuros.
- Generar feedback para una etapa de estimación.

Vale aclarar que ya existen en el mercado herramientas que cubren parcialmente algunos de estos puntos, por eso parte del trabajo de esta tesina ha sido investigar sobre ellas y tomar sus puntos mas sobresalientes y falencias, para reunir lo mejor de cada una y plasmarlas en una nueva herramienta así también como dejarlas documentadas para servir de referencia a la hora de diseñar una nueva herramienta en el futuro.

El trabajo a realizar, es un trabajo conjunto con la Universidad Austral, dado que para el seguimiento de proyectos, la herramienta cuenta con un background teórico, dado en la Tesis de doctorado de Gabriela Robiolo. [MDOO]

Se espera entonces, poder desarrollar una herramienta que contemple en conjunto las mejores características de las herramientas existentes en el mercado y a su vez implemente los conceptos planteados en la Tesis de doctorado de Gabriela Robiolo.

# 2 Motivación

Como se mencionaba anteriormente, a la hora de hacer seguimiento y control de proyectos es deseable conocer información sobre sus atributos y como estos evolucionan con el tiempo. Esto permitirá al ingeniero de software tomar decisiones oportunas que marquen el rumbo de un proyecto y ayuden a mejorar su calidad, muy necesaria construir los cimientos de proyectos futuros y para lograr que el software este a la altura de nuevas exigencias del mercado. Este seguimiento y control es posible realizarlo en forma manual si el proyecto es relativamente pequeño, pero se dificulta progresivamente a medida que el tamaño se hace cada vez más grande. Esto hace necesario el contar con información en las distintas etapas del proyecto, para una buena toma de decisiones y que sirvan a su vez como línea base para futuros proyectos.

Por esto, el puntapié inicial de este trabajo fue de investigación sobre las herramientas existentes en el mercado. Se detectaron las siguientes situaciones:

- Existen una gran variedad de herramientas que miden en forma automática métricas de producto, pero suelen estar **ligadas a ambientes de desarrollo específicos**, lo que implica que deben ser modificadas o desechadas cada vez que se cambia de ambiente de desarrollo.

- Relacionado con el punto anterior, hay una clara **falta de separación entre el proceso de obtención de los datos y su procesamiento** para la obtención de métricas. La obtención de los datos se hace implementando parcers, que deben ser rediseñados cada vez que se cambia de herramienta o cambia algún componente interno.

- Las necesidades de información sobre el avance de la construcción de un producto, para los diferentes interesados (gerentes, líderes y desarrolladores) **suele estar dispersa en un conjunto, no siempre integrado** de herramientas.

- En general, las herramientas de este tipo se ofrecen como un producto separado, **sin capacidad de integrarse** con ambientes de diseño ya existentes.

- La mayoría de las herramientas dispone de un **conjunto de métricas acotado** a calcular. No existe la posibilidad de definir métricas propias que sirvan de indicadores para objetivos específicos.

Como resultado del análisis de las herramientas existentes en el mercado se obtuvo un conjunto de características muy deseables en una herramienta, entre las que se destacan:

- Separación entre el proceso de obtención y de procesamiento, logrando interoperabilidad con otras herramientas mediante la adhesión a un estándar
- Integración con un ambiente de desarrollo familiar
- Extracción de métricas a partir de diagramas UML
- Resultados reusables
- Seguimiento de proyectos en cuanto a tamaño y calidad
- Monitoreo de múltiples proyectos
- Capacidad de definir métricas personalizadas

Por lo tanto, se intentará desarrollar una herramienta que cumpla con estos requisitos. Principalmente se buscará que agrupe información a distintos niveles, que sirva a los interesados en un proyecto para hacer seguimiento en cuanto a tamaño y en cuanto a calidad, ayudando en la recolección automática y en el procesamiento de los datos que dan origen a la información de interés.

De esta manera, se distinguen dos grandes partes en la herramienta:

## a) Seguimiento al tamaño del proyecto

El seguimiento en tamaño permitiría contar con información cuantitativa de proyectos, para poder:

- conocer el estado actual de progreso
- tener la historia del avance del proyecto
- generar una línea base para futuros proyectos
- obtener métricas de tamaño y métricas de productividad

## b) Evaluación de la calidad de un producto o proyecto

El seguimiento en cuando a calidad complementaría el seguimiento en tamaño y podría ser utilizado con dos fines: informar sobre calidad de un proyecto en ejecución o analizar rápidamente la calidad de un determinado producto ya desarrollado.

Existen ocasiones en las que puede ser necesario obtener en poco tiempo un conocimiento certero de un determinado producto, ya sea porque se necesita seleccionar un producto entre otros, o bien porque se quiere conocer la calidad lograda en un determinado avance reportado sobre un desarrollo en particular. Aquí es donde

se intentará determinar la calidad de un producto mediante su diseño arquitectónico, plasmado en diagramas UML.

Esto puede funcionar por un carril independiente al seguimiento en cuanto a tamaño, pero si validáramos el producto desarrollado de manera controlado podríamos encontrar una la relación tamaño – calidad, respondiendo a la siguiente pregunta:

"¿El esfuerzo medido en un producto corresponde a un diseño de calidad?"

Esto lleva a percibir la necesidad de encontrar un conjunto de métricas que permitan apreciar la calidad de un determinado producto. Pero, sin duda que el concepto de calidad es un concepto amplio y puede ser tratado desde diferentes puntos de vista.

Davin Garbin describe "la calidad es un concepto complejo y multifacético que puede describirse desde cinco perspectivas: visión trascendental, visión del usuario, visión de manufactura, visión del producto, visión basada en el valor". En la herramienta a desarrollar se contemplará la visión del producto, esto es, ver el producto hacia adentro, considerando sus características inherentes, vistas desde el diseño [PFL]. Evaluar la calidad midiendo las características internas de un producto es atractivo porque ofrece una visión de calidad objetiva e independiente del contexto [SQ].

Evaluando las posibles bases sobre las que se van a tomar las mediciones, se plantean las siguientes alternativas: el modelo de requerimientos, el modelo de análisis, el modelo de diseño o el código. Este trabajo intentará poder aplicar un subconjunto de las métricas definidas por la Mg. Gabriela Robiolo en su tesis, quien definió un framework de métricas en código Java, pero con características generales de cualquier lenguaje de programación orientado a objetos. De esta manera, se espera que la herramienta pueda obtener métricas definidas, pero a partir de diagramas UML, logrando de esta manera la independencia del lenguaje.

Se buscará implementar métricas que puedan ser útiles para analizar un aspecto de la calidad del diseño de un producto de software planteado con una orientación a objetos y modelado con UML.

# 3 Conceptos previos

En esta sección, se describirán algunos conceptos claves que forman la base para el planteamiento de la solución. Se mencionan primeramente los estándares UML y XMI. Luego, varios conceptos importantes introducidos en la Tesis de Doctorado de Gabriela Robiolo [MDOO], claves para entender los componentes que hacen al desarrollo de la herramienta. Finalmente, se introducen las herramientas Eclipse y Racional Software Architect, herramientas sobre las cuales se ha implantado la solución.

## 3.1 UML (Unified Modeling Language)

Cualquier rama de ingeniería o arquitectura ha encontrado útil desde hace mucho tiempo la representación de los diseños de forma gráfica. Desde los inicios de la informática se han estado utilizando distintas formas de representar los diseños de una forma más bien personal o con algún modelo gráfico. La falta de estandarización en la manera de representar gráfica-mente un modelo impedía que los diseños gráficos realizados se pudieran compartir fácilmente entre distintos diseñadores.

Se necesitaba por tanto un lenguaje no sólo para comunicar las ideas a otros desarrolladores sino también para servir de apoyo en los procesos de análisis de un problema. Con este objetivo se creo el Lenguaje Unificado de Modelado (UML: Unified Modeling Language). UML es un lenguaje para especificar, construir, visualizar y documentar los artefactos de un sistema de software orientado a objetos (OO). Un artefacto es una información que es utilizada o producida mediante un proceso de desarrollo de software.

Tal como indica su nombre, UML es un lenguaje de modelado. Un modelo es una simplificación de la realidad. El objetivo del modelado de un sistema es capturar las partes esenciales del sistema. Para facilitar este modelado, se realiza una abstracción y se plasma en una notación gráfica. Esto se conoce como modelado visual.

El modelado visual permite manejar la complejidad de los sistemas a analizar o diseñar. De la misma forma que para construir una choza no hace falta un modelo, cuando se intenta construir un sistema complejo como un rascacielos, es necesario abstraer la complejidad en modelos que el ser humano pueda entender.

UML sirve para el modelado completo de sistemas complejos, tanto en el diseño de los sistemas software como para la arquitectura hardware donde se ejecuten.
Otro objetivo de este modelado visual es que sea independiente del lenguaje de implementación, de tal forma que los diseños realizados usando UML se puedan implementar en cualquier lenguaje que soporte las posibilidades de UML (principalmente lenguajes orientados a objetos).

UML es además un método formal de modelado. Esto aporta las siguientes ventajas:

- Mayor rigor en la especificación
- Permite realizar una verificación y validación del modelo realizado.
- Se pueden automatizar determinados procesos y permite generar código a partir de los modelos y a la inversa (a partir del código fuente generar los modelos). Esto permite que el modelo y el código estén actualizados, con lo que siempre se puede mantener la visión en el diseño, de más alto nivel, de la estructura de un proyecto.

Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones:

- **Visualizar**: UML permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- **Especificar**: UML permite especificar cuáles son las características de un sistema antes de su construcción.
- **Construir**: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- **Documentar**: Los propios elementos gráficos sirven como documentación del sistema des-arrollado que pueden servir para su futura re-visión.

Aunque UML está pensado para modelar sistemas complejos con gran cantidad de software, el lenguaje es los suficientemente expresivo como para modelar sistemas que no son informáticos, como flujos de trabajo (workflow) en una empresa, diseño de la estructura de una organización y por supuesto, en el diseño de hardware.

Una de las partes que componen UML es un metamodelo formal. Un metamodelo es un modelo que define el lenguaje para expresar otros modelos. Un modelo en OO es una abstracción cerrada semánticamente de un sistema. Sin embargo, los modelos se pueden utilizar en muchas actividades de la vida humana: antes de construir una casa el arquitecto utiliza un plano, los músicos representan la música en forma de notas musicales, los artistas pintan sobre el lienzo con carboncillos antes de empezar a

utilizar los óleos, etc. Un sistema puede ser descripto por uno o más modelos, posiblemente desde distintos puntos de vista.

Para mantener su característica de estándar abierto, UML es independiente de los lenguajes, de las bases de datos y de la marca del software y del equipo de cómputo. Deliberadamente, los proponentes de este lenguaje unificado lo han dejado únicamente como un medio para especificar, visualizar y documentar los artefactos del desarrollo e implantación de sistemas.

UML ofrece una amplia variedad de diagramas para visualizar el sistema desde varias perspectivas. UML incluye los siguientes diagramas:

- Diagrama de casos de uso.
- Diagrama de clases.
- Diagrama de objetos.
- Diagrama de secuencia.
- Diagrama de colaboración
- Diagrama de estados
- Diagrama de actividades
- Diagrama de componentes
- Diagrama de despliegue.

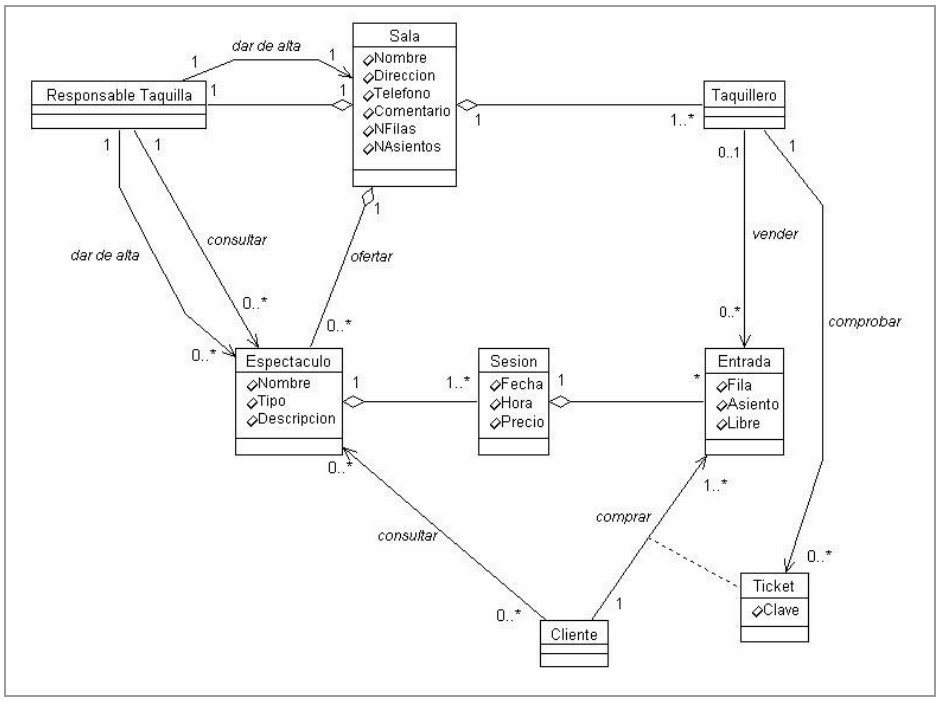


**Figura 3-1 Diagrama de clases UML**

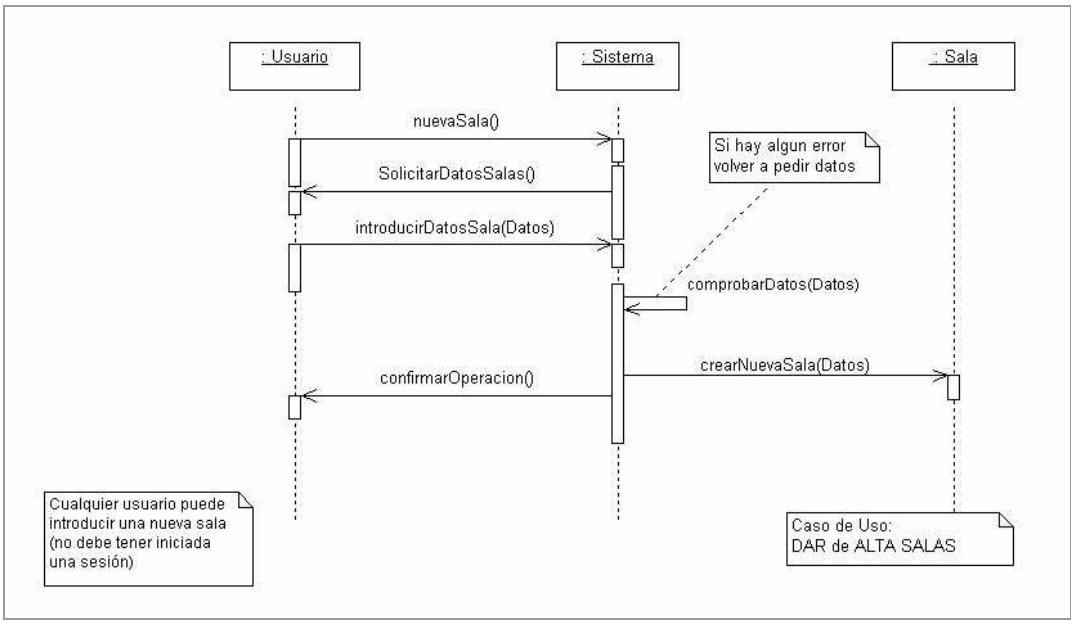


**Figura 3-2 Diagrama de secuencia UML**

UML incluye una característica que será de gran importancia en este trabajo: Los estereotipos: los estereotipos son una nueva clase de elemento de modelado que debe basarse en ciertas clases ya existentes en el metamodelo y constituye un mecanismo de extensión del modelo. Así, los estereotipos pueden enriquecer semánticamente a un modelo UML, extendiéndolo según la necesidad lo determine.

**Concepto de meta-metamodelo y metamodelo**

Un meta-metamodelo (OMG, 2003) es un modelo que define el lenguaje formal para representar un metamodelo. La relación entre un meta-metamodelo y un metamodelo es análoga a la relación entre un metamodelo y modelo. Un metamodelo (OMG, 2003) es un modelo que define el lenguaje formal para representar un modelo.

## 3.2 XMI (XML Metadata Interchange)

XMI [XMI] es un uso propuesto de XML que tiene la intención de proveer una manera estándar de intercambiar información sobre meta datos. XMI intenta ayudar a los programadores y diseñadores de sistemas a intercambiar diagramas UML entre diferentes herramientas. Adicionalmente, XMI también puede intercambiar información sobre datawarehouses.

El formato XMI estandariza como cualquier conjunto de meta-datos puede ser descripto y permite a usuarios de diferentes ambientes visualizar la información consistentemente.

A continuación se muestra la estructura de un archivo ejemplo serializado en XMI:

```
<XMI version="1.1" xmlns:UML="org.omg/UML1.3">
… (header)
<XMI.content>
<UML:Class name="Departamento"
xmi.id="Departamento"/>
<UML:Class name="Instructor"
xmi.id="Instructor"/>
<UML:Class name="Profesor"
xmi.id="Profesor“ generalization="Instructor"/>
<UML:Class name="Catedrático"
xmi.id="Catedrático" generalization="Instructor"/>
<UML:Class name="Monitor"
xmi.id="Monitor“ generalization="Instructor"/>
<UML:Association>
<UML:Association.connection>
<UML:AssociationEnd name="instructores“
type="Instructor"/>
<UML:AssociationEnd name="miembroDe“
type="Departamento"/>
</UML:Association.connection>
</UML:Association>
</XMI.content>
</XMI>
```

**Figura 3-3 Ejemplo de archivo XMI**

A continuación se muestra como XMI facilita la interoperabilidad entre herramientas serializando la información y proporcionando un medio único para la comunicación.

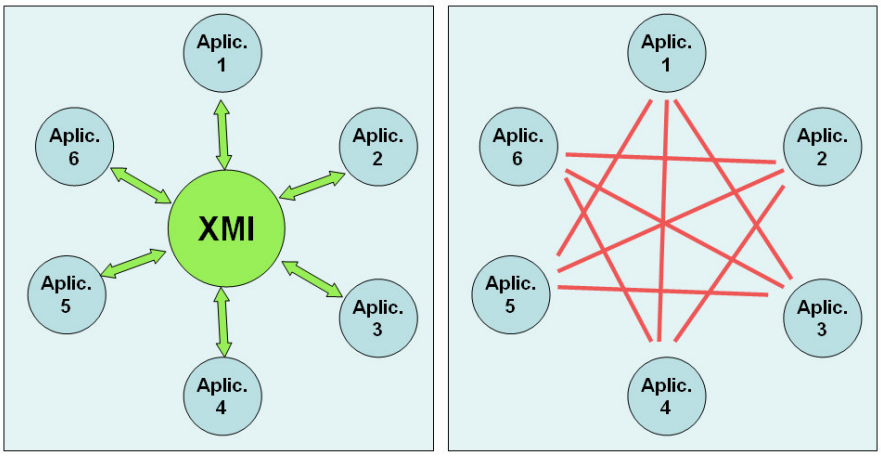


**Figura 3-4 Comparación del uso de XMI**

Se debe mencionar que XMI proporciona el medio para serializar los meta-datos, sin embargo, no contempla la representación grafica de los meta-modelos. Por lo tanto, al intercambiar información de un diagrama UML de una herramienta a la otra, se pierde la disposición grafica.

## 3.3 Objetos de entidad y transacciones

La tesis de G. Robiolo introduce conceptos que son tenidos en cuenta por la herramienta que se desarrollo. Son conceptos tomados de Jacobson, pero utilizados como base para realizar estimaciones y mediciones de tamaño de software.

### a) Objetos de entidad

Jacobson define a los objetos de entidad como aquellos que manejan información persistente del modelo. La mayoría de estos son identificados en los casos de uso y son generalmente los objetos más obvios que participan en el modelo de objetos del dominio del problema. Los objetos de entidad están asociados con conceptos en la vida real, existentes fuera de las barreras del sistema. Por ejemplo, se muestra a continuación los pasos en el caso de uso "Ingresar un nuevo producto en el catalogo" (un caso de uso ejemplo para mostrar la identificación de los objetos de entidad). Los objetos de entidad se resaltan en negrita.

El sistema muestra una pantalla con la lista de **categorías** y **subcategorías**
El usuario selecciona la **categoría** a la cual el nuevo **producto** pertenece.
Presionando "enter", el usuario podrá ver una pantalla donde llenar los campos de las propiedades del nuevo **producto**
El caso de uso termina cuando el usuario completa los campos del **producto** y presiona "Enter", o presiona cancelar.

**Figura 3-5 Ejemplo de un escenario de un Caso de Uso**

### b) Casos de uso y transacciones

Jacobson define un caso de uso como una manera especial de usar el sistema accionando ciertas partes de su funcionalidad. Cada caso de uso define eventos lanzados por un actor y especifica al interacción que se da entre el actor y el sistema [JAC]. Un caso de uso es una secuencia especial de transacciones ejecutadas por el actor y el sistema en forma de dialogo. El caso de uso esta definido desde la perspectiva del actor.

Desde el punto de vista del sistema, se puede decir que un caso de uso es un flujo completo del sistema, lo que es, una transacción. De hecho, uno puede tener una o varias transacciones, dependiendo del criterio de la persona que escribe el caso de uso. Las transacciones pueden verse claramente considerando los estímulos del actor hacia el sistema.

## 3.4 Introducción a la herramienta "Eclipse"

Eclipse [ECLI] es una plataforma de desarrollo open source basada en Java. Es un desarrollo de IBM cuyo código fuente fue puesto a disposición de los usuarios. En si mismo Eclipse es un marco y un conjunto de servicios para construir un entorno de desarrollo a partir de componentes conectados (plug-ins). Hay plug-ins para el desarrollo de Java así como para el desarrollo en C/C++, COBOL, etc.

Se dice que es un marco por que la misma tecnología que se utiliza para crear eclipse (SWT y JFace) puede ser usada para crear nuevas aplicaciones de escritorio (llamadas Aplicaciones de cliente rico), o para crear mismos plugins para el Eclipse o productos derivados de el.

Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

## 3.5 Introducción al producto "Rational Software Architect"

IBM Rational Software Architect [RSA] es una herramienta de diseño y desarrollo que promueve el desarrollo guiado por modelos con el UML para crear aplicaciones y servicios.

Este producto, unifica todos los aspectos del diseño y del desarrollo de software en una herramienta que es fácil de usar. Da soporte a la comprensión, el diseño, la administración y la evolución de soluciones y servicios. El producto incluye todos los dispositivos de J2EE, de la Web y servicios de la Web que se encuentran en Rational Application Developer for WebSphere Software.

Rational Software Architect está creado sobre la plataforma abierta y extensible de Eclipse, lo que promueve varios estándares abiertos industriales. Esto permite agregarle funcionalidad mediante el desarrollo de plugins para Eclipse. Entre sus funcionalidades se destacan:

- **Modelado y especificación arquitectónicos**
  - Da soporte a los principales diagramas de UML 2
  - Permite exportar diagramas a XMI
  - Da soporte a patrones y transformaciones para automatizar el ajuste de los modelos y la transición entre el análisis, el diseño y la implementación
  - Da soporte a OCL para especificar las limitaciones arquitectónicas

- **Revisión y control estructural de las aplicaciones de Java**
  - Detección automática de los patrones y anti-patrones estructurales para facilitar la reelaboración de las aplicaciones de Java
  - Define las normas estructurales para el control arquitectónico de Java

- **Facilidad en la adopción y el uso**
  - Interfaz de usuarios simplificada y receptiva
  - Navegación y exploración de modelos o código utilizando diagramas
  - Generación de diagramas automática y asistida

- **Integración del ciclo de vida y de los equipos**
    - Se Integra a IBM Racional RequisitePro, IBM Rational ClearCase LT e IBM Rational ClearQuest
    - Incluye una configuración de IBM Rational Unified Process para Software Architects
    - Da soporte a los CVS y SVN para la administración de la configuración de software

# 4 Ideando una solución

En este capitulo se realizara un acercamiento de la solución al problema planteado: el desarrollo de una herramienta que sirva al ingeniero de software a la hora de controlar el avance de un proyecto y a su vez brinde rápidamente una imagen acerca de la calidad de un determinado producto.

## 4.1 Características principales en la herramienta

Aquí se describirán las características generales que se consideraron a la hora de desarrollar el sistema propuesto, que también pueden servir de referencia a otros sistemas de este tipo. Estas características son las elegidas motivadas por el estudio realizado a las herramientas existentes en el mercado, tomando lo mejor de cada una de ellas. Más adelante se detallarán los casos de uso que reflejan la funcionalidad específica del sistema.

### 1. Capacidad de obtención de información desde modelos UML

Se espera que la herramienta pueda tomar como base para el calculo de métricas diagramas UML serializados de alguna manera. UML es ampliamente utilizado para el análisis y desarrollo de sistemas de distintos tipos, por ello, el apego a el facilita su incorporación a procesos productivos existentes.

En especial, para poder lograr la implementación de las métricas definidas en la Tesis de Doctorado de Gabriela Robiolo [MDOO] se buscará tomar información desde diagramas UML de clases y secuencia.

### 2. Separación del proceso de extracción de información del modelo

Para lograr la capacidad de procesar diagramas UML generados con diferentes herramientas se debe tener necesariamente una separación entre el proceso de extracción y el de procesamiento de la información.[AMUM]
Para esto se consideró fundamental la adhesión al estándar XMI para captar la información del modelo. XMI, como se describió previamente, se perfila como el estándar a adoptar por la industria para el intercambio de modelos entre distintas herramientas, por eso se intentará apegarse a el.

### 3. Generar información relevante para los distintos niveles

Se pretende que la herramienta pueda brindar información a los distintos interesados en un proyecto:

- desarrolladores
- lideres de proyecto
- gerentes

Dados estos roles, es deseable en un sistema de estas características que se permita el acceso a la información según el perfil del usuario, permitiendo visualizar a los desarrolladores solamente los proyectos en los que participan junto con su información relacionada a esos proyectos. Para los líderes de proyecto, poder seguir la labor de los desarrolladores solo en los proyectos que ellos lideran. Finalmente para los gerentes, tener un panorama general del funcionamiento de los proyectos.

### 4. Integración con un ambiente de desarrollo habitual

El frontend de la herramienta (la interfaz visible para el usuario) deberá insertarse en una herramienta de desarrollo que resultar familiar y a su vez seguir los criterios de construcción de interfaz definidos, para lograr que la curva de aprendizaje y capacitación del usuario sea mínima.

### 5. Realizar seguimiento de tamaño y calidad en el tiempo

Es tan importante controlar el avance del proyecto en cuanto a tamaño como en cuanto a calidad. Estos dos indicadores son clave a la hora del monitoreo. Generalmente, el mayor esfuerzo en un proyecto de software esta puesto en la etapa de mantenimiento (o evolución), por lo que es importante, además de informar el tamaño absoluto del proyecto poder brindar información cuantitativa de cambios entre distintas versiones, tal y como lo mencionan Luigi Lavazza y Alberto Agostini en el paper "Automated Measurement of UML Models: an open toolset approach" [AMUM]

### 6. Monitorear múltiples proyectos

En las organizaciones y empresas desarrolladoras de software es común tener varios proyectos en ejecución simultáneamente. Por esto se espera que la herramienta tenga

soporte para el manejo de múltiples proyectos. Esto también implica que los usuarios deberán tener visibilidad y acceso solo a los proyectos que se les asigne. [FCSMT]

## 7. Resultados usables

Será deseable que los resultados obtenidos con esta herramienta (métricas, mediciones, horas trabajadas, etc) puedan ser utilizados y sirvan de entrada a otras herramientas de alguna manera. [RASM]

## 8. Portabilidad

Se intentará que el desarrollo de la herramienta se adapte a distintas plataformas fácilmente. Se entiende como plataforma la combinación entre hardware y el sistema operativo. [FCSMT]

## 9. Capacidad para definir métricas personalizadas

En el campo de la ingeniería de software, no hay un único conjunto de métricas que sirvan de base para un set completo de indicadores satisfactorios. La definición de métricas es una actividad orientada a objetivos, de aquí la importancia de permitir definir libremente métricas sobre un modelo. [RASM]

## 4.2 Análisis y diseño de la herramienta

### 4.2.1 Casos de Uso

A continuación se describen los casos de uso que se especificaron:

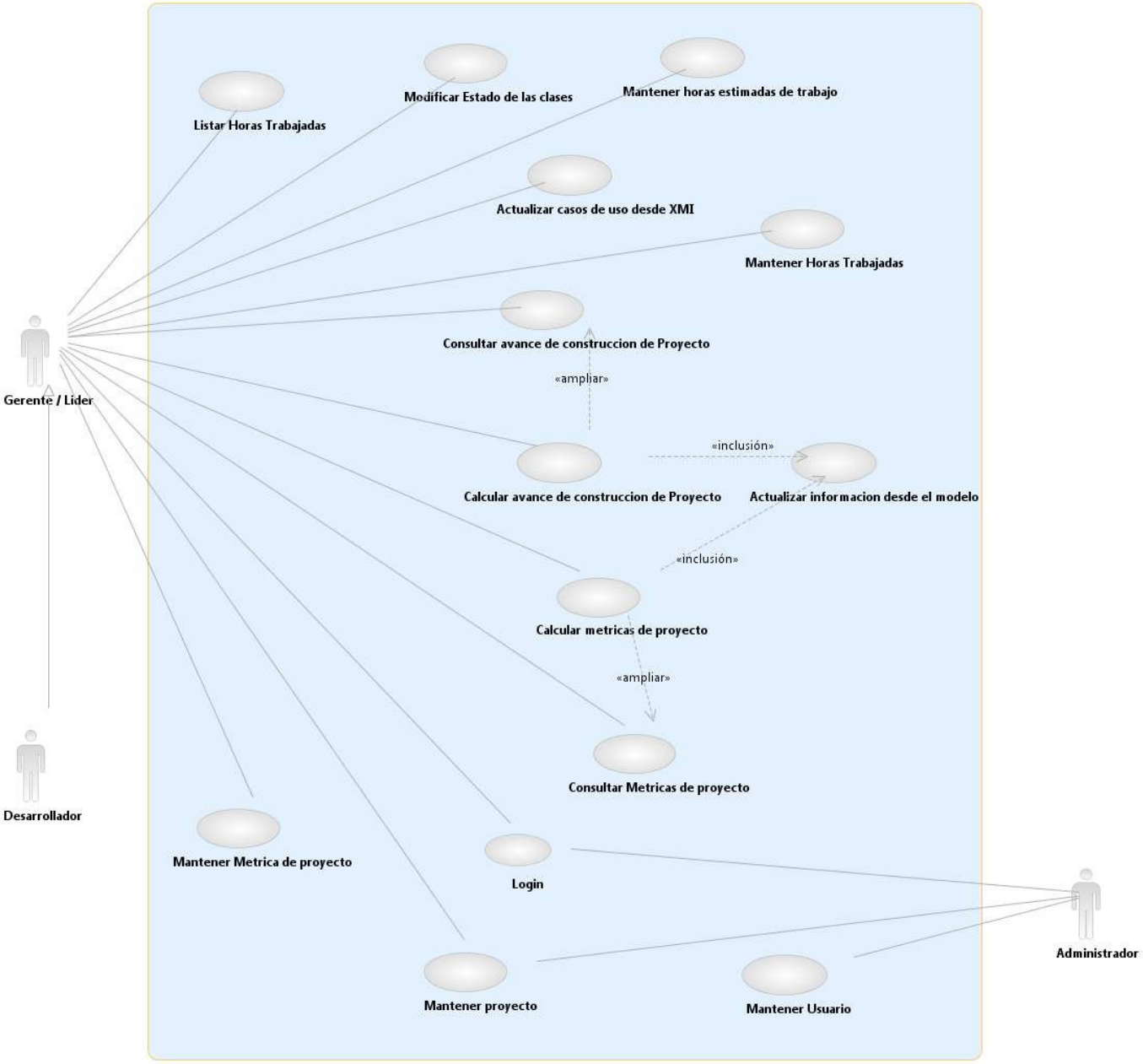


**Figura 4-1 Diagrama de Casos de Uso de la herramienta**

#### 4.2.1.1 Acceso al sistema

**Actor**
Desarrollador – Líder – Administrador – Gerente

**Descripción**
El actor necesita loguearse al sistema para poder acceder a las funciones que brinda el mismo. Al loguearse se carga su perfil, se muestran las operaciones que tiene permitido realizar y los proyectos a los que tiene acceso.

**Flujo de eventos normal**
- El sistema solicita al usuario ingresar nombre de usuario y password
- El usuario ingresa nombre de usuario y password y presiona entrar
- El sistema muestra los proyectos que el usuario tiene acceso y activa las funciones que puede acceder mediante el menú.

**Flujo de eventos anormal**
Si el usuario ingresa el nombre de usuario o password invalido el sistema le muestra un mensaje de error y le niega el acceso.

**Precondiciones**
-

**Post condiciones**
El usuario accedió al sistema y se carga en sesión su perfil para poder acceder a operaciones y proyectos.

#### 4.2.1.2 Actualizar información desde el modelo

**Actor**
Desarrollador – Líder

**Descripción**
Este caso de uso se ejecutará para retornar un objeto medición de tamaño **o** calidad dependiendo de un parámetro recibido.

**Flujo de eventos normal**
- Se recibe un parámetro que determina si se quiere actualizar información para generar una medida de tamaño o de calidad.

a. **Generar medición de tamaño**
   - Se procesa el los archivos XMI para la obtención de diagramas de clases y secuencia.
   - Se devuelve la información del diagrama de clases.
   - Se obtiene y devuelve de los diagramas de secuencia: cantidad de transacciones no finalizadas, finalizadas y rehechas. Se calcula el porcentaje de avance.

b. **Generar medición de calidad**
   - Se procesa los archivos XMI para la obtención de diagramas de clases. Se consulta este modelo mediante un parámetro que es una consulta SQL que se ejecuta sobre este, devolviendo la información deseada.

**Flujo de eventos anormal**
*El proyecto tiene asociado un diagrama de secuencia pero no de clases.*
El sistema le muestra un mensaje de error.

**Precondiciones**
El usuario se encuentra logueado y existen proyectos ya dados de alta.

**Post condiciones**
Se actualizan en la Base de Datos los siguientes valores:
- Los estados de las clases del proyecto.
- La cantidad de transacciones que tiene cada diagrama de secuencia, diferenciando las incorporadas, las no finalizadas, las rehechas y las finalizadas.

#### 4.2.1.3 Calcular métricas de proyecto

**Actor**
Desarrollador - Líder

**Descripción**
Este caso de uso se ejecutará para actualizar las métricas de Proyecto en la base de datos. Se calculará sobre los datos del proyecto almacenados en la base de datos.

**Flujo de eventos normal**
- El sistema muestra los proyectos que tiene dados de alta y las posibles métricas que puede calcular.
- El usuario selecciona un proyecto y las métricas que va a calcular.
- Se ejecuta el caso de uso "Actualizar información desde el modelo" con el parámetro "medición de calidad".
- Se calcula cada métrica del proyecto que ha sido seleccionada y el resultado se guarda en la base de datos.
- El sistema muestra un mensaje de información donde se solicita consultar las métricas recientemente guardadas.
- Si el usuario selecciona aceptar se ejecuta el caso de uso "Consultar métricas de proyecto".

**Flujo de eventos anormal**
*El Diagrama de clases no ha sido generado.*
El sistema le muestra un mensaje de error.

**Precondiciones**
El usuario se encuentra logueado.

**Post condiciones**
Se actualiza la Base de Datos para el proyecto elegido sus valores de cada una de las métricas seleccionadas.

#### 4.2.1.4 Modificar estado de las clases

**Actor**
Desarrollador

**Descripción**
Este caso de uso lo realiza el usuario en el ambiente de edición de las clases. El desarrollador marcará en la herramienta de desarrollo las clases que están finalizadas.

| Estado | Próximo estado |
|---|---|
| Por default | No finalizado |
| No finalizado | Finalizado |
| Finalizado | Re-hecho |
| Re-hecho | Finalizado |

**Flujo de eventos normal**
El usuario marcará en el ambiente de diseño las clases con los diferentes estados.

**Flujo de eventos anormal**
-

**Precondiciones**
El usuario se encuentra logueado

**Post condiciones**
Se editó el estado de las clases del diagrama de clases. Esto se hará visible cuando el diagrama sea exportado a XML

### 4.2.1.5 Mantener horas trabajadas

**Actor**
Desarrollador

**Descripción**
El actor podrá ingresar, modificar o consultar horas trabajadas.

Cada desarrollador ingresa las horas trabajadas en forma periódica. Cada período puede ser un día o cada vez que cierre y abra la herramienta. Se el facilitará al desarrollador la carga de horas trabajadas. Una forma puede ser por medio de alarmas, si es que no lo ha hecho, en el lapso de 24 hs.

A cada usuario se le permite seleccionar todos los proyectos a los que tiene acceso para consultar las horas trabajadas. Sólo el Desarrollador puede modificar sus horas trabajadas.

**Flujo de eventos normal**
El sistema solicita al usuario que seleccione la funcionalidad a ejecutar.
- Si el usuario selecciona "Ingresar hs. Trabajadas", se ejecuta el Sub-flujo Ingresar hs. trabajadas.

- Si el usuario selecciona "Consultar hs. Trabajadas", se ejecuta el Sub-flujo Consultar hs. Trabajadas.

a. **Ingresar horas trabajadas**
   El sistema periódicamente muestra una pantalla con los proyectos que tiene dados de alta por el desarrollador, la última carga de horas trabajadas realizada para cada proyecto y le permite realizar para una fecha dada, una nueva carga de horas trabajadas en horas y minutos, por caso de uso.

| Fecha | Proyecto | [Casos de uso] | Horas Trabajadas |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

b. **Consultar hs trabajadas**

- El usuario selecciona si quiere ver el detalle de horas trabajadas por proyecto o por desarrollador.
- El sistema muestra los proyectos o los desarrolladores que tiene dados de alta.
- El actor selecciona las entidades de interés y se muestra el resultado de la consulta, obteniéndose de la base de datos.
- Si se ha cargado las horas trabajadas por caso de uso se podrá ver el detalle de horas trabajadas por caso de uso.
- El desarrollador puede modificar los valores de sus horas trabajadas.

**Flujo de eventos anormal**

-

**Precondiciones**

El usuario se encuentra logueado y existen proyectos dados de alta a los cuales se puedan ingresar o consultar hs. trabajadas.

**Post condiciones**

- Si se ejecuto el subflujo "Ingresar hs Trabajadas", se actualizan la Base de Datos con valores de las horas trabajadas para cada Proyecto, por cada desarrollador y por cada caso de uso si la hubieren detallado.
- Si se ejecuto el subflujo "Consultar hs. trabajadas", se consulto horas trabajadas por desarrollador o por proyecto. Si el usuario es un desarrollador y accedió a sus horas trabajadas eventualmente pudo modificarlas

### 4.2.1.6 Calcular avance de construcción de proyecto

**Actor**
Desarrollador - Líder

**Descripción**
Este caso de uso se ejecutará para actualizar el avance de construcción de todos los casos de uso de un Proyecto en la base de datos.

**Flujo de eventos normal**
- El sistema muestra los proyectos visibles al perfil del usuario logueado.
- El usuario selecciona un proyecto.
- El sistema llama al caso de uso Actualizar información desde el modelo (con el parámetro medición de tamaño) quien genera una nueva medición de tamaño sobre el proyecto donde se actualiza para cada caso de uso: cantidad de transacciones no finalizadas, finalizadas, rehechas y porcentaje de avance.
- El sistema solicita si se desea consultar la medición tomada.
- Si el usuario presione en aceptar se ejecuta el caso de uso "Consultar avance de construcción de proyecto"

**Flujo de eventos anormal**
*El Diagrama de clases no ha sido generado.*
El sistema le muestra un mensaje de error.

*Los Diagramas de secuencia no han sido generados.*
El sistema le muestra un mensaje de error.

**Precondiciones**
El usuario se encuentra logueado.

**Post condiciones**
Se registro en la base de conocimiento la medición de tamaño sobre el proyecto solicitado.

### 4.2.1.7 Consultar avance de construcción de proyecto

**Actor**

Desarrollador – Líder - Gerente

**Descripción**

Cada actor tiene una visión más amplia de la consulta. Un gerente tiene líderes y desarrolladores a cargo. Un Líder tiene desarrolladores a cargo. Un desarrollador tiene proyectos a cargo. Teniendo el mismo formato, a cada uno se le permite seleccionar todos los proyectos a los que tiene acceso.

**Flujo de eventos normal**

- El sistema muestra los proyectos que tiene dados de alta.
- El usuario selecciona los proyectos a consultar.
- Se muestra por proyecto los ítems detallados en el cuadro.

| Proyecto | Caso de uso | Cantidad de trans. pactadas | Cantidad de trans. incorporadas | Cantidad de trans. no finalizadas |
|---|---|---|---|---|
| | | | | |
| | | | | |
| Total | | | | |

| Caso de uso | Cantidad de trans. pactadas | Cantidad de trans. incorporadas | Cantidad de trans. no finalizadas |
|---|---|---|---|
| | | | |
| | | | |
| | | | |

**Flujo de eventos anormal**

-

**Precondiciones**

- El usuario se encuentra logueado.
- Existen proyectos dados de alta con información actualizada desde el modelo.

**Post condiciones**

Se consulto el avance de construcción de los proyectos elegidos

### 4.2.1.8 Consultar métricas de proyecto

**Actor**
Desarrollador – Líder - Gerente

**Descripción**
Cada actor tiene una visión más amplia de la consulta. Un gerente tiene líderes y desarrolladores a cargo. Un Líder tiene desarrolladores a cargo. Un desarrollador tiene proyectos a cargo. El sistema muestra un listado de los proyectos visibles al usuario. Permite visualizar los valores de las distintas métricas de los proyectos.

**Flujo de eventos normal**
- El sistema muestra los proyectos que puede acceder el usuario logueado.
- El usuario selecciona los proyectos a consultar.
- El sistema muestra por las mediciones tomadas a lo largo del tiempo
- El usuario selecciona una o varias mediciones
- El sistema muestra las métricas de cada medición

**Flujo de eventos anormal**
-

**Precondiciones**
- El usuario se encuentra logueado.
- Existen proyectos almacenados con información del modelo.

**Post condiciones**
Se consultaron las métricas de los proyectos deseados

#### 4.2.1.9 Mantener usuario

**Actor**
Administrador

**Descripción**
Escenario general donde se realiza la administración de usuarios, perfiles con permisos y asociaciones entre usuarios y perfiles.

**Flujo de eventos normal**
- El sistema muestra la lista de usuarios del sistema, los perfiles, con sus permisos relativos. Además permite dar de alta un perfil, si es necesario, o modificar los permisos.
- El sistema permite dar de alta, modificar o remover un usuario.

**Flujo de eventos anormal**
-

**Precondiciones**
El administrador se encuentra logueado.

**Post condiciones**
Se dio de alta, modifico o dio de baja un usuario

#### 4.2.1.10 Mantener métrica de proyecto

**Actor**
Líder - Gerente

**Descripción**
Se da de alta, de baja o se modifica una métrica en la base de datos con los siguientes campos:
- Nombre
- Descripción
- Forma de calculo

En cualquiera de las tres operaciones los cambios son visibles para todos los usuarios y todos los proyectos.

**Flujo de eventos normal**
- Si el usuario accede al submenú "Crear métrica de proyecto" se ejecuta el sub-flujo Crear métrica de proyecto.
- Si el usuario accede al submenú "Modificar métrica de proyecto" se ejecuta el sub-flujo Modificar métrica de proyecto.
- Si el usuario accede al submenú "Borrar métrica de proyecto" se ejecuta el sub-flujo Borrar métrica de proyecto.

a. **Crear métrica de proyecto**
   - El usuario ingresa el nombre de la nueva métrica, una descripción y su forma de cálculo en forma de consulta SQL.
   - El sistema toma los datos ingresados y da de alta la nueva métrica.

b. **Modificar métrica de proyecto**
   - El sistema muestra un listado de la totalidad de métricas dadas de alta por usuario.
   - El usuario modifica el nombre de la métrica, una descripción y su forma de cálculo en forma de consulta SQL.
   - El sistema toma los datos editados y actualiza la métrica en la base de datos.

c. **Borrar métrica de proyecto**
   - El sistema muestra un listado de la totalidad de métricas dadas de alta por usuario (esto excluye a las predefinidas por el sistema).
   - El usuario elije una métrica de la lista.

- El sistema da de baja la métrica de la base de datos.

**Flujo de eventos anormal**

*Nombre de métrica ya existente*

Si en el subflujo "Crear métrica de proyecto", o en el subflujo "Modificar métrica de proyecto" se ingresa un nombre para la métrica ya existente, se muestra un mensaje de error y el sistema solicita que se ingrese otro nombre.

*La métrica a borrar ya fue usada en cálculos*

Si en el subflujo "Borrar métrica de proyecto", se intenta borrar una métrica que ya fue utilizada en cálculos, se muestra un mensaje de error y no se permite la operación.

**Precondiciones**

El usuario se encuentra logueado

**Post condiciones**

Se dio de alta, baja o modifico en la base de datos una métrica con su nombre, descripción y forma de cálculo

#### 4.2.1.11 Mantener proyecto

**Actor**
Líder

**Descripción**
Se puede dar de alta un nuevo proyecto en la base de datos. Se ingresa un nombre único y opcionalmente se asocia con un diagrama de clases y varios diagramas de secuencia.

También se puede editar un proyecto en conjunto con sus relaciones con archivos xml que representan el diagrama de clases que tiene asociado y los diagramas de secuencia y métricas.

**Flujo de eventos normal**
- Si el usuario accede al submenú "Crear proyecto" se ejecuta el sub-flujo Crear proyecto.
- Si el usuario accede al submenú "Modificar proyecto" se ejecuta el sub-flujo Modificar proyecto.
- Si el usuario accede al submenú "Borrar proyecto" se ejecuta el sub-flujo Borrar proyecto.

a. **Crear proyecto**
   - El sistema solicita un nombre para el nuevo proyecto.
   - El usuario completa el campo de texto con el nombre del proyecto.
   - El sistema solicita que se asocie un diagrama de clases al proyecto.
   - El usuario navega por directorios para indicar el path donde se encuentra el archivo .xml (XMI) que representa el diagrama de clases.
   - El sistema solicita que se asocie 1 o varios diagramas de secuencia al proyecto.
   - El usuario navega por directorios para indicar el path donde se encuentran el o los archivos .xml (XMI) que representa los diagramas de secuencia.
   - El sistema solicita al usuario que se asocie los usuarios para los cuales el proyecto será visible.
   - El usuario selecciona los usuarios que tendrán acceso a las operaciones sobre el nuevo proyecto.
   - Finalmente, el sistema permite ingresar opcionalmente el número de transacciones pactadas.

b. **Modificar proyecto**
- El sistema muestra la lista de proyectos del sistema.
- El usuario selecciona un proyecto.
- El sistema abre una ventana con la siguiente información perteneciente al proyecto:
  * Nombre
  * Dirección del archivo xml correspondiente al diagrama de clases
  * Dirección de los archivos xml pertenecientes a los diagramas de secuencia con su nombre (ligado al Caso de uso).
  * Usuarios que tienen acceso al proyecto
  * Cantidad de transacciones pactadas
- El usuario puede modificar libremente los ítems mencionados arriba, incluso agregando diagramas de secuencia y usuarios que puedan acceder.

c. **Borrar proyecto**
- El sistema muestra la lista de proyectos del sistema.
- El usuario selecciona un proyecto.
- El sistema borra el proyecto de la base de datos (no se borran archivos xml ni métricas asociadas).

**Flujo de eventos anormal**

*El nombre del proyecto ingresado ya existe.*

Si en el sub-flujo "Crear proyecto", o "Modificar Proyecto" se ingresa un nombre de proyecto ya existente, se muestra un mensaje de error. El sistema solicita al usuario que ingrese otro nombre.

*Se intenta borrar un proyecto que tiene asociaciones con diagramas UML.*

Si en el sub-flujo "Borrar proyecto", se intenta borrar un proyecto que esta asociado a diagramas UML se muestra un mensaje de error. No puede ser borrado un proyecto con asociaciones a diagramas UML.

**Precondiciones**

El usuario se encuentra logueado.

**Post condiciones**

Se dio de alta, de baja o se modificó un proyecto en la base de datos.

### 4.2.2 Diagrama de clases

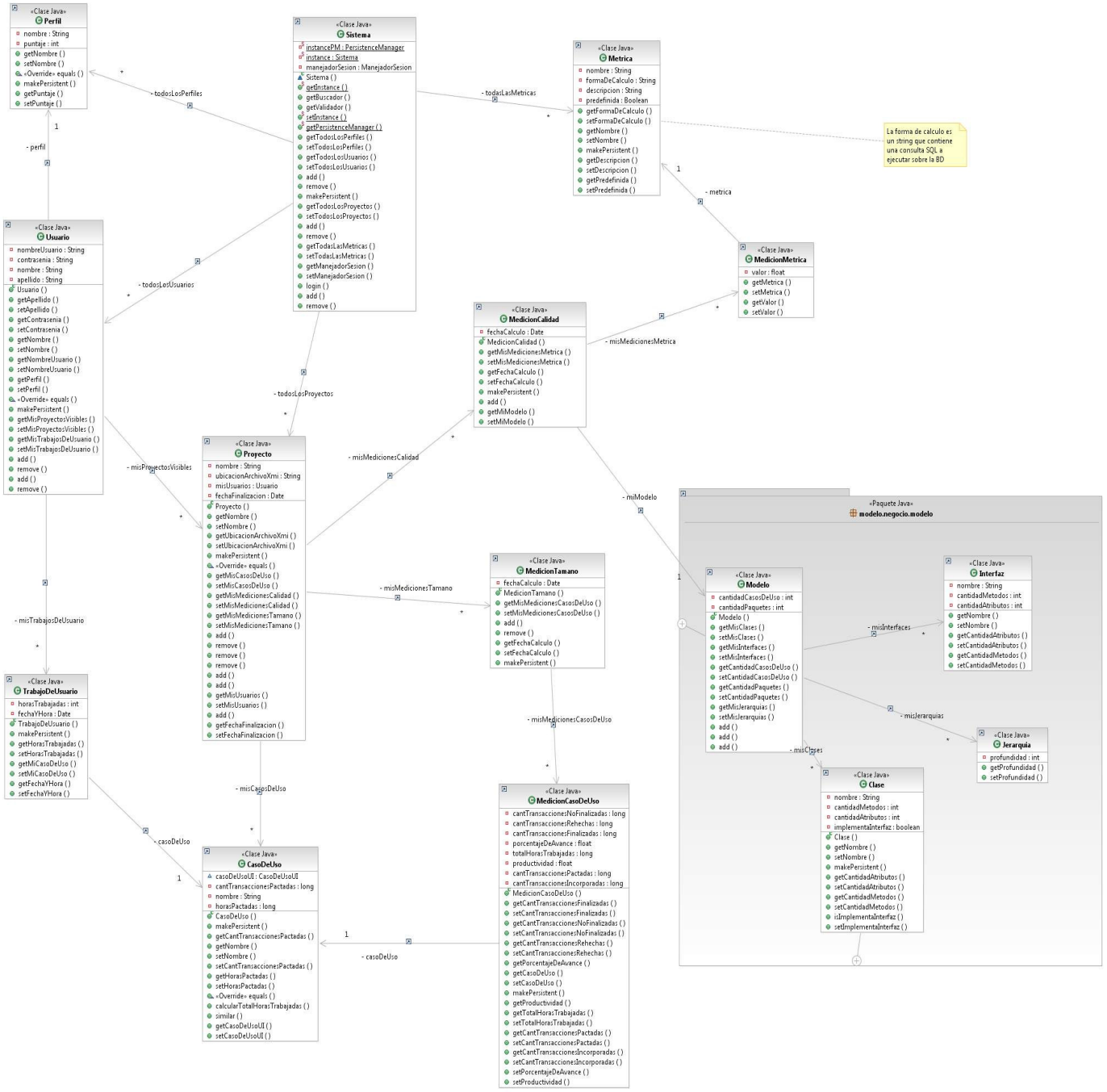


**Figura 4-2 Diagrama de clases de la herramienta**

# 5 Desarrollando el sistema

Teniendo los objetivos claros, se buscaron ideas y soluciones para la implementación e integración con un ambiente de diseño y desarrollo. Finalmente se tomaron las siguientes decisiones:

## Herramienta de desarrollo / diseño a integrar

Dado que el sistema debía resultar familiar al usuario, se decidió la realización del mismo en forma de modulo (plugin) para Eclipse [ECLI]. Por esto, se utilizó Eclipse para desarrollarlo, como framework, y también como objetivo de la aplicación final.
De esta manera se logra la inserción de la herramienta en el entorno de trabajo habitual, dado que Eclipse es uno de los entornos de desarrollo y diseño mas utilizados. Eclipse también es independiente de la tecnología: hay distribuciones para Java, C, C++, Python, Ruby, PHP, etc. Esto lo hace un objetivo ideal para la incorporación de la herramienta.

Por otro lado se decide tomar la herramienta Racional Software Architect (RSA), de IBM, para diseñar los diagramas UML a medir. Como el RSA esta construido sobre el Eclipse, entonces el mismo modulo que se incorpora a Eclipse puede incorporarse al también RSA y tener diseño y control de proyecto en un ambiente integrado. De esta manera, se obtendría un producto final multi-plataforma y también adaptable a múltiples productos.

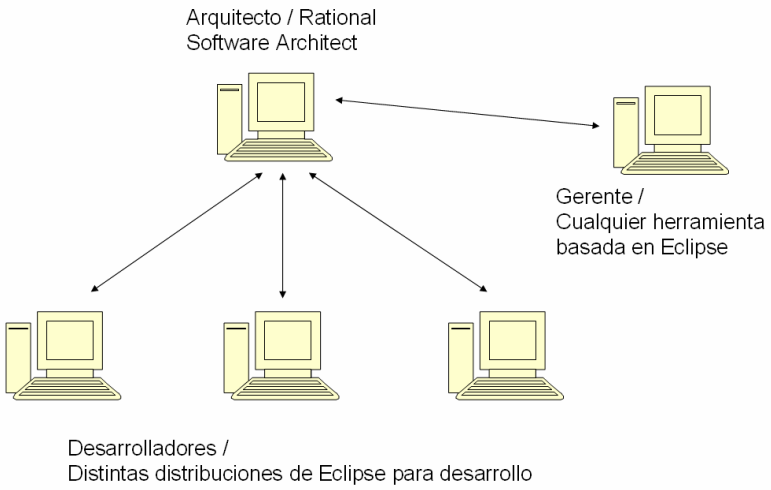


**Figura 5-1 Modelo de distribución de la herramienta**

## 5.1 Visión arquitectónica

Como se mencionaba anteriormente, se decidió implementar una herramienta como un plugin del Rational Software Architect [RSA] (versión 7), en el lenguaje Java, con SWT [SWT] y jFace, para hacer que el plugin junto con la herramienta sean portables.

Para la persistencia de la información se utilizará el motor de bases de datos relacional MySQL, dada su madurez, difusión y aceptación. De esta manera, los resultados generados por el sistema deben poder ser eventualmente utilizados por otro sistema. [MYSQL]

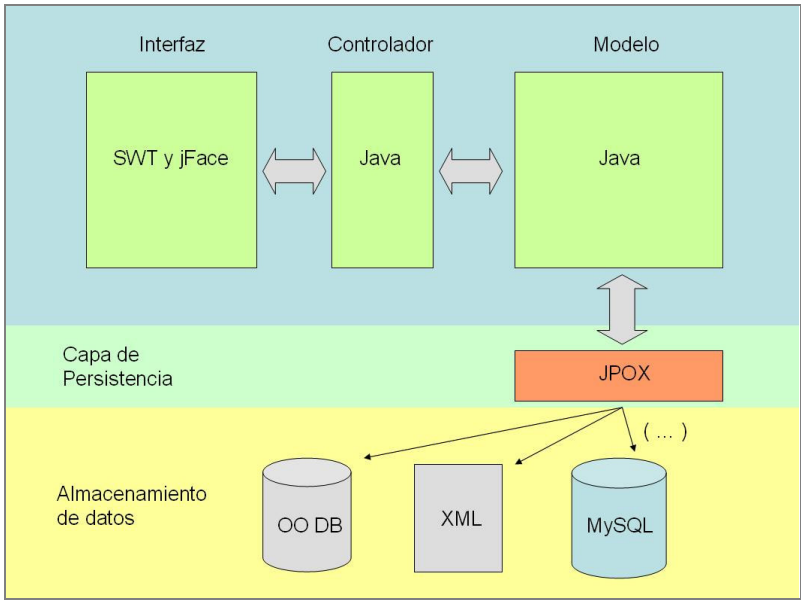


**Figura 5-2 Visión arquitectónica**

Se consideró la utilización del estándar XMI para la extracción de la información, de manera de posibilitar la eventual portabilidad de la herramienta a diferentes ambientes de diseño y desarrollo.

Se decidió utilizar para la persistencia de la información el framework de JPOX (Java Persistent Objects) [JPOX], que implementa las interfaces de persistencia JDO y JPA, permitiendo independizar el tipo de la base de datos y permitiendo realizar consultas en SQL, JDOQL y JPQL.

## 5.2 Integrando el plugin con "Rational Software Architect"

Como se mencionaba en el capitulo introductoria, el objetivo final del desarrollo es desarrollar una nueva herramienta CASE, capaz de contemplar las mejores características según el estudio realizado de las herramientas en el mercado y que además logre implementar los conceptos definidos en la Tesis de doctorado de G. Robiolo. Este desarrollo supuso varios desafíos, algunos estrictamente técnicos, otros relacionados con estrategias de integración y usabilidad.

El principal desafío técnico fue entender Eclipse como framework para desarrollar aplicaciones como plugins y como luego integrarlos en Eclipse y RSA (Racional Software Architect).

Luego, se requirió comprender el ambiente de diseño del RSA, para integrar la herramienta dentro de este ambiente, explotar sus características e interactuar con el.
La comprensión del ambiente de diseño del RSA tuvo como objetivo final encontrar una estrategia de integración e interacción entre el Racional Software Architect y la herramienta a desarrollar.

Durante el desarrollo, se presentaron dificultades que supusieron la toma de decisiones claves para el éxito del trabajo. Estas dificultades se mencionan abajo. Es importante que queden documentadas para su uso en situaciones similares.

## 5.3 Algunas dificultades encontradas

### 5.3.1 Incorporación de estados a las clases en el ambiente de diseño

Uno de los principales problemas tuvo que ver con implementar una característica presente en la Tesis de G. Robiolo, donde se definen estados a las Clases en el diagrama de clases. Por ej. Una clase puede estar en estado "No Finalizada", "Re-hecha", "Finalizada". Esto supone tener que entrar en la parte del ambiente de diseño para agregar una funcionalidad donde se permita elegir un estado y asociársela a la clase, y aun mas, se debería tener en cuenta que XMI (el formato estándar para intercambio de diagramas) debería contemplar la información de los estados.

Bien, la solución adoptada y más adecuada fue el uso de **estereotipos**. Como se definió en el capitulo de conceptos previos, un estereotipo básicamente permite enriquecer semánticamente a los diagramas UML, extendiéndolo, para que soporten nuevas características. Los estereotipos están soportados por el estándar XMI.

En el RSA, los estereotipos se definen por medio de los "profiles". En este caso, se definió un profile con un estereotipo que extiende a las clases, definiendo 3 posibilidades: "No Finalizada", "Re-hecha" y "Finalizada". Este "profile" fue exportado a un archivo "EstadoClase.profile.uml" y debe ser aplicado al proyecto en el cual se quiere tener estas características de extensión.

### 5.3.2 Extracción de información del modelo

Otra problema que planteo una incógnita en su momento fue la forma de tomar la información del modelo UML creado por el usuario en el ambiente de diseño RSA. Como se mencionaba anteriormente, XMI es el estándar que permite representar la información de un metamodelo en XML. Ahora, ¿Cómo procesar dicha información? Las alternativas fueron:

- Implementar un parcer.
- Utilizar librerías ya desarrolladas para procesar XMI
- Utilizar el componente MDR (desarrollado por NetBeans)
- Utilizar el componente UML2 (desarrollado por IBM)

Finalmente se decidió por utilizar el componente UML2 [UML2]. Este fue desarrollado por IBM y es utilizado por el RSA para Importar / Exportar archivos en XMI. Las otras alternativas fueron descartadas por diferentes motivos: En el caso de implementar un parcer el esfuerzo a cambio de flexibilidad era demasiado alto. En el caso de utilizar librerías ya desarrolladas el motivo más importante por el que se descartó fue la falta de confiabilidad. Por ultimo, en el caso de MDR, la razón por la que no se utilizó fue el riesgo en la integración con la herramienta a desarrollar y con el RSA.

### 5.3.3 Falta de aceptación completa del estándar XMI

Ya se ha hablado de XMI y su potencial capacidad para independizar a las aplicaciones del formato de representación de los modelos UML. En un comienzo, se presentó como ideal su utilización pensando en mostrar el plugin desarrollado funcionando con

modelos UML diseñados en diferentes ambientes. Luego, se noto que XMI, si bien es un estándar bien definido (ya se encuentra en su versión 2.1), carece de una completa adopción por parte de los fabricantes de herramientas de diseño UML. Si bien las herramientas de diseño mas utilizadas, casi todas implementan la característica de intercambio de diagramas mediante XMI, si se considera el total de ellas, solo un pequeño conjunto poseen esta característica, y de ellas, el pasaje se hace difícil, se generan errores a la hora de realizar esta operación debido a diferentes razones:

- El modelo puede estar en diferentes versiones de UML
- El modelo puede estar serializado en diferentes versiones de XMI
- Se pueden dar las combinaciones de estas dos primeras opciones

Esto mencionado fue la conclusión del testeo en diferentes herramientas el intercambio de diagramas XMI. Se testearon herramientas de diseño UML tanto de código abierto como comerciales, y con diferentes componentes para la extracción de la información de los archivos XMI. Se probaron las siguientes herramientas:

- **"Racional Software Architect" de IBM, Version 7**. (Herramienta comercial que utiliza UML2 como conector con XMI) [RSA]
- **"ArgoUML" v0.24** (Herramienta opensource que utiliza NovoSoft UML API como conector con XMI) *http://argouml.tigris.org/*
- **"Poseidon UML"** (Herramienta comercial que utiliza MDR –de NetBeans- como conector para XMI)
- **"Enterprice Architect" v 6.5** (Herramienta con versiones gratuita y comercial) *http://www.sparxsystems.com.ar/products/ea.html*
- **"Visual Paradigm for UML" 3.2** (Herramienta con versiones gratuita y comercial) http://*www.visual-paradigm.com/product/vpuml/*
- **"Altova UML"** (Herramienta comercial) *http://www.altova.com/products/umodel/uml_tool.html*

Se testeó con gran cantidad de combinaciones entre las distintas herramientas, sin éxito alguno en el intercambio de diagramas, exceptuando el caso entre "Altova UML" y "Visual Paradigm for UML 3.2".

Esto afecta indirectamente al trabajo, ya que limita las fuentes de donde tomar la información, sin embargo, es un problema ajeno. El sistema al utilizar el componente UML2 supone la potencial capacidad de lectura de archivos XMI generados desde cualquier herramienta de diseño y modelado UML.

### 5.3.4 Manteniendo la consistencia entre la aplicación y XMI

Una de las características de la herramienta desarrollada es que guarda información del modelo en cada medición que se toma. De esta manera, se dispone de información historia de los valores de calidad y porcentaje de avance del sistema en cada etapa. Dicho de otra manera, se toma una fotografía del sistema para almacenarlo como una versión a medir.
Esto supone guardar información en la base de datos de los casos de uso involucrados, así también como información general del modelo en asociación con un registro de medición.

Ahora, puede darse la siguiente situación: El usuario genera una medición con un determinado conjunto de casos de uso involucrados: c1, c2 y c3. En este momento, estos son los tres casos de uso que el sistema registra como activos en la base de datos. Sobre ellos se ingresarán horas trabajadas por parte de los usuarios. Ahora, el usuario, en el ambiente de diseño, realiza cambios en dichos casos de uso: elimina c1, cambia el nombre de c2 por un error de tipeo, cambia el nombre a c3 por uno completamente distinto y agrega un caso de uso c4. Estos cambios impactan en la herramienta desarrollada y deberían reflejarse de la siguiente manera:

- Eliminar c1 como caso de uso actual vigente, pero mantenerlo en la historia
- Considerar que c2 sigue siendo el mismo caso de uso por más que se le haya cambiado el nombre.
- Considerar que c3 ya no existe mas en el sistema, eliminarlo, e incorporar c3' (modificado) como nuevo caso de uso.
- Incorporar c4 como nuevo caso de uso.

En todos los casos, el único elemento del cual se dispone para comparar es el nombre del caso de uso. La tarea de descubrir que un caso de uso debe ser eliminado es trivial: simplemente notar que un caso de uso esta presente en un conjunto pero no en otro. De forma similar se puede agregar un nuevo caso de uso. La incógnita entonces es, ¿Cómo determinar que un caso de uso es el mismo cuando se le cambia el nombre? O bien, ¿Cómo determinar que un caso de uso no es el mismo cuando se le cambia el nombre?

Bien, la solución tomada combina decisión por parte del usuario soportado por funcionalidad del sistema que procesa las palabras y da sugerencias. Esta funcionalidad se ha llamado "Sincronización con XMI", y sirve tanto para dar de alta automáticamente los casos de uso en un proyecto, cuando este ultimo esta vació, como

también para actualizar el conjunto de casos de uso sincronizándolo con el conjunto de casos de uso proveniente del archivo XMI.
Para lograr implementar la ayuda de la herramienta al usuario se necesitaba poder determinar que tan diferentes, o que tan distantes son dos palabras. Para esto, se recurrió a la utilización del concepto de "la distancia de Levenshtein"

**Distancia de Levenshtein**

Se llama Distancia de Levenshtein o distancia de edición el número mínimo de operaciones requeridas para transformar una cadena de caracteres en otra. Se entiende por operación, bien una inserción, eliminación o la substitución de un carácter. Esta distancia recibe ese nombre en honor al científico ruso Vladimir Levenshtein, quien se ocupara de esta distancia en 1965. Es muy útil para el caso de los correctores de ortografia. La complejidad del algoritmo es O(m*n), donde n y m son las longitudes de cadena1 y cadena2.

La función "distancia de Levenshtein" devuelve un entero que da una idea de la distancia (o parecido) entre ellas. Por ejemplo, la distancia entre "HOLA" y "AROLA" es 2, ya que hay que hacer 2 operaciones sobre la palabra "HOLA" para obtener "AROLA": sustituir la H por una R e insertar una A. Por lo tanto, cuanto más corta es la distancia entre las dos cadenas, más parecidas son. Si la distancia es 0, las dos palabras son iguales.

Utilizando este concepto se logro determinar cuando dos palabras son lo suficientemente parecidas como para pedirle al usuario que confirme que son la misma. De esta manera se saltean todos aquellos casos donde es muy probable que las palabras no tengan nada en común y se refieran a casos de uso diferentes. Aún con un grado alto de aciertos, siempre se pide al usuario la confirmación, dado que siempre es el quien sabe si realmente se refiere al mismo concepto. El algoritmo de identificación de similitud es simplemente una herramienta para alivianarle, y mucho, el trabajo.

Si bien el algoritmo es O (m*n), la implementación final tiene una modificación para que en la practica se perciba una ejecución mucho mas rápida, ya que se pasa un parámetro adicional n que indica dejar de procesar la palabra cuando la distancia sea mayor a ese n. De esta manera se deja de comparar palabras que se sabe, ya no tienen similitud a ese punto.

# 6 Descripción del sistema resultante

En esta sección se describen los resultados obtenidos, las características del producto que finalmente se implementaron y los tiempos empleados. Finalmente, se presenta un caso de prueba con el que se analiza un sistema, su evolución y como se hace el seguimiento del mismo con el plugin desarrollado. Me gustaría remarcar los siguientes puntos sobre el resultado obtenido:

- Para la elicitacion de requerimientos se realizaron 3 reuniones con la Mg. Gabriela Robiolo de la Universidad Austral, donde se definieron los requerimientos y gran parte del diseño. El tiempo destinado al análisis fue el 30% del total empleado para la ejecución de todo el proyecto.

- El tiempo total estimado del proyecto fue de 1 año, y pudo ser cumplido sin demoras.

- Se desarrollaron los 11 casos de uso planteados anteriormente, logrando un producto final que cumple plenamente con las especificaciones y las expectativas depositadas.

- Se logró una buena estrategia de integración con el "Rational Software Architect" y también con el Eclipse.

## 6.1 Guía de uso del sistema desarrollado

A continuación se describe el sistema desde la perspectiva del usuario. Se intentará dar una guía sobre como usar y acceder a la funcionalidad que ofrece.

1. **Accediendo al sistema**

Para acceder al sistema se debe activar la perspectiva "Seguimiento y control". Esto activara las vistas correspondientes al menú e ítems en la barra principal.

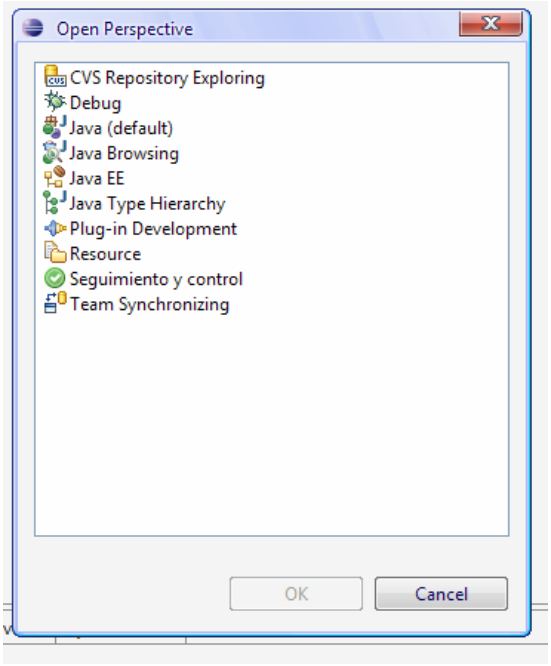


**Figura 6-1 Accediendo al sistema**

Luego, para poder loguearse y desloguearse del sistema, se deben tener en cuenta los iconos que están en el menú principal que se muestran al costado  . Es lo mismo si se accede mediante el menú "Seguimiento y control" -> Iniciar Sesión, o "Seguimiento y control" -> Salir del sistema de seguimiento y control. Cuando se presione en dicha acción se abrirá el siguiente dialogo:

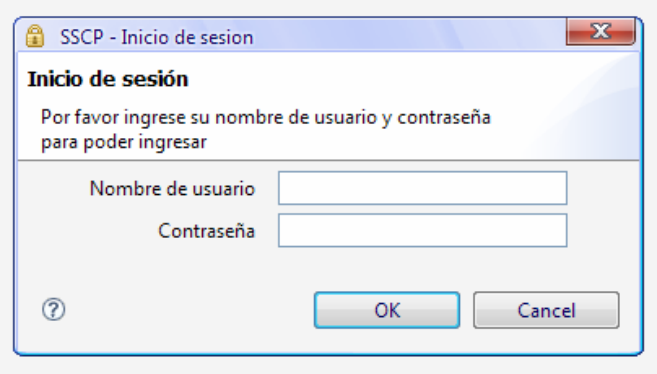


**Figura 6-2 Inicio de sesión**

Una vez que se ha ingresado al sistema con un usuario valido, se muestra los proyectos visibles para ese usuario en la vista "Proyectos", como se muestra a continuación:

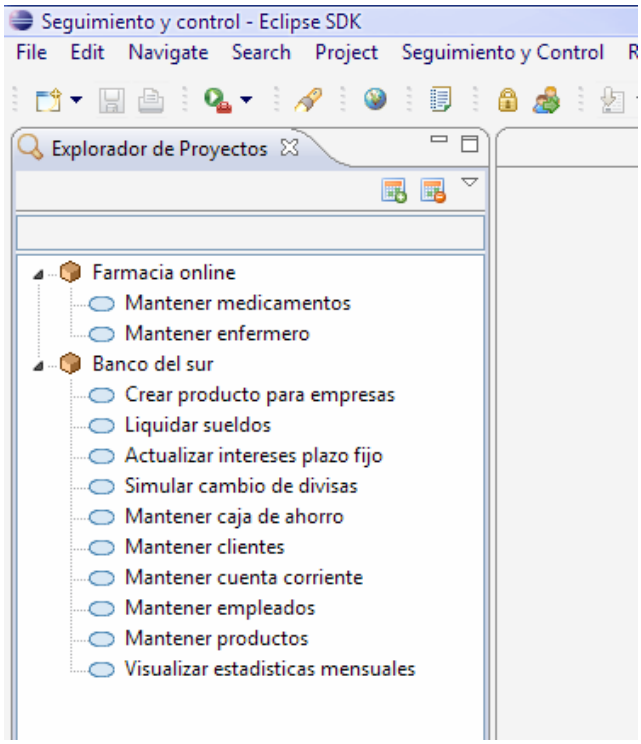


**Figura 6-3 Explorador de proyectos**

Este panel es el eje central del sistema. Mediante esta interfaz donde se muestran los proyectos (con el icono ), y los casos de uso del proyecto (con el icono ), se accede a toda la funcionalidad del sistema.

Se debe tener en cuenta también el menú que esta en la parte superior de la vista, quien contiene acciones que solo podrán ser realizadas por el administrador:

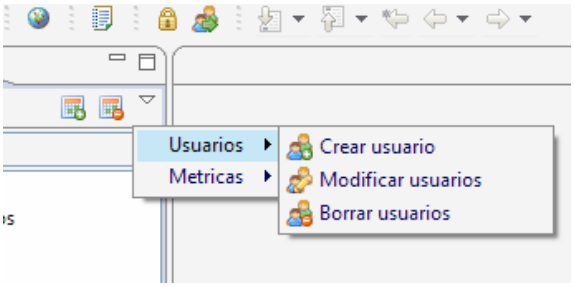


**Figura 6-4 Menú de administración**

## 2. Agregando, modificando y borrando proyectos

Para poder manejar proyectos (crear y borrar), se debe tener privilegios de administrador. Para ello, se debe loguearse con nombre de usuario admin y contraseña de administrador. En la vista "Explorador de proyectos", se debe clickear en el siguiente botón:

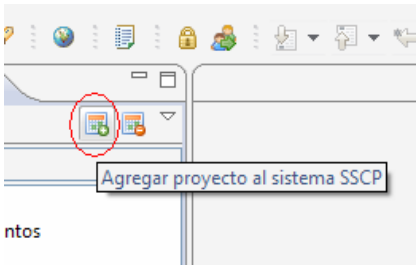


**Figura 6-5 Agregar proyecto**

El mismo se encuentra en la parte superior del Explorador de proyectos. Cuando se clickee, aparecerá un dialogo como el siguiente, donde se podrán cargar datos referentes al proyecto:

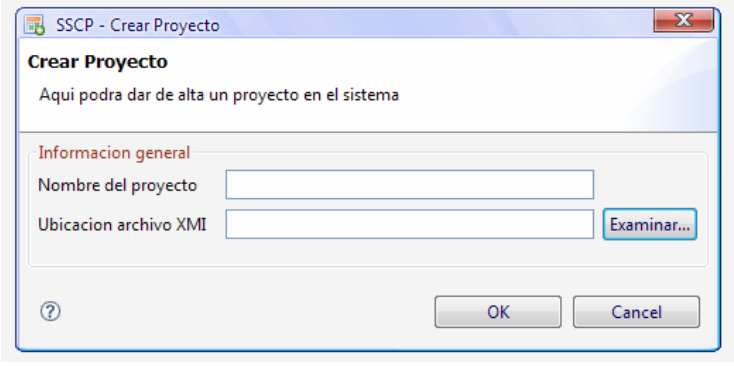


**Figura 6-6 Completando campos del proyecto**

Para poder borrar un proyecto, se debe navegar por el Explorador de proyectos hasta encontrar el proyecto deseado, clickear con el botón derecho y presionar en "Borrar proyecto", como se muestra a continuación:

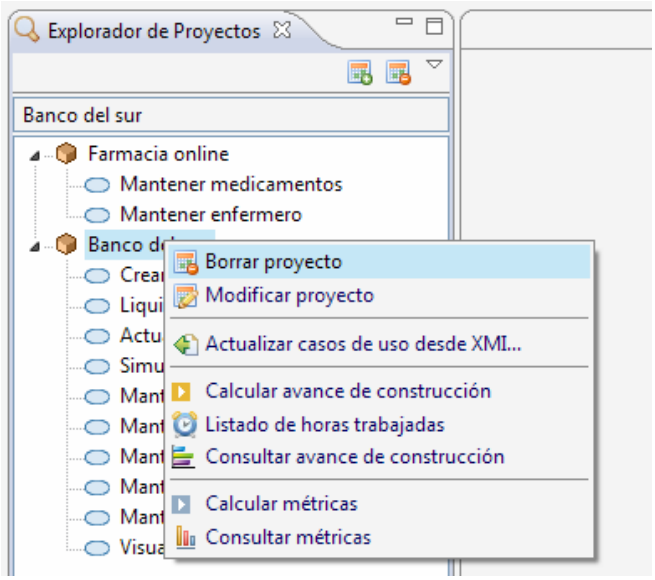


**Figura 6-7 Borrar proyecto**

También se puede eliminar un proyecto presionando en el siguiente botón:

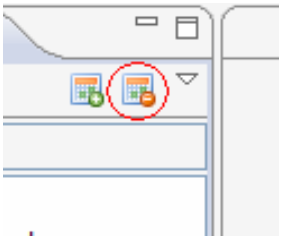

**Figura 6-8 Boton para borrar proyecto**

Nota: Esta operación en el menú contextual esta disponible solo si el usuario esta logueado como administrador. El administrador puede también modificar el proyecto, así como realizar el resto de las operaciones destinadas a usuarios comunes. El dialogo para modificar un proyecto aparecerá al clickear en "Modificar proyecto" en el menú contextual del "Explorador de Proyectos":

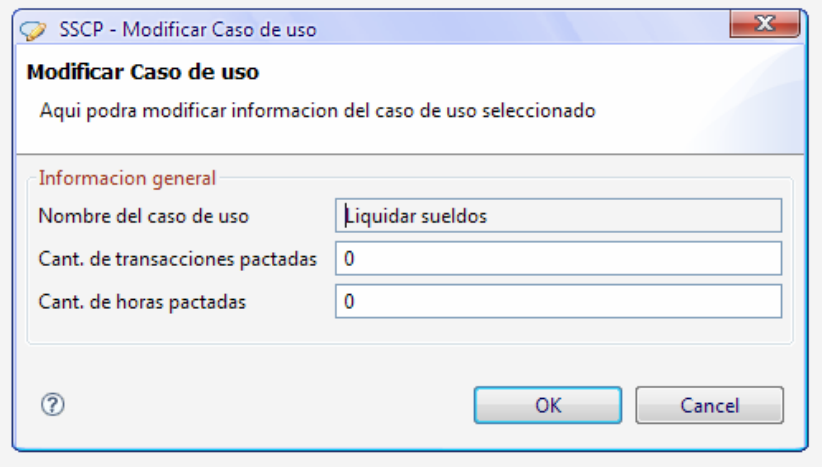


**Figura 6-9 Modificar caso de uso**

### 3. Agregando, borrando y editando usuarios

Para agregar usuarios al sistema, el usuario que va a ejecutar esta acción debe estar logueado como administrador. Luego, se debe acceder mediante el menú ubicado en la parte superior del Explorador de proyectos.

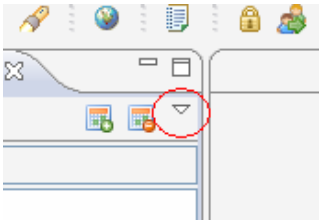

**Figura 6-10 ABM de usuarios**

Clickeando como se muestra en la figura, se accede al menú donde se manejan usuarios y métricas. Para crear un usuario acceder por: Usuarios -> Crear usuario. Aparecerá un wizard como se muestra a continuación que consta de dos pasos:

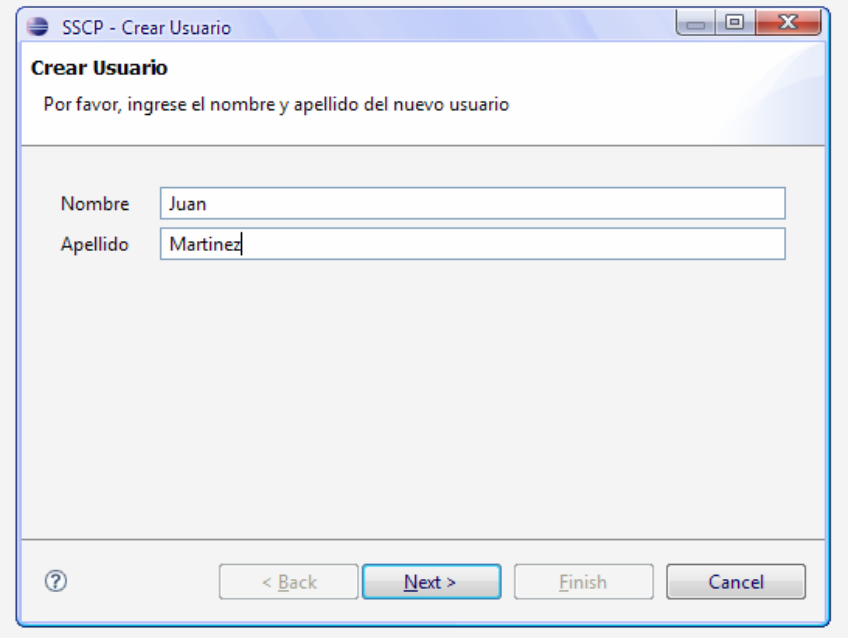


**Figura 6-11 Completando campos del usuario**

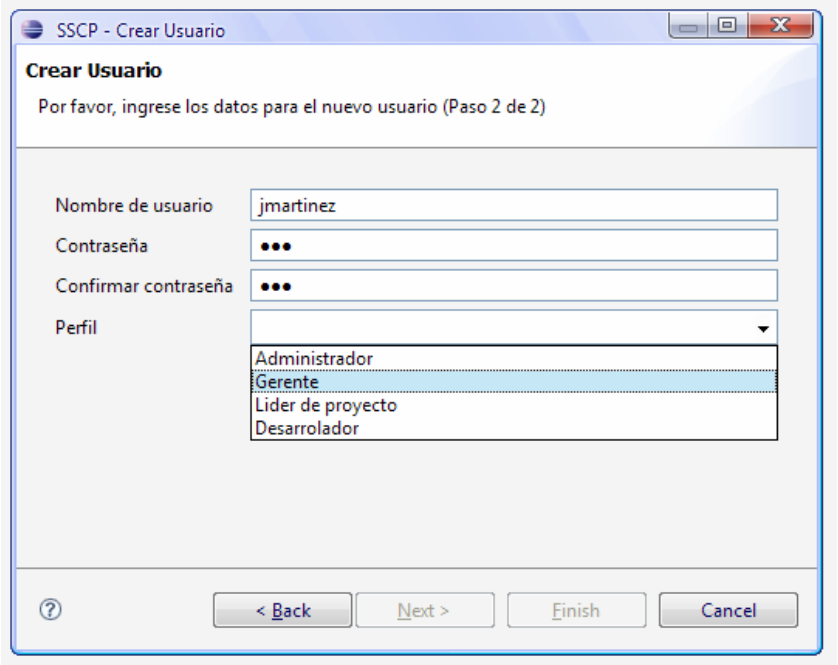


**Figura 6-12 Completando paso 2 en el alta de usuario**

Así se termina el proceso para de dar de alta un usuario. Para editar y borrar usuarios el proceso es similar. Se accede de la misma manera, Usuarios -> Editar usuario. Aparecerá el siguiente dialogo:

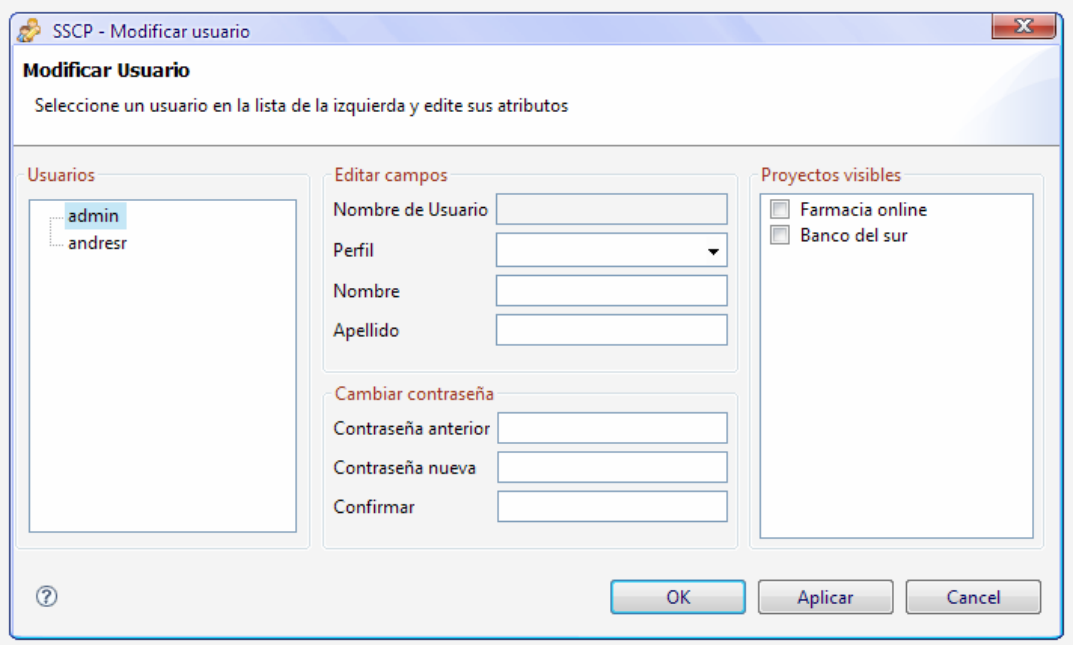


**Figura 6-13 Modificar usuario**

Se debe clickear en el usuario a modificar desde la lista de la izquierda y luego editar el campo deseado en la parte de la derecha. Básicamente puede editarse los atributos del usuario y su visibilidad a los distintos proyectos del sistema.

Para borrar un usuario se accede mediante Usuarios-> Borrar usuarios. Aparecerá un dialogo como se muestra a continuación:

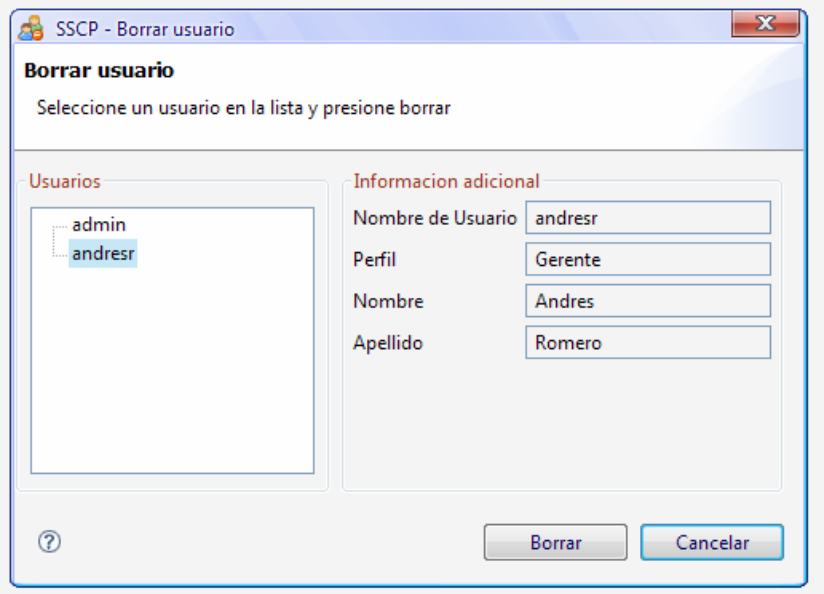


**Figura 6-14 Borrar usuario**

Es un dialogo muy simple donde se selecciona un usuario de la lista de la izquierda, se muestra información acerca del mismo para tener certeza de que el usuario seleccionado es el correcto, y se puede borrar clickeando en el botón "Borrar" que aparece en la parte inferior.

### 4. Ingresando horas trabajadas

Para ingresar las horas trabajadas, una vez logueado con el usuario que registrara las horas, se debe clickear con el botón derecho sobre el caso de uso sobre el que se ha estado trabajando y luego en "Ingresar horas trabajadas", como se muestra en la siguiente figura:

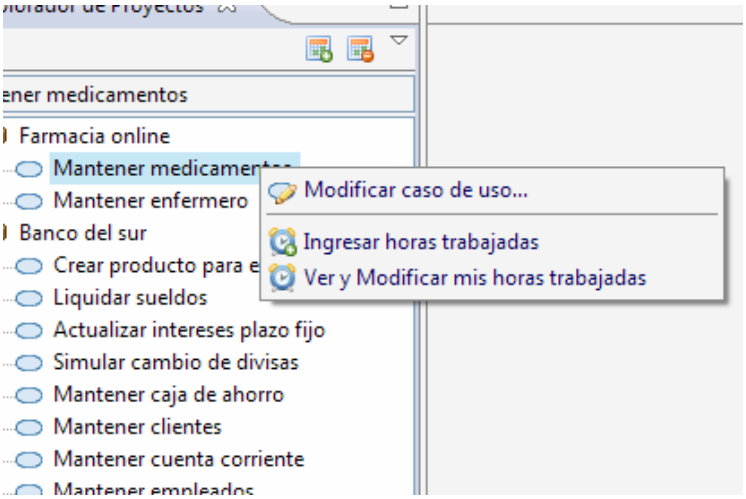


**Figura 6-15 Menu contextual**

Al hacer esto, se abrirá un dialogo que permitirá el ingreso de la cantidad de horas trabajadas. En la parte superior se muestran los últimos ingresos de horas trabajadas para que el usuario tenga noción de cuando fue la última vez que las ingreso.

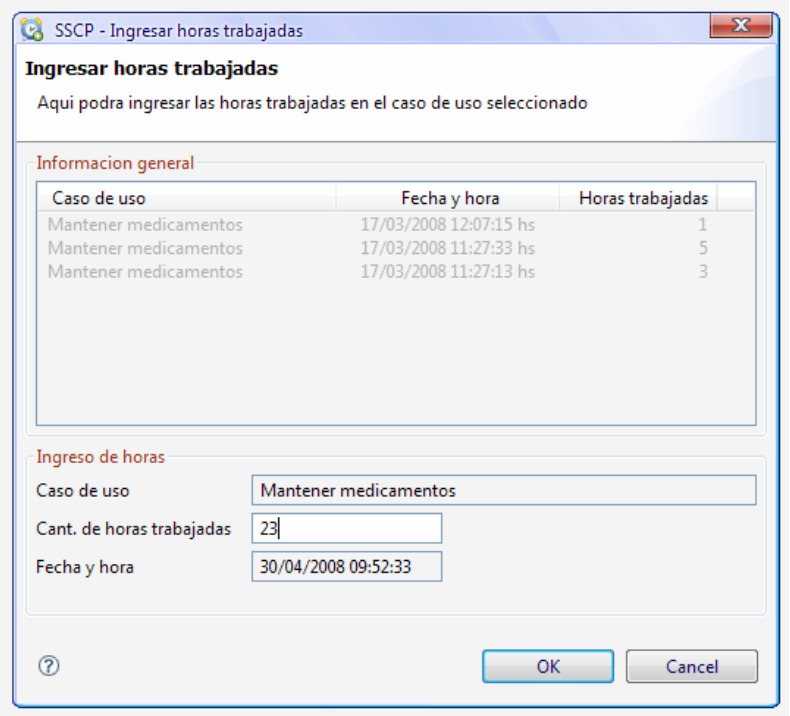


**Figura 6-16 Ingresar horas trabajadas**

### 5. Visualizando y editando horas trabajadas

Si lo que se desea es visualizar y/o editar las horas trabajadas se debe entrar de manera análoga al ingreso de horas, solo que clickeando en "Ver y modificar mis horas trabajadas"

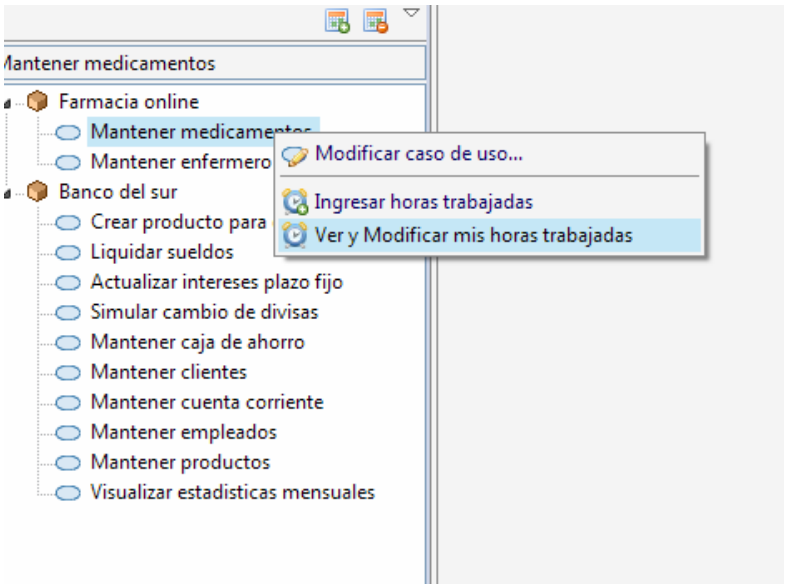


**Figura 6-17 Ver y modificar horas trabajadas**

Se abrirá un dialogo que muestra el listado de horas trabajadas y da la posibilidad de editarlas.

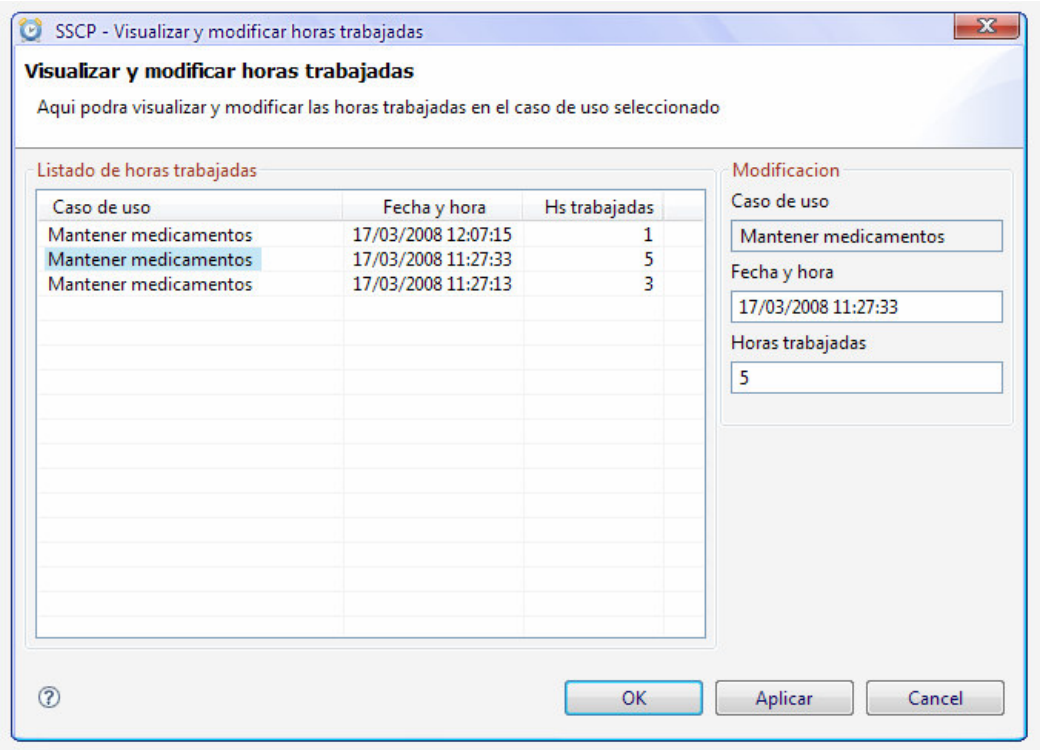


**Figura 6-18 Visualización y modificación de horas trabajadas**

### 6. Listando horas trabajadas

Todo usuario tiene la capacidad de poder visualizar las horas trabajadas de los usuarios con un perfil con puntaje inferior a el. Los perfiles (de mayor a menor puntaje) son: administrador, gerente, líder de proyecto, desarrollador.

Para acceder a esta funcionalidad, se debe seleccionar el proyecto sobre el que se quiere visualizar las horas trabajadas, hacer clic derecho y clickear en "Listado de horas trabajadas", como se muestra a continuación:

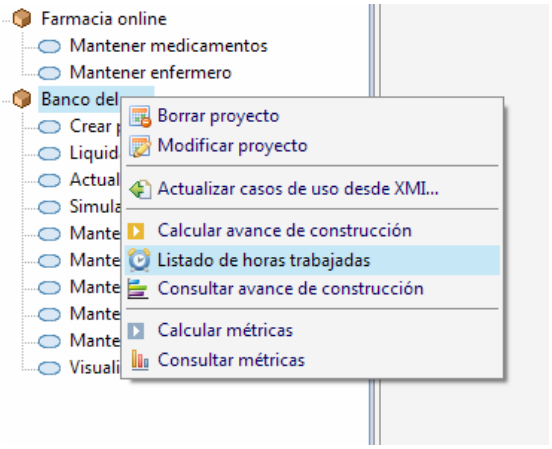


**Figura 6-19 Obtener listado de horas trabajadas**

Entonces, se abrirá un cuadro donde se puede elegir el usuario de quien ver las horas trabajadas, y al seleccionarlo un listado de las mismas:

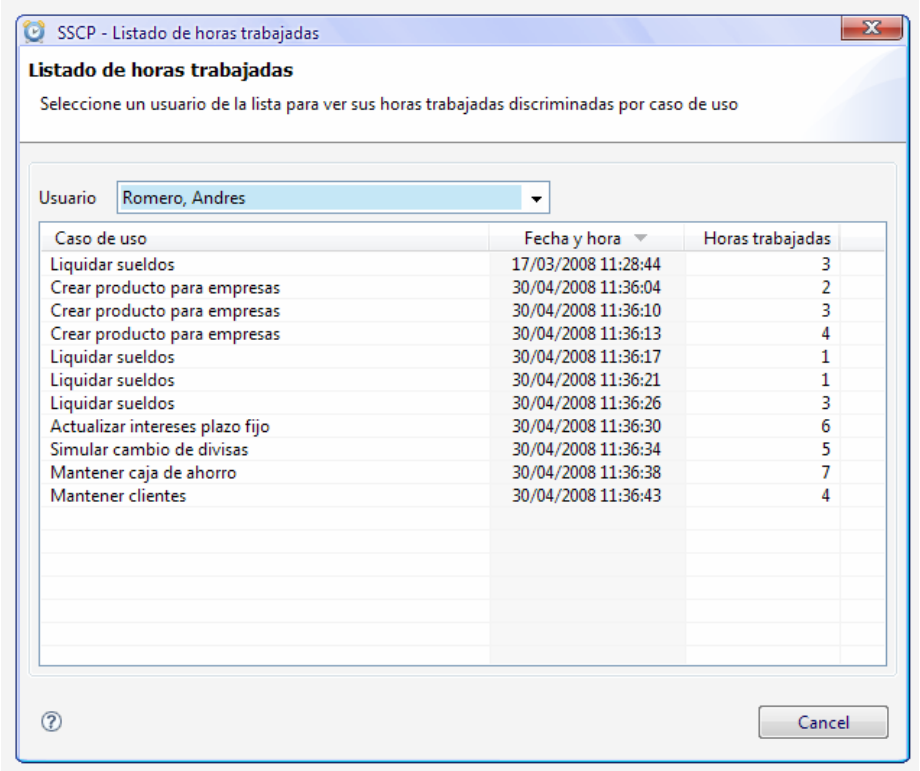


**Figura 6-20 Listado de horas trabajadas**

Nota: Recordar que solo se mostraran aquellos usuarios que tengan un perfil con puntaje inferior al del usuario logueado.

## 7. Modificando el estado de las clases

Esto, como se mencionaba en capítulos anteriores, no es una funcionalidad propia del sistema sino parte de una estrategia de solución que requiere ciertas convenciones en el ambiente de diseño. El estado de las clases, en el Racional Software Architect deberá ser cambiado mediante el uso de Profiles y Estereotipos.

Para tener habilitados los estereotipos requeridos se necesita configurar el proyecto. Para esto, se debe hacer clic derecho sobre el proyecto de modelado -> propiedades UML y agregar una entrada de un perfil como se muestra a continuación:

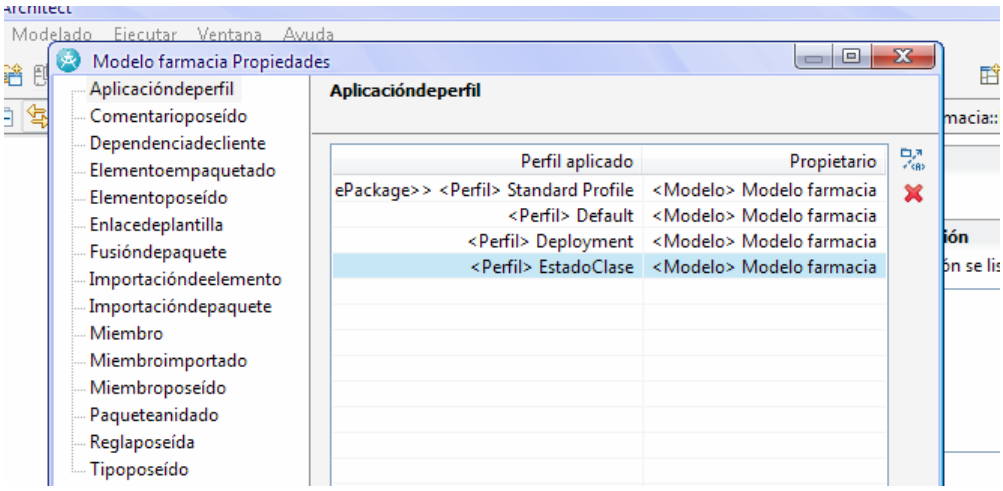


**Figura 6-21 Agregar un perfil al Rational Software Architect**

Para mayor comodidad del usuario, el perfil ya se encuentra en un archivo cargado con los estereotipos para que pueda ser rehusado.

Para agregar una clase con estereotipo se debe seleccionar el botón "Clase Estereotipada en el ambiente de diseño" y elegir el estado deseado.

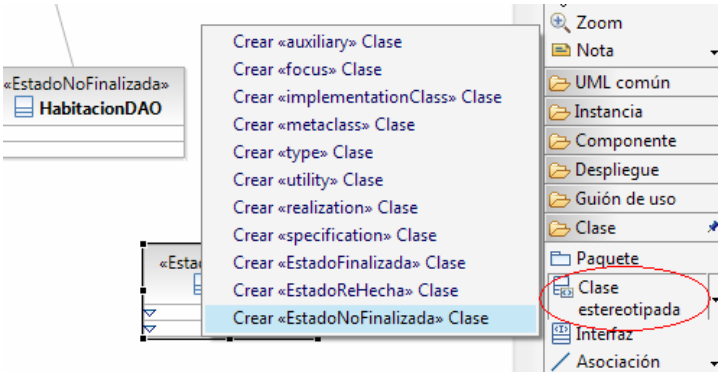

**Figura 6-22 Agregar una clase estereotipada**

Finalmente, para cambiar el estereotipo, se debe ir a Propiedades -> Estereotipos -> Aplicar Estereotipos y se abrirá una pequeña ventana como se muestra en la siguiente figura:

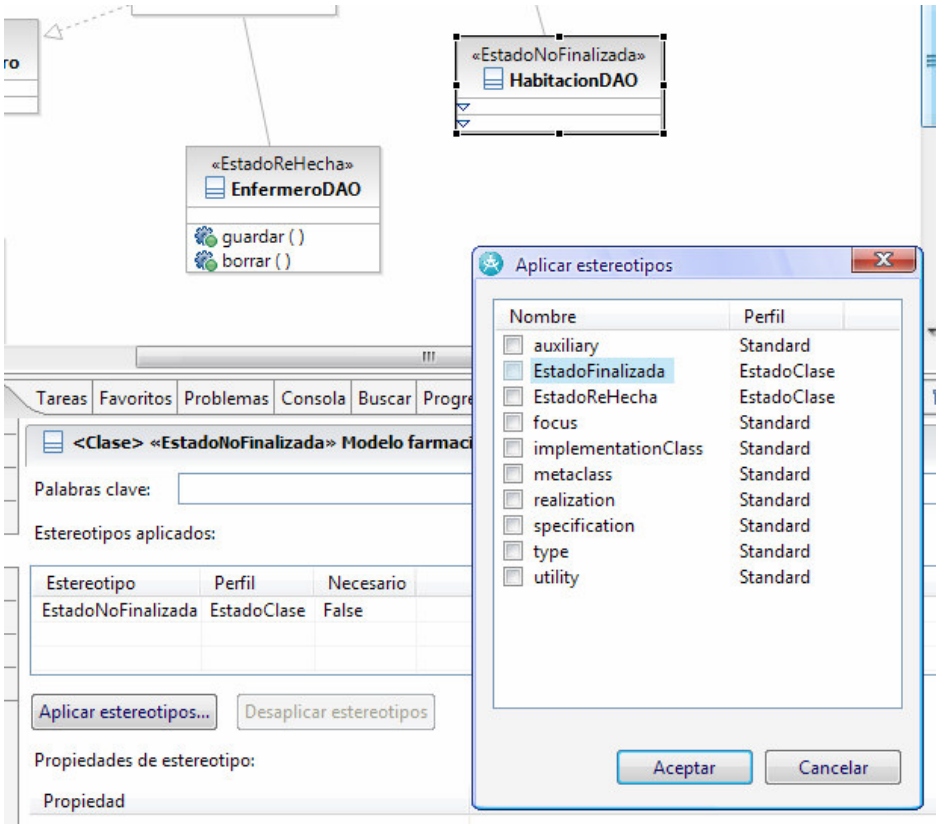


**Figura 6-23 Aplicar estereotipos a clase**

## 8. Sincronizando casos de uso

Para sincronizar los nombres de los casos de uso del sistema con los nombres de los casos de uso provenientes del archivo .uml (XMI), hay dos formas. Una es realizando la sincronización de manera explicita, mediante el menú principal, como se muestra en la siguiente figura:

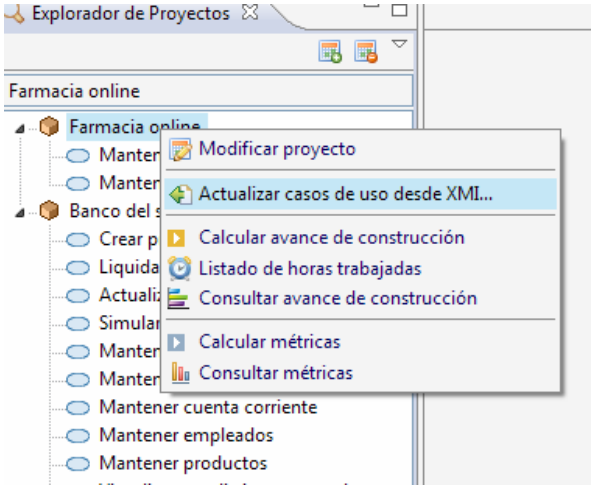


**Figura 6-24 Actualizar casos de uso desde XMI**

Esto hará que el sistema comience a buscar diferencias entre ambos conjuntos y comience a pedir ayuda al usuario para determinar si dos nombres de caso de uso son el mismo.

La otra forma, es ejecutando la tarea "Calcular avance de construcción", la cual exige como paso previo la sincronización. La misma se describe a continuación.

### 9. Calculando el avance de construcción de proyecto

Esta es una de las funcionalidades principales del sistema. Para calcular el avance de constricción de un producto es muy importante:

- Tener el producto con casos de uso definidos
- Tener horas trabajadas cargadas en los casos de uso
- El archivo XMI debe contener información de clases con estados asociados

Si no se cumple alguna de las condiciones citadas el cálculo no será consistente. Queda en responsabilidad del usuario preparar el terreno para lograr un cálculo exitoso. Una vez que se cuente con estos prerrequisitos, se puede calcular el avance de construcción accediendo mediante el menú contextual del proyecto que se desea calcular y clickeando en "Calcular avance de construcción".

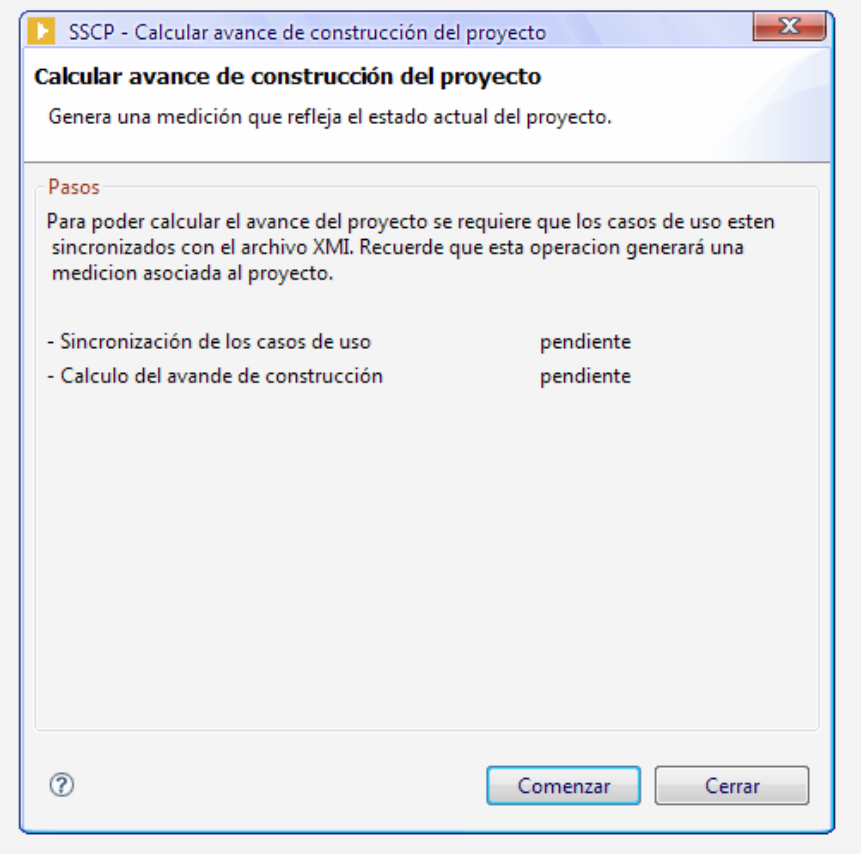


**Figura 6-25 Calcular avance de construcción del proyecto**

Apretando en comenzar se ejecuta la sincronización primero, y luego el calculo en si. A medida que se van completando las tareas se van mostrando con un indicador en color verde.

### 10. Consultar avance de construcción de proyecto

Una vez calculado el avance, se puede consultar el resultado clickeando con el botón derecho sobre el proyecto para que aparezca el menú contextual principal y entrando en "Consultar avance de construcción".

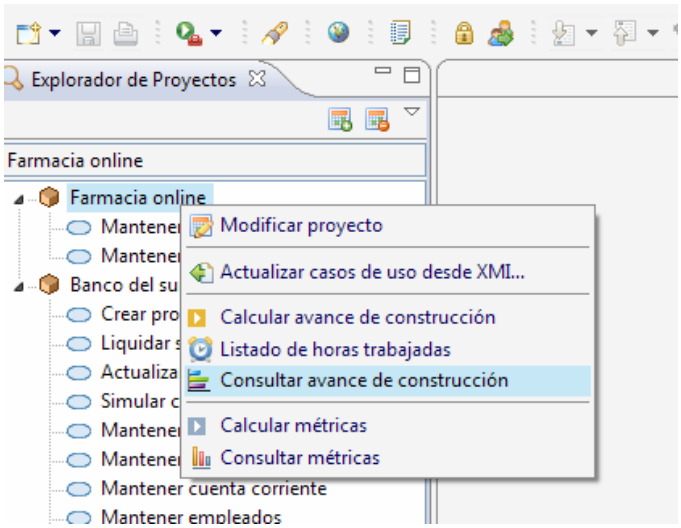


**Figura 6-26 Consultar avance de construcción**

Así, se abrirá un dialogo donde se muestran las diferentes veces que se calcularon las métricas. Cada toma de métrica tiene asociada la fecha que la identifica. Aquí, se puede seleccionar una o varias opciones.

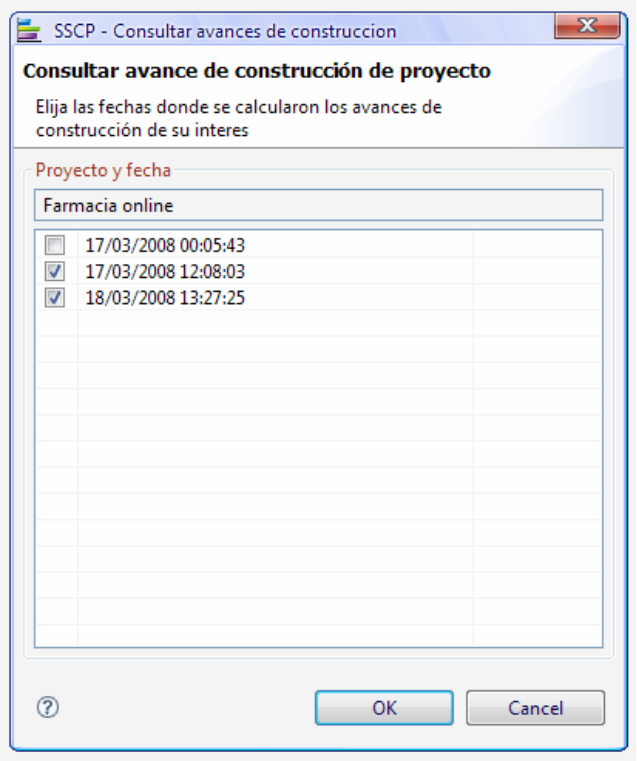


**Figura 6-27 Listado de mediciones de avance**

Seleccionando varias opciones permite luego comparar los valores en una misma ventana, mientras que también se puede seleccionar una sola opción para que muestre

los valores de esa medición. Presionando en "OK", aparecerá el siguiente cuadro, que muestra las métricas de tamaño asociadas a la medición:

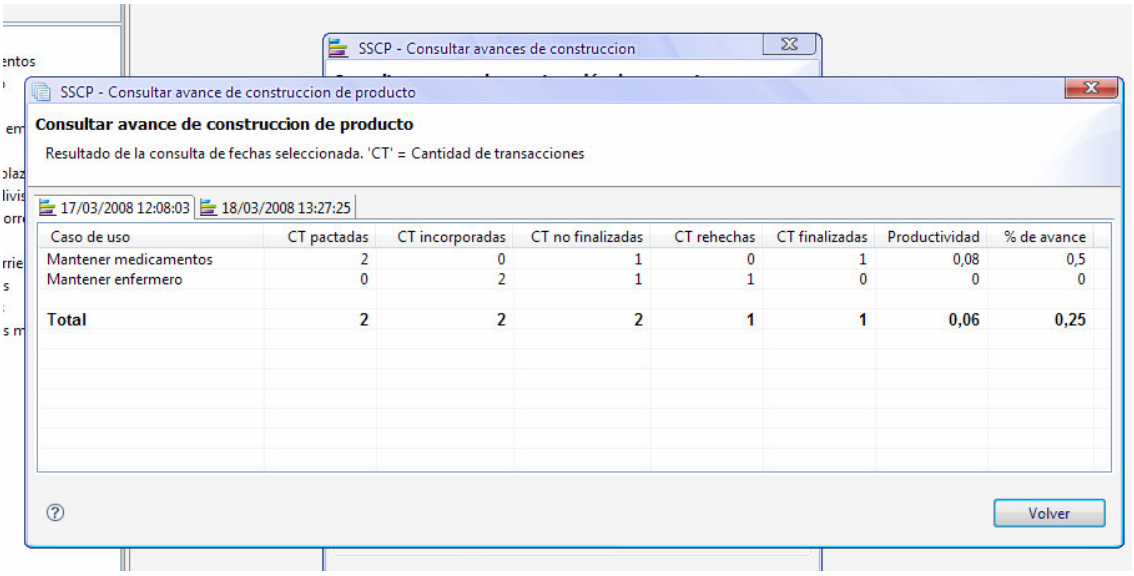


**Figura 6-28 Cuadro general de monitoreo**

Por el hecho de haber seleccionado varias opciones, en este caso aparecen dos pestañas que permiten ir rápidamente de una medición a otra:

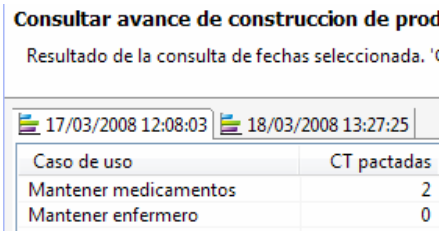


**Figura 6-29 Pestañas con diferentes mediciones**

## 11. Agregando, borrando y editando métricas

La tarea de agregar, editar y borrar métricas solo puede ser realizada por el administrador. Para acceder a estas funcionalidades, debe loguearse y luego clickear sobre el menú superior del Explorador de Proyectos, como se muestra en la siguiente figura:

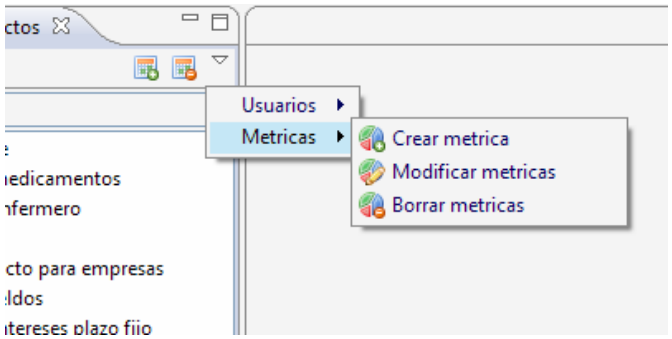


**Figura 6-30 Menu contextual para gestionar las métricas**

Para agregar una métrica se accede en "Crear métrica". Aparecerá un cuadro de dialogo como el que se muestra a continuación:

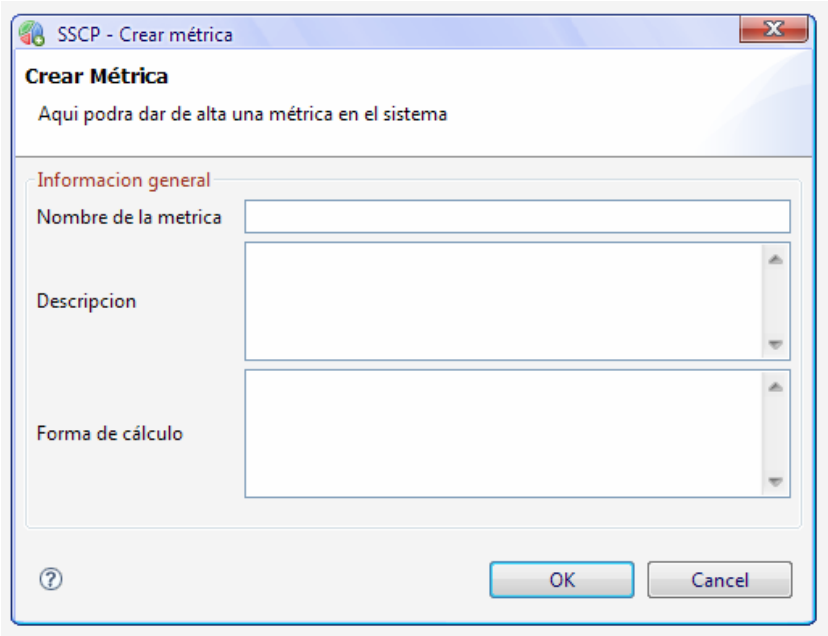


**Figura 6-31 Crear métrica**

Aquí se completa la información sobre la métrica: el nombre, una descripción y lo más importante la forma de cálculo, en forma de una consulta SQL. La consulta SQL deberá tener una consideración especial que se explicará a continuación.

A la hora de modificar métricas el escenario es parecido. Se abrirá un dialogo donde se muestra el listado de métricas y se permite la modificación de sus atributos.

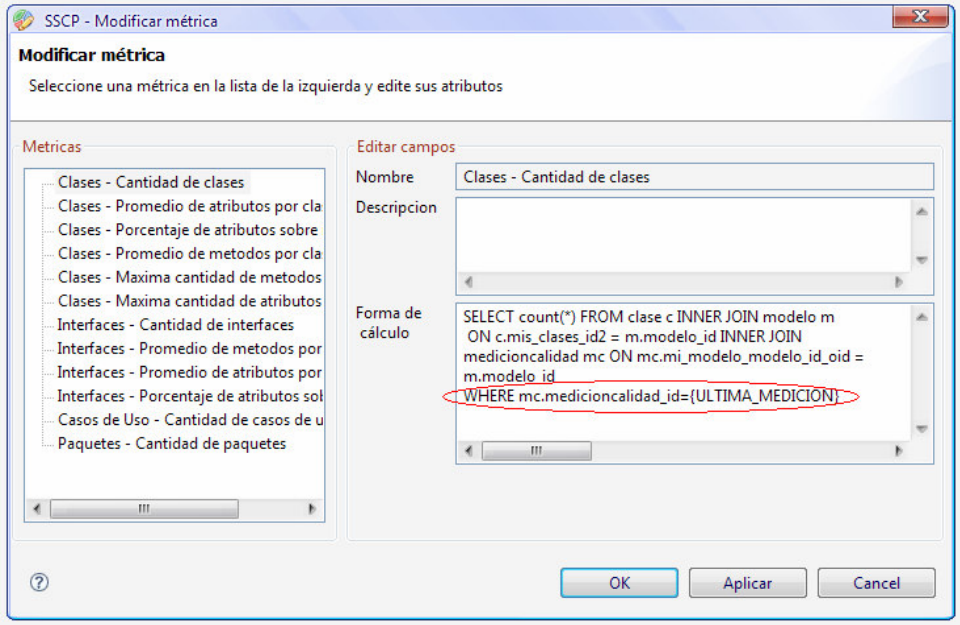


**Figura 6-32 Modificación de la métrica**

La consideración especial que se mencionaba anteriormente tiene que ver con el texto que esta marcado con rojo. Para poder establecer que la medición se realizará sobre el estado actual del sistema se debe incorporar con un join la tabla medición y agregar la condición que se marca en rojo. Para eliminar una métrica se muestra un dialogo muy similar al de modificación, con una lista de métricas sobre la izquierda y los atributos como solo lectura en la derecha.

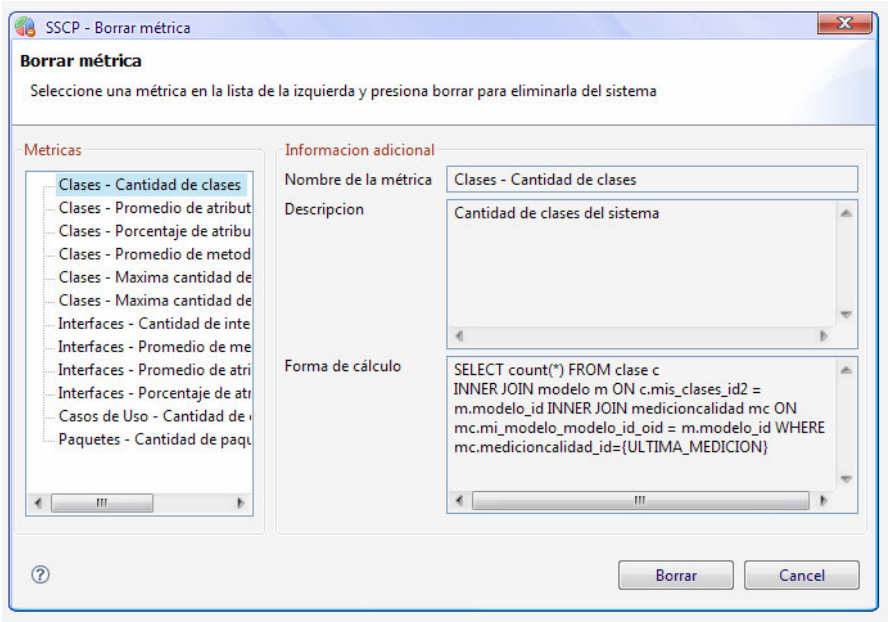


**Figura 6-33 Borrar métrica**

## 12. Calculando métricas de proyecto

Una vez definido el conjunto de métricas y su forma de cálculo, se puede realizar el cálculo de ese conjunto de métricas sobre el estado actual de un proyecto dado. Para ello, se accede mediante el menú contextual en el Explorador de Proyectos y se presiona en "Calcular métricas"

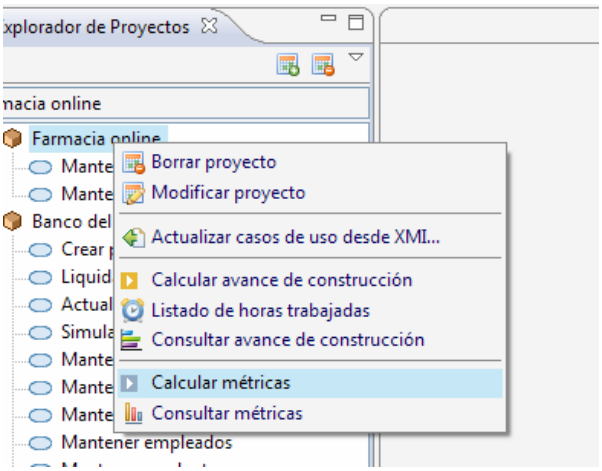


**Figura 6-34 Menu contextual para calcular métricas**

Aparecerá un dialogo donde se pide seleccionar que métricas (del conjunto de métricas disponibles) se desean calcular.

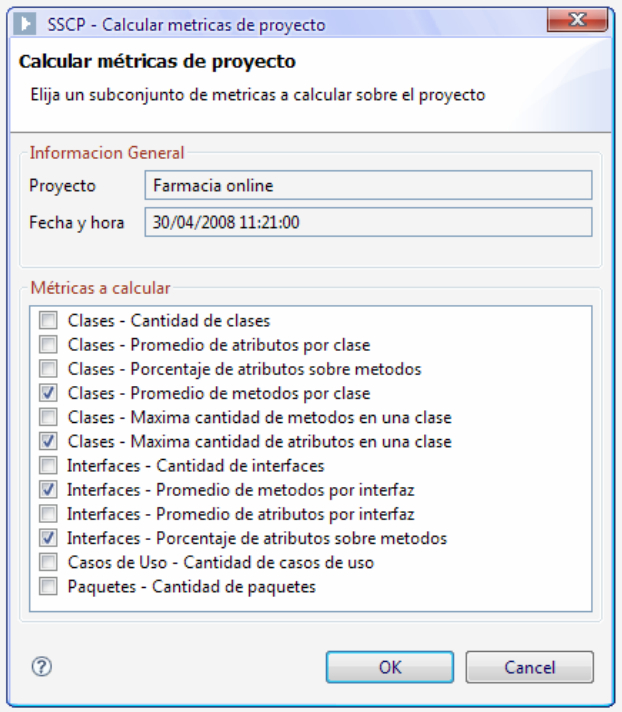

**Figura 6-35 Selección de las métricas a calcular**

Una vez seleccionadas las métricas se presiona en OK, y el sistema realizara el cálculo correspondiente.

### 13. Consultando métricas de proyecto

El paso final es consultar las métricas calculadas previamente. Para ello se accede a "Consultar métricas" en el menú contextual del explorador de proyectos, como se muestra en la siguiente figura:

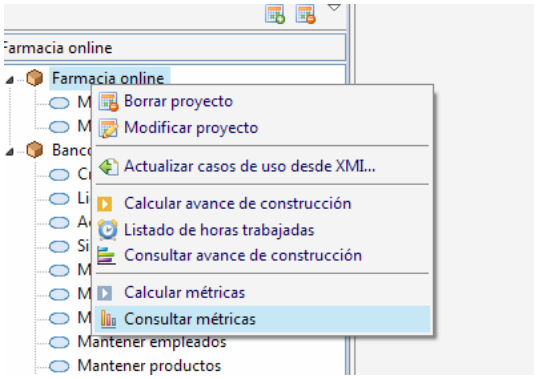


**Figura 6-36 Consultar métricas**

Al hacer clic aparecerá un dialogo análogo al que se muestra para ver las mediciones del avance de construcción. Aquí se requiere marcar que mediciones de calidad se quiere visualizar. Seleccionando varias luego aparecerán pestañas que permitirán comprar los resultados.

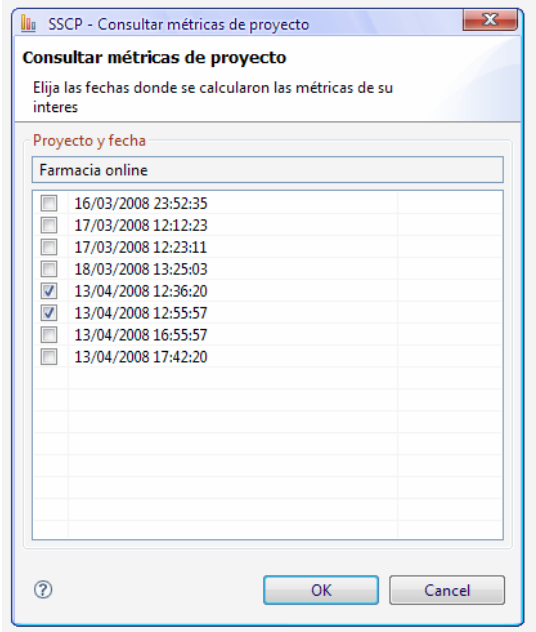

**Figura 6-37 Selección de las mediciones a consultar**

A continuación se muestra el cuadro de resultados, donde se detalle el nombre de la métrica y su valor. Si se seleccionaron varias opciones en la pantalla anterior, entonces aparecerán tantas pestañas como opciones seleccionadas para saltar rápidamente a visualizar los datos de otra medición.

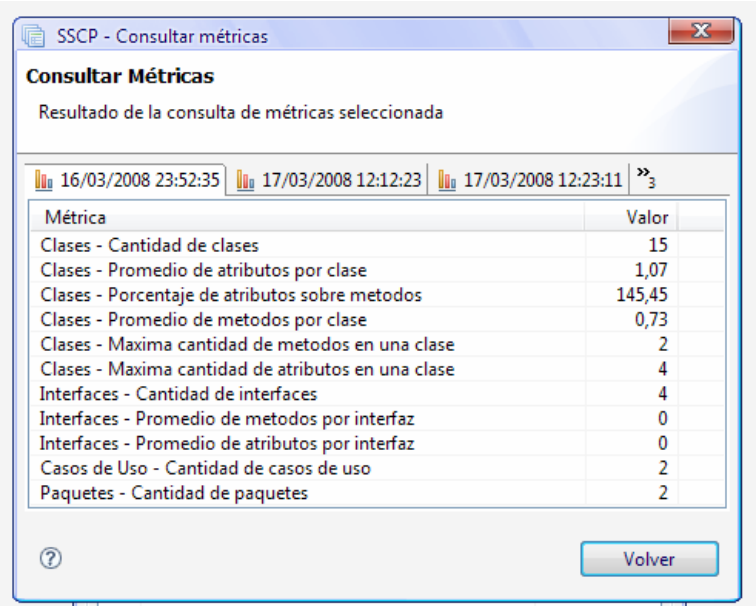


**Figura 6-38 Cuadro de resultados de métricas**

Si las mediciones no entran en el tamaño de la ventana, notar el botón que se encuentra sobre el sector superior derecho:

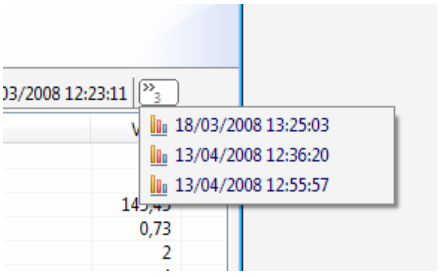


**Figura 6-39 Posibilidad de ver múltiples mediciones en múltiples pestañas**

De esta manera se pueden visualizar las métricas generadas y lograr una rápida comparación con otras mediciones en distinta fecha y hora.

# 7 Conclusiones

Simplemente mencionar que se concluyo exitosamente con el proyecto planteado hace un año atrás. Los resultados fueron los esperados y se pudieron respetar los tiempos estimados.

La herramienta obtenida incluye todos los requerimientos que se plantearon, permitiendo hacer seguimiento de proyectos su revisión de calidad, integrándose con herramientas de uso común como "Eclipse" y "Rational Software Architect".

Haciendo una comparación con otras herramientas existentes en el mercado, dos puntos claves del plugin desarrollado son el background teórico que implementa y el hecho de implementar las características más deseables en su conjunto.
La herramienta desarrollada proporciona un medio para fortalecer los conceptos de "Objetos de entidad" y "Transacciones", como indicadores de peso del sistema en datos y en funcionalidad, respectivamente. También marca conceptos importantes como es la definición de un esquema de implementación que permita que el usuario defina su propio conjunto de métricas. Esto es importante ya que generalmente la calidad es relativa, y depende del contexto donde se aplique. Por esto mismo, la definición de las métricas es una actividad dependiente de los objetivos que se persigan en ese contexto.

El sistema desarrollado también muestra la importancia de la separación del proceso de obtención de la información y el de procesamiento de la misma, para generar mediciones, métricas, indicadores, etc. Para esto fue importante la adopción de XMI, un estándar que permite serializar los diagramas UML en un formato capaz de independizar la fuente de datos de cualquier herramienta de diseño.

Algo que no fue un impedimento, pero afecta al potencial de aplicación de la herramienta desarrollada es la falta de aceptación del estándar XMI. Fueron decepcionantes los resultados encontrados cuando se testeo al intercambio de diagramas con diferentes herramientas. Sin embargo, el desarrollo realizado esta preparado para cuando el estándar reciba la aceptación que merece.

## 7.1 Extensiones a la herramienta

Aquí expongo las potenciales extensiones a la herramienta, que han sido dejadas a un lado cuando se analizo su alcance.

La principal extensión que se puede realizar es sobre la funcionalidad de "Calcular y Consultar métricas". A la hora de calcular las métricas la herramienta captura un subconjunto del modelo que se esta analizando y lo persiste en la base de datos. Este modelo es el objetivo de las consultas SQL que definen las métricas. Esta es la forma que el usuario puede lograr escribir sus propias métricas. Aquí se plantean varias extensiones:

1. Permitir alternativas a la hora de definir las métricas. Una extensión muy simple es ofrecer alternativamente a SQL, poder escribir las consultas en OQL o JDOQL, ya que el diseño de la herramienta esta hecho con JPOX, un framework de persistencia que implementa JDO. Sin embargo, cualquiera de estas formas requieren un usuario bastante técnico. Si se apuntara a un usuario menos técnico, se podría pensar también en ofrecer escribirlas en XML o mediante una interfaz. De cualquier manera, esto hace a la herramienta más amigable, pero no permite generar más cantidad de métricas. El siguiente punto trata de cómo lograr esto último.

2. Aumentar el potencial conjunto de métricas a generar. Para esto sería necesario poder capturar mas elementos del modelo serializado en XMI. Algunos puntos que serian deseables son:

    a. Capturar las jerarquías
    b. Capturar la profundidad de las jerarquías
    c. Capturar los casos de uso y actores, etc.

Otra extensión que sin duda es deseable en un producto como este es la capacidad de visualizar los datos de la mejor manera posible de manera de potenciar los sentidos del usuario. [PTCV] En este punto las posibilidades son muchas y mejor cada día. Además del agregado de gráficos, la generación de reportes también seria una característica muy útil, sobre todo si los desarrolladores, líderes de proyecto o gerentes tienen que reportar a otro sector o a otra persona el avance del proyecto o el nivel de calidad del proyecto.

# 8 Referencias


[FCSMT] Auer Martin, Bernhard Graser, Stefan Biffl. "A Survey on the Fitness of Commercial Software Metric Tools for Service in Heterogeneous Environments: Common Pitfalls". Ninth International Software Metrics Symposium (METRICS'03) p. 144, 2003.

[DAUM] Berthold Daum. "Professional Eclipse 3 for Java Developers", Wiley, 2004.

[SQL] C. J. Date, H. Darwen, A Guide to The SQL Standard, Third Edition, Addison-Wesley Publishing Company, Inc, 1993.

[DI] Diagram Interchange (DI) Especificación Version 2.0 - http://www.omg.org/cgi-bin/apps/doc?ptc/05-06-04.pdf

[GAM] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.

[ECLI] Eclipse (IDE Y Framework de desarrollo de aplicaciones), http://www.eclipse.org

[UML] G. Booch, J. Rumbaugh y I. Jacobson, "El Lenguaje Unificado de Modelado", Addison Wesley, 1999

[SUCC] G. Succi, E. Liu "A Relations-Based Approach for Simplifying Metrics Extraction," ACM Applied Computing Review, ACM Press, 27:32

[JAC] Jacobson, I., Christerson, M., Jonsson, P. and Övergaard, G. Object Oriented Software Engineering, Addison Wesley, Harlow, England, 1992.

[JPOX] Java Persistent Objects (Mapeador Objeto-Relacional que implementa la interfaz JDO). http://www.jpox.org

[AMUM] Luigi Lavazza, Alberto Agostini. "Automated Measurement of UML Models: an open toolset approach". ETH Zurich, Chair of Software Engineering JOT, 2005.

[RASM] Marco Scotto, Alberto Sillitti, Giancarlo Succi, Tullio Vernazza. "A relational approach to software metrics". ACM Symposium on Applied Computing. 2004.

[LIB3] Matthew Scarpino, Stephen Holder, Stanford Ng, and Laurent Mihalkovic, "SWT/JFace in Action", Manning Publications, 2004.

[METRIF] Metri Flame
http://virtual.vtt.fi/virtual/proj1/products/metriflame/

[MYSQL] MySQL (Motor de Bases de datos relacional), http://www.mysql.com

[RSA] Rational Software Architect (Herramienta de análisis, modelado, diseño y desarrollo de aplicaciones), http://www.ibm.com/software/awdtools/architect/swarchitect

[RSA] Rational Software Modeler (Herramienta de modelado y diseño de aplicaciones), http://www.ibm.com/software/awdtools/modeler/swmodeler

[PSMOO] Robiolo Gabriela, Ricardo Orozco. "A Practical Metrics Set for Object Oriented Application". CSITeA-04, 2004.

[UCEEE] Robiolo Gabriela, Ricardo Orosco. "An alternative method employing uses cases for early effort estimation". 3rd IEEE Systems and Software Week. Marzo, 2007.

[MDOO] Robiolo Gabriela, Tesis de Grado en la carrera "Maestría en Ingeniería de Software". "Métricas de Diseño Orientado a Objetos. Aplicadas en Java." Noviembre 2004.

[Chidamber94] S.R. Chidamber and C.F. Kemerer: "A Metric Suite for Object-Oriented Design", *IEEE Trans. Software Eng.*, vol.20, no.6, pp.476-493, June1994.

[SWT] Standard Widget Toolkit, http://www.eclipse.org/swt/

[PFL] Software Metrics, A Rigorous & Practical Approach, PWS Publishing Company, 1997

[SQ] Software Quality: The Elusive Target, Barbara Kitchenham, Shari Lawrence Pfleeger, IEEE, 1996

[UML2] UML2 – Librería desarrollada por IBM, para la representación en memoria y acceso a información de metamodelos. http://www.eclipse.org/uml2

[LI] W. Li and S. Henry. Object-oriented metrics that predict maintainability. Journal of Systems and Software, 23:111.122, 1993.

[PTCV] Warwick Irwin and Neville Churcher. "Object Oriented Metrics: Precision Tools and Configurable Visualisations". 9th International Software Metrics Symposium, September 2003.

[XMI] XML Metadata Interchange (XMI) Version 2.1
http://www.omg.org/technology/documents/formal/xmi.htm

# 9 Anexo I

## Conjunto de métricas implementadas

**Clases - Cantidad de clases**

```
"SELECT count(*) FROM clase c INNER JOIN modelo m
ON c.mis_clases_id2 = m.modelo_id
INNER JOIN medicioncalidad mc
ON mc.mi_modelo_modelo_id_oid = m.modelo_id
WHERE mc.medicioncalidad_id={ULTIMA_MEDICION}"
```

**Interfaces - Cantidad de interfaces**

```
"SELECT count(*) FROM interfaz i INNER JOIN modelo m
ON i.mis_interfaces_id2 = m.modelo_id
INNER JOIN medicioncalidad mc
ON mc.mi_modelo_modelo_id_oid = m.modelo_id
WHERE mc.medicioncalidad_id={ULTIMA_MEDICION}"
```

**Casos de Uso - Cantidad de casos de uso**

```
"SELECT cantidad_casos_de_uso FROM modelo m
INNER JOIN medicioncalidad mc
ON mc.mi_modelo_modelo_id_oid = m.modelo_id
WHERE mc.medicioncalidad_id={ULTIMA_MEDICION}"
```

**Paquetes - Cantidad de paquetes**

```
"SELECT cantidad_paquetes FROM modelo m
INNER JOIN medicioncalidad mc
ON mc.mi_modelo_modelo_id_oid = m.modelo_id
WHERE mc.medicioncalidad_id={ULTIMA_MEDICION}"
```

**Clases - Promedio de metodos por clase**

```
"SELECT avg(cantidad_metodos) FROM clase c INNER JOIN modelo m
ON c.mis_clases_id2 = m.modelo_id
INNER JOIN medicioncalidad mc
ON mc.mi_modelo_modelo_id_oid = m.modelo_id
WHERE mc.medicioncalidad_id={ULTIMA_MEDICION}"
```

**Clases - Promedio de atributos por clase**

```
"SELECT avg(cantidad_atributos) FROM clase c INNER JOIN modelo m
ON c.mis_clases_id2 = m.modelo_id
INNER JOIN medicioncalidad mc
ON mc.mi_modelo_modelo_id_oid = m.modelo_id
WHERE mc.medicioncalidad_id={ULTIMA_MEDICION}"
```

**Clases - Porcentaje de atributos sobre métodos**

"SELECT (sum(cantidad_atributos)/sum(cantidad_metodos))*100 FROM clase c INNER JOIN modelo m
ON c.mis_clases_id2 = m.modelo_id
INNER JOIN medicioncalidad mc
ON mc.mi_modelo_modelo_id_oid = m.modelo_id
WHERE mc.medicioncalidad_id={ULTIMA_MEDICION}"

**Clases - Maxima cantidad de métodos**

"SELECT max(cantidad_metodos) FROM clase c INNER JOIN modelo m
ON c.mis_clases_id2 = m.modelo_id
INNER JOIN medicioncalidad mc
ON mc.mi_modelo_modelo_id_oid = m.modelo_id
WHERE mc.medicioncalidad_id={ULTIMA_MEDICION}"

**Clases - Máxima cantidad de atributos en una clase**

"SELECT max(cantidad_atributos) FROM clase c INNER JOIN modelo m
ON c.mis_clases_id2 = m.modelo_id
INNER JOIN medicioncalidad mc
ON mc.mi_modelo_modelo_id_oid = m.modelo_id
WHERE mc.medicioncalidad_id={ULTIMA_MEDICION}"

**Interfaces - Promedio de metodos por interfaz**

"SELECT avg(cantidad_metodos) FROM interfaz i INNER JOIN modelo m
ON i.mis_interfaces_id2 = m.modelo_id
INNER JOIN medicioncalidad mc
ON mc.mi_modelo_modelo_id_oid = m.modelo_id
WHERE mc.medicioncalidad_id={ULTIMA_MEDICION}"

**Interfaces - Promedio de atributos por interfaz**

"SELECT avg(cantidad_atributos) FROM interfaz i INNER JOIN modelo m
ON i.mis_interfaces_id2 = m.modelo_id
INNER JOIN medicioncalidad mc
ON mc.mi_modelo_modelo_id_oid = m.modelo_id
WHERE mc.medicioncalidad_id={ULTIMA_MEDICION}"

**Interfaces - Porcentaje de atributos sobre métodos**

"SELECT (sum(cantidad_atributos)/sum(cantidad_metodos))*100 FROM interfaz i INNER JOIN modelo m
ON i.mis_interfaces_id2 = m.modelo_id
INNER JOIN medicioncalidad mc
ON mc.mi_modelo_modelo_id_oid = m.modelo_id
WHERE mc.medicioncalidad_id={ULTIMA_MEDICION}"