

# Memoria

## Metodología de Desarrollo PSI

**Autores:**

Esteban Gesto

Karim Hallar

Marina Prats

**Tutor:**

Mg. Albert Osiris Sofía

Proyecto de Software II

Licenciatura en Sistemas

Unidad Académica Río Gallegos

Universidad Nacional de la Patagonia Austral



-¿Cómo se atrasa  
un año un proyecto  
grande de software?

*-Un día a la vez.*

*Fred Brooks  
"The Mythical Man-Month"*



*Dedicamos este trabajo a nuestras familias, quienes nos han apoyado incondicionalmente durante todos estos años. Sin su ayuda y aliento no nos hubiese sido posible llegar hasta aquí. ¡Gracias!*

*También queremos agradecer a todos los docentes que han sabido transmitirnos su entusiasmo y conocimiento, contribuyendo a nuestra formación a lo largo de la carrera, y en especial a nuestro Tutor Mg. Osiris Sofía, por el apoyo y la confianza depositada en nosotros.*

*Esteban, Karim y Marina*



## Tabla de contenido

<b>Proyecto de Software II .....</b>	<b>6</b>
<i>Introducción.....</i>	<i>6</i>
<i>Resumen .....</i>	<i>6</i>
<i>Fundamentación e importancia.....</i>	<i>6</i>
<i>Objetivos.....</i>	<i>7</i>
Objetivo General.....	7
Objetivos Específicos.....	7
<i>Metodología de trabajo.....</i>	<i>8</i>
<i>Estado del arte.....</i>	<i>9</i>
Proceso Unificado de Desarrollo.....	10
Características .....	10
PSP y TSP .....	11
Metodologías ágiles .....	11
Librerías de plantillas reutilizables para ingeniería de software.....	11
ReadySet .....	11
MeRinde.....	11
MoProSoft .....	12
SPEM (Software & Systems Process Engineering Meta-Model) .....	12
Tutelkan .....	12
<b>Metodología de Desarrollo PSI .....</b>	<b>13</b>
<i>Descripción .....</i>	<i>13</i>
<i>Características .....</i>	<i>13</i>
<i>Influencias.....</i>	<i>14</i>
<i>Elementos de Contenido.....</i>	<i>16</i>
Roles.....	16
Tareas.....	16
Productos de Trabajo.....	16
Guías .....	17
Categorías.....	18
<i>Procesos.....</i>	<i>19</i>
Actividades .....	19
Fases .....	19
Iteraciones.....	19
Hitos.....	20
<i>Herramientas Sugeridas .....</i>	<i>20</i>
<b>Herramienta de Acceso a la metodología .....</b>	<b>21</b>
<i>Descripción .....</i>	<i>21</i>
<i>Proceso de desarrollo de la herramienta .....</i>	<i>21</i>
<i>Contribución al Proyecto Eclipse .....</i>	<i>23</i>

<b>Epílogo .....</b>	<b>24</b>
<i>Experiencias de uso .....</i>	<i>24</i>
<i>Vigencia del trabajo y perspectiva a futuro .....</i>	<i>24</i>
<i>Conclusiones .....</i>	<i>26</i>
<b>Referencias.....</b>	<b>27</b>
<b>Anexo I. Disciplinas.....</b>	<b>30</b>
<b>Anexo II. Roles.....</b>	<b>41</b>
<b>Anexo III. Productos de Trabajo .....</b>	<b>43</b>
<b>Anexo IV. Catálogo de Software .....</b>	<b>47</b>

# Proyecto de Software II

## Introducción

La obtención de un software con calidad implica la utilización de metodologías o procedimientos estándares para el análisis, diseño, programación y prueba del software que permitan uniformar la filosofía de trabajo, con el fin de lograr una mayor confiabilidad, mantenimiento y facilidad de prueba, a la vez que eleven la productividad, tanto para la labor de desarrollo como para el control de la calidad del software.

En este trabajo se presenta una iniciativa integradora de diferentes propuestas de mejora de procesos de software para pequeños y medianos proyectos, adaptándola en base a la experiencia adquirida como alumnos y durante nuestros primeros años de actividad profesional haciendo hincapié tanto en las necesidades de los alumnos de las carreras Analista de Sistemas y Licenciatura en Sistemas de nuestra Universidad como también en los requerimientos de un desarrollo en el ámbito profesional donde se apliquen practicas de ingeniería de software.

## Resumen

Esta memoria de tesis se encuentra organizada de la siguiente manera:

En la primera sección del documento se encuentra una reseña del proyecto del software II, sus objetivos e importancia junto con un breve análisis del estado del arte actual.

La segunda sección describe las características y componentes de la metodología propuesta.

Dentro de la tercera sección se puede encontrar una explicación del desarrollo de la herramienta de software para el acceso a la metodología.

Por último se encuentra el epilogo de esta memoria, donde se presentan las experiencias de uso, perspectivas de trabajos futuros y conclusiones de esta tesis.

## Fundamentación e importancia

A la hora de comenzar una tarea de desarrollo de software, nos encontramos con que en la actualidad existen diversas metodologías disponibles, muchas de las cuales requieren un largo período de práctica para su correcta utilización, además de estar, por lo general, ideadas para proyectos de gran envergadura.

Esta situación se agrava en el caso de los trabajos de alumnos, donde todo el grupo carece de experiencia profesional en el campo del desarrollo de software, y además no se cuenta con la disponibilidad de tiempo que requeriría una preparación práctica en el uso de estas técnicas.

“Si buscas resultados distintos, no hagas siempre lo mismo.”

*Albert Einstein*

Esta tesis de grado nos permitió generar un marco de trabajo que no sea costoso de aplicar (fácil de entender y de llevar a la práctica), cuyo uso se adecue en concreto al desarrollo de proyectos de los alumnos de la Universidad, en correspondencia con los contenidos de las carreras de Sistemas, lo que permite una adaptación más rápida por parte del alumnado, ahorrando valioso tiempo que podrá ser empleado en las tareas propias del software a realizar.

Al iniciar nuestro Proyecto de Software I, debimos dedicar un tiempo considerable a la preparación de insumos y definición de tareas que luego llevaríamos a cabo a lo largo del proyecto. Este también es el caso de cualquier grupo de desarrollo que se encuentre en una situación similar a la comentada. Debido a estas circunstancias, es que consideramos de importancia el poder ofrecer a los alumnos que estén dando sus primeros pasos dentro de un grupo de desarrollo una serie de herramientas y guías que los orienten durante toda la realización de un proyecto.

El desarrollo de este trabajo involucra a todas las áreas de sistemas. En relación al área de programación se han incluido técnicas para la documentación del código fuente y métodos para facilitar el trabajo en paralelo del grupo de programadores por medio del uso de un repositorio común. El área de ingeniería de software se ve reflejada en que este proyecto muestra y provee herramientas para poder documentar un proceso de desarrollo, fomentado el uso de estándares de calidad de software y proveyendo herramientas para la administración del proyecto.

Estas características hacen que esta tesis sea de alguna manera transversal a las distintas áreas de las carreras de Sistemas, siendo de utilidad para la mayoría de ellas, dando soporte y brindando herramientas a las distintas actividades que en cada una de estas se desarrollan.

## Objetivos

### Objetivo General

El objetivo general de esta tesis de grado consiste en establecer un marco de trabajo para el desarrollo de productos de software dentro de la UNPA-UARG, utilizando el proceso de desarrollo Orientado a Objetos y basado en un ciclo de vida iterativo e incremental.

Haciendo énfasis en las buenas prácticas propuestas por la Ingeniería de Software, se pretende generar un modelo de referencia que permita la mejora de la calidad, tanto de los procesos como del producto software final, facilitando la especialización de sus componentes para una gran variedad de sistemas software, considerando diferentes áreas de aplicación, niveles de aptitud y tamaños de proyectos, lo que facilitará su adaptación para el uso dentro de los trabajos prácticos de las distintas asignaturas de las carreras de Sistemas de esta Universidad.

### Objetivos Específicos

- Confección de un kit de plantillas, estándares y planes, los cuales servirán de soporte para el desarrollo de software en cada una de sus etapas.
- Inclusión de comentarios, guías, *checklists*, y ejemplos que faciliten la interpretación del material, permitiendo la adopción del mismo por parte de usuarios con poca experiencia en las actividades de ingeniería de software.

- Generación de una metodología para el desarrollo de software, que pueda adaptarse para su utilización tanto en proyectos de campo de las distintas asignaturas de la carrera, como así también en sistemas reales que puedan ser generados en el ámbito universitario, como por ejemplo los realizados a través del ProDAT.
- Implementación de una Herramienta de Acceso a la Metodología, que guíe a los usuarios a través de todo el ciclo de vida de desarrollo del software, facilitándole tanto la identificación de las actividades necesarias en cada etapa del mismo, como así también las tareas que cada integrante debe llevar a cabo durante el proceso.
- Promover la utilización de Software Libre dentro de ámbito universitario, mediante la sugerencia de distintas herramientas gratuitas y de código abierto que permiten la asistencia al desarrollador.

## Metodología de trabajo

El grupo de trabajo se compone de tres integrantes, que han compartido varios proyectos, entre ellos el Proyecto de Software I.

Por razones de tiempo, al estar abocados a tareas laborales, no resultaba factible trabajar en conjunto frecuentemente. Dado que anteriormente se utilizó con buenos resultados un repositorio de versiones para la administración de la configuración, en este proyecto se optó por utilizar la misma forma de trabajo. De esta manera, cada integrante trabaja en forma asincrónica guardando sus avances en un repositorio alojado en el Servidor de la Carrera de Sistemas de la Universidad (1).

A la hora de comenzar con este desarrollo, la primera actividad que se realizó fue un relevamiento de algunas metodologías existentes. Para realizar esta tarea se dedicó bastante tiempo, ya que era de importancia interiorizarse en nuevas metodologías de desarrollo actuales, como fue el caso de XP (2), Scrum (3), OpenUP (4), entre otras. También resultó necesario profundizar los conocimientos adquiridos durante la carrera, como por ejemplo en lo referente al Proceso Unificado (5), Desarrollo de Software Orientado a Objetos (6), actividades de Ingeniería de Software (7) (8) (9), y actividades de Gestión de Proyectos.

Además de la recopilación y análisis de textos afines, se tuvo la posibilidad de asistir a charlas y cursos que fueron de importante utilidad para clarificar y comprender a fondo los objetivos de esta Tesis. Entre estos se destacaron el Curso de Gestión de Calidad y Estándares brindado por la Dra. Alejandra Cechich, la consultoría del Dr. Alejandro Fernández, integrante del Laboratorio de Investigación y Formación en Informática Avanzada (LIFIA) (10) y el curso de Tecnologías XML, que fue de utilidad para la etapa de elaboración de la Interfaz de Visualización.

Llegado a este punto, se pudo tener una visión más completa del alcance de esta Tesis. De esta manera se orientó el trabajo a la definición del conjunto de entregables que esta metodología comprendería.

Para esto, se realizó en primer lugar una tarea de recopilación, análisis y selección de distintos set de plantillas, estándares, *checklists*, planes y guías para el desarrollo de proyectos de software, los cuales sirvieron de insumo para la generación del nuevo set de entregables.



En la sección siguiente se podrá apreciar un resumen de las metodologías con más actividad en la actualidad. Esta información se puede ampliar consultando la sección Referencias.

Durante esta etapa, se hizo un fuerte hincapié en la documentación requerida para la presentación de los Proyectos de Software de fin de carrera, dado que ésta abarcaba todas las disciplinas contempladas dentro de los planes de estudio de las carreras de sistemas, sirviendo además de insumo la experiencia adquirida durante el Proyecto de Software I “Horizon” (11). De igual manera se consideraron los contenidos de las distintas asignaturas de los nuevos planes de estudio. Para esto se realizaron una serie de entrevistas a docentes con el fin de interiorizarnos con el enfoque particular de cada asignatura, y obtener un panorama de la posible aplicación de los artefactos generados.

Terminado este periodo exploratorio, se dispuso a la preparación de los distintos entregables, siguiendo las etapas del desarrollo de un proyecto. Para esto se tuvo en cuenta quienes serían los futuros usuarios del material. Por un lado se planteó la generación de plantillas simples y auto contenidas, aptas para su utilización dentro del ámbito de aprendizaje, tanto en trabajos prácticos de asignatura, en los proyectos de los distintos laboratorios y también dentro del marco del ProDAT. A su vez, se buscaría realizar documentación versátil y suficientemente completa de manera de poder ser aplicada a una gran variedad de tipos de proyectos, tanto en tamaño y complejidad, como en campos de aplicación. Esto permitiría su aplicación en desarrollos reales de mediana complejidad, como por ejemplo los realizados en el marco del Programa de Desarrollo y Asistencia Técnica para Terceros (ProDAT), ayudando a acortar de esta manera la brecha existente entre los conceptos teóricos desarrollados dentro del ámbito universitario y su aplicación práctica como profesionales de Sistemas.

Durante los primeros meses de este desarrollo se tuvo la posibilidad de validar algunas plantillas en sus versiones preliminares dentro de asignaturas como Procesos de Desarrollo y Requerimientos de Software, entre otras. De estas primeras interacciones con los alumnos se obtuvieron interesantes observaciones, que serán vistas con mayor detalle en el apartado “Experiencias de Uso”.

Uno de los principales objetivos que se plantearon al comienzo de esta Tesis, fue la generación de una herramienta de Visualización que sirviera de soporte al proceso de desarrollo. Por esto, una vez que se contó con la mayor parte del conjunto de entregables terminados, se comenzó con la elaboración de un producto de software que brindara un acceso intuitivo a toda la metodología. Para el desarrollo del mismo se utilizó como insumo el material generado hasta ese momento, acotándolo de acuerdo a las características de un proyecto pequeño como lo es esta herramienta. Una descripción más detallada de las tareas realizadas en esta etapa se podrá encontrar en la sección “Herramienta de Acceso a la Metodología”.

## Estado del arte

En la actualidad existe un gran número de metodologías de desarrollo de software, cada una enfocada a diferentes tipos de productos de software. A continuación se presenta un panorama actual de distintas metodologías, estándares, prácticas y librerías que resultaron de influencia para el desarrollo de esta tesis.

## Proceso Unificado de Desarrollo

El Proceso Unificado de Desarrollo Software (5) o simplemente Proceso Unificado es un marco de desarrollo de software que se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y por ser iterativo e incremental. El refinamiento más conocido y documentado del Proceso Unificado es el Proceso Unificado de Rational (12) o simplemente RUP.

El Proceso Unificado no es simplemente un proceso, sino un marco de trabajo extensible que puede ser adaptado a organizaciones o proyectos específicos. De la misma forma, el Proceso Unificado de Rational, también es un marco de trabajo extensible, por lo que muchas veces resulta imposible decir si un refinamiento particular del proceso ha sido derivado del Proceso Unificado o del RUP. Por dicho motivo, los dos nombres suelen utilizarse para referirse a un mismo concepto.

### Características

#### *Iterativo e Incremental*

El Proceso Unificado es un marco de desarrollo iterativo e incremental compuesto de cuatro fases denominadas Inicio, Elaboración, Construcción y Transición. Cada una de estas fases es a su vez dividida en una serie de iteraciones (la de inicio sólo consta de varias iteraciones en proyectos grandes). Estas iteraciones ofrecen como resultado un incremento del producto desarrollado que añade o mejora las funcionalidades del sistema en desarrollo.

Cada una de estas iteraciones se divide a su vez en una serie de disciplinas que recuerdan a las definidas en el ciclo de vida clásico o en cascada: Análisis de requisitos, Diseño, Implementación y Prueba. Aunque todas las iteraciones suelen incluir trabajo en casi todas las disciplinas, el grado de esfuerzo dentro de cada una de ellas varía a lo largo del proyecto.

#### *Dirigido por los casos de uso*

En el Proceso Unificado los casos de uso se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones. La idea es que cada iteración tome un conjunto de casos de uso o escenarios y desarrolle todo el camino a través de las distintas disciplinas: diseño, implementación, prueba, etc.

#### *Centrado en la arquitectura*

El Proceso Unificado asume que no existe un modelo único que cubra todos los aspectos del sistema. Por dicho motivo existen múltiples modelos y vistas que definen la arquitectura de software de un sistema.

#### *Enfocado en los riesgos*

El Proceso Unificado requiere que el equipo del proyecto se centre en identificar los riesgos críticos en una etapa temprana del ciclo de vida. Los resultados de cada iteración, en especial los de la fase de Elaboración, deben ser seleccionados en un orden que asegure que los riesgos principales son considerados primero.

## PSP y TSP

PSP (13) y TSP (14) son encarnaciones del *Capability Maturity Model* (CMM) (15) adaptadas al contexto de un desarrollador individual y de un equipo pequeño de desarrollo respectivamente. Una de las ideas fundamentales de CMM es de buscar la mejora de un proceso de desarrollo bien establecido a través de la recolección de métricas. Hoy en día, CMM se ha vuelto prácticamente un estándar para medir el nivel de madurez de una organización de desarrollo.

## Metodologías ágiles

Las metodologías ágiles de desarrollo hicieron su aparición de manera relativamente reciente y fueron propuestas por un grupo de personas que buscaban limitar la burocracia de los proyectos. Las metodologías ágiles comparten principios como:

- Favorecer la entrega de software funcional sobre la entrega de documentos.
- Dar más importancia a los desarrolladores (no verlos como piezas reemplazables)
- Adaptarse continuamente al cambio, usando iteraciones muy cortas.

Una de las metodologías ágiles de desarrollo más populares hoy en día es la programación extrema (XP) (2). Dos de las características de XP son la realización de pruebas unitarias antes de la escritura del código y la programación por parejas.

## Librerías de plantillas reutilizables para ingeniería de software

Existen varios desarrollos enfocados a la generación de sets de plantillas reutilizables para la documentación de procesos de desarrollo de software. A continuación se resumen algunos de mayor actividad en la actualidad:

### ReadySet

ReadySet (16) es un proyecto *open-source* que proporciona paquetes de plantillas de ingeniería de software. Las mismas están diseñadas para ser siempre usadas en un ambiente web.

Estas plantillas no son universales y no intentan proveer guías prescriptivas en el proceso general de desarrollo.

ReadySet está orientado a ingenieros de software que desean que sus proyectos puedan fluir de una forma más sencilla y profesional. ReadySet puede ser usado por cualquier persona que sea capaz de utilizar un editor HTML o editar HTML en un editor de texto

### MeRinde

MeRinde (17) es un proyecto de Software Libre que propone un estándar para el proceso de desarrollo de software que puede ser empleado y adaptado según los requerimientos de cualquier comunidad u organización para el desarrollo de sistemas y además para producir y mantener una librería de plantillas reutilizables para la ingeniería de software. Estas plantillas proveen un punto partida para los documentos utilizados en proyectos de desarrollo

de software, con lo que pueden ayudar a los desarrolladores a trabajar más rápido y evitar pasar por alto aspectos importantes del proceso de desarrollo.

### MoProSoft

MoProSoft (18) es un modelo de procesos para la industria de software mexicano, que fomenta la estandarización de su operación a través de la incorporación de las mejores prácticas en gestión e ingeniería de software. La adopción del modelo permite elevar la capacidad de las organizaciones que desarrollan o mantienen software para ofrecer servicios con calidad y alcanzar niveles internacionales de competitividad.

El modelo pretende apoyar a las organizaciones en la estandarización de sus prácticas, en la evaluación de su efectividad y en la integración de la mejora continua

### SPEM (Software & Systems Process Engineering Meta-Model)

SPEM (19) ofrece un marco de trabajo para el modelado, documentación, presentación, gestión e intercambio de los procesos de desarrollo Software y sus componentes, dando una sintaxis y una estructura común para cada aspecto del proceso de desarrollo, incluyendo: Roles, Tareas, Productos de Trabajo y guías.

Con este fin, permite disponer de un formato estandarizado por el cual crear una base de conocimiento para la ingeniería de procesos de desarrollo de software y sistemas que pueda ser desplegada y gestionada por los propios desarrolladores.

### Tutelkan

Tutelkan (20) es una metodología de mejoramiento continuo de la calidad en los procesos de las empresas, que tiene como foco principal iniciar en el corto plazo un proceso de certificación de calidad.

Tutelkan en el idioma mapuche quiere decir: “contentar a alguien con un trabajo, hacer bien alguna cosa”.

# Metodología de Desarrollo PSI

## Descripción

La metodología PSI presenta un marco de trabajo genérico para el desarrollo de sistemas de software. En el mismo se brindan los elementos necesarios para llevar adelante un proyecto siguiendo los lineamientos de la Ingeniería de Software, y apoyado en la metodología de desarrollo Orientada a Objetos.

En las siguientes secciones se presentarán las características principales de la metodología y una descripción de los componentes fundamentales de la misma.

“Si tu proyecto no funciona, busca en la parte que no pensaste que fuera importante”

Arthur Bloch

## Características

La metodología de desarrollo PSI está pensada para ser flexible y adaptarse al contexto particular del laboratorio de Desarrollo de Software de la UNPA UARG, y a su vez permitir la utilización de sus componentes en los trabajos prácticos de las distintas asignaturas de las carreras de sistemas. Por otra parte, se ha pensado en que este material no solo cumpla con las necesidades académicas, sino que pueda ser utilizado en desarrollos reales en el ámbito profesional, como por ejemplo en el marco del programa ProDAT.

**Trabajo en Grupo.** Las prácticas propuestas están enfocadas al trabajo en conjunto, como grupo de desarrollo, donde existe una clara división de roles con distintas responsabilidades asociadas a cada uno. De esta manera se fomentan actividades como planificaciones, estimaciones, reuniones y revisiones del trabajo.

**Aplicación Práctica en Asignaturas.** La metodología PSI define un conjunto de buenas prácticas, como así también un conjunto de plantillas que permiten que un trabajo de campo pueda ser construido gradualmente por los alumnos a medida que cursan el *Laboratorio de Desarrollo de Software*.

**Control de versiones.** El código y toda la documentación inherente al proyecto se almacenan en un repositorio que permite llevar el control de versiones. El repositorio está organizado de tal manera que, además de contener el código de la versión en desarrollo, contiene una serie de “fotografías” de las distintas versiones tanto de los ejecutables como de toda la documentación. Esto permite poder mostrar en todo momento la última versión 'estable' del desarrollo, como así también se facilitan tareas de auditoría al permitir el acceso al proyecto en un punto anterior. Por otra parte, el aplicar normas de gestión de configuraciones, evita problemas de concurrencia entre los usuarios del repositorio, al resguardar las distintas versiones y proteger a los elementos de modificaciones indeseadas.

**Desarrollo en paralelo.** Gracias a la planificación y división de tareas, se facilita la ejecución de tareas de mediana complejidad de manera individual o en pequeños grupos, adecuándose a la disponibilidad de tiempo de los distintos integrantes. Esto permite que las actividades del desarrollo de un proyecto puedan efectuarse en forma paralela.

**Documentación continua.** Aunque el proceso pone énfasis en la producción de software funcional, también le da importancia a la generación de documentación de forma continua y colaborativa. Para esta tarea, se contempla la generación de un set de plantillas propio, el cual está pensado para ser utilizado en todas las etapas del desarrollo, brindando material de soporte para cada una de las actividades.

**Gestión de Calidad.** Esta metodología fomenta el uso y buenas prácticas de tareas enfocadas a la calidad del proceso. Las mismas están basadas en normas ampliamente aceptadas por la comunidad, y avaladas por organismos como la IEEE (21) y la OMG (22).

**Auto contenida y enfocada a alumnos.** Se ha hecho hincapié en que los insumos que provee esta metodología contengan explicaciones y ejemplos de uso, de manera de permitir la utilización directa de las distintas plantillas sin la necesidad de recurrir a otras fuentes de información. El empleo de las mismas está, por este motivo, enfocado a la utilización como material de cátedra por los alumnos de las carreras de sistemas. Sin embargo también resulta de utilidad para profesiones con poca experiencia o faltos de hábitos en las tareas de ingeniería de software.

**Facilidad de acceso.** Para el acceso a esta metodología se ha desarrollado una herramienta que facilita el acceso y la visualización de todos los componentes, de una manera intuitiva y práctica.

**Adecuación para proyectos particulares.** El proyecto ha sido pensado para adaptarse fácilmente a distintos tipos de desarrollos, desde trabajos de campos de asignaturas a desarrollos profesionales. Cada componente puede tomarse individualmente o en conjunto y ajustarse a las necesidades particulares de cada caso.

**Licencia Libre.** Este proyecto ha sido desarrollado bajo la forma de licencia Copyleft (23). La metodología se encontrará disponible a través de internet para su libre acceso y modificación por cualquier persona.

## Influencias

Si bien hoy en día existe una gran variedad de metodologías para el desarrollo de software, cada una con características particulares que permiten una mejor adaptación a diferentes necesidades, en esta Tesis hemos decidido inclinarnos hacia el paradigma de la Orientación a Objetos (6) (24), el Proceso Unificado de Desarrollo (5) y el lenguaje de Modelado Unificado UML (25) (26) (27), con los cual tenemos mayor experiencia y se encuentran más familiarizados los alumnos de Sistemas de nuestra Universidad.

Como se mencionó en secciones anteriores, fue analizada la posibilidad de trabajar con una metodología de desarrollo “ágil”. Si bien este estilo de trabajo se está imponiendo en muchos ámbitos, y con muy buenos resultados, uno de los requisitos para su exitosa aplicación es el contar con un grupo de desarrolladores con gran experiencia práctica, tanto en las técnicas propuestas como en programación, lo cual contradice los objetivos de este trabajo, de poder ser utilizado como material de cátedra y como insumo para desarrollos de grupos con poca experiencia en tareas de ingeniería.

El Proceso Unificado de Desarrollo, por otra parte, nos permitió volcar la experiencia adquirida durante nuestros años de estudio y adaptar los insumos desarrollados durante nuestro Proyecto de Software I, permitiendo generar un marco de trabajo que no resulte extraño para alumnos con base teórica en esta metodología.

Además, el uso de un ciclo de vida iterativo e incremental resulta importante para que una metodología sea flexible a distintos tipos de proyectos, lo que facilita su puesta en práctica en proyectos de pequeña y mediana envergadura.

PSP y TSP resultaron también de gran influencia para este trabajo, a través de la introducción del concepto de calidad, y que la misma se transfiere al producto aplicando calidad al proceso de desarrollo. Esto llevó a resaltar la importancia de la generación de estándares y la aplicación de métricas como parte de la metodología.

Durante nuestra etapa de investigación, nos encontramos con proyectos como ReadySet y MoProSoft, ambos *open-source* (de código abierto), los cuales nos inspiraron a decidir que nuestra metodología debería estar disponible completamente a la comunidad. En cuanto a la estructuración de su contenido, no lo consideramos suficientemente práctico para los futuros usuarios a los que nuestra metodología apunta. Sin embargo tomamos en cuenta algunos ítems interesantes durante la confección de nuestro set de plantillas.

MeRinde nos pareció una de las mejores iniciativas en castellano de estandarización de procesos de software disponibles en la actualidad. Sin embargo encontramos que esta metodología cuenta con muy pocas guías de trabajo y sus plantillas son demasiado breves, lo cual sería insuficiente a la hora de intentar utilizarla directamente en un proyecto real. Es destacable la forma de acceso a la metodología, aunque la misma aún no está completa, encontrándose en una etapa preliminar. Además, nos pareció muy interesante que los insumos se encuentran disponibles para la descarga en diferentes formatos, lo cual resulta muy útil para diferentes tipos de usuarios.

Otra metodología consultada fue Tutelkan. Esta hace hincapié en las tareas de gestión del proyecto, y cuenta con plantillas bien descriptas para tareas como gestión de riesgos y gestión del equipo de trabajo.

Para la segunda etapa de nuestro desarrollo, consistente en la estructuración de la metodología, optamos por basarnos en SPEM, un meta-modelo estándar para la generación de procesos de ingeniería. El mismo es mantenido por la OMG (22), lo cual le brinda el respaldo de distintos sectores tanto del mercado como de la comunidad científica.

Todos los componentes de SPEM se encuentran estandarizados (28). Asimismo, existen herramientas de software que implementan este modelo, lo que permite generar metodologías basadas en este estándar y adaptadas a contextos particulares. En nuestro caso hemos utilizado una versión especial del *framework* de desarrollo de Eclipse, EPF Composer (29), herramienta con la cual se encuentran habituados los alumnos, al utilizarla frecuentemente en las asignaturas del área de Programación.



## Elementos de Contenido

Los elementos de contenido son los constructores básicos y se derivan del patrón primitivo de trabajo: alguien (rol) hace algo (tarea) para obtener algo (producto de trabajo) basándose o ayudándose en algo (guía). En consecuencia, los cuatro tipos de elementos de contenido son: Tarea, Rol, Producto de Trabajo y Guía. Adicionalmente, existe un quinto tipo de elemento de contenido incluido con fines de clasificación y agrupación, llamado Categorías. En los siguientes apartados se presenta cada uno de ellos.

### Roles



Un Rol define un conjunto de habilidades, competencias y responsabilidades relacionadas, de un individuo o de un grupo. No se deben confundir roles con personas, ya que la vinculación entre personas y roles se realiza durante la planificación del proyecto y puede ocurrir que un individuo desempeñe varios roles o que un rol sea desempeñado por varios individuos. Un rol es un Elemento de Método usado en las Definiciones de Tareas para señalar quienes las realizan.

### Tareas



Una Tarea describe una unidad de trabajo asignable y gestionable, es decir, es la unidad atómica de trabajo para definir procesos. Su granularidad es de unas pocas horas a unos pocos días, afectando a unos pocos productos de trabajo y vinculando a unos pocos roles.

Es un Elemento de Método que define el trabajo realizado por roles, pero también es una Definición de Trabajo (en procesos).

Una definición de tarea se utiliza como un elemento de la definición de un proceso. Las definiciones de trabajo son más utilizadas para la planificación y seguimiento de los progresos de dicho proceso.

Una definición de tarea tiene un propósito claro en el que se desempeñan un conjunto de roles para alcanzar un objetivo bien definido. Proporciona una lista completa paso a paso con las explicaciones necesarias para lograr el objetivo determinado.

### Productos de Trabajo

Los productos de trabajo son Elementos de Contenido que se utilizan, modifican, y son producidos por definiciones dentro de las Tareas. Pueden servir como una base para la definición de los activos reutilizables.

Existen tres tipos predefinidos de productos de trabajo:



**Artefacto:** De naturaleza tangible (modelo, documento, código, archivos, etc.). Un artefacto puede estar formado por otros artefactos más simples.



**Entregable:** Provee una descripción y definición para empaquetar otros productos de trabajo con fines de entrega a un cliente interno o externo. Representa una salida de un proceso que tiene valor para un usuario, cliente u otro participante.





**Resultado:** Un producto de trabajo de naturaleza intangible (resultado o estado), o que no está formalmente definido.

## Guías

Las guías son un elemento descriptible que proporciona información adicional relacionada con otros elementos. Por ejemplo: Ayuda o información sobre cómo trabaja un rol, cómo crear un producto de trabajo, cómo usar una herramienta o cómo realizar una tarea. Los tipos de guías predefinidos son:



**Activo Reutilizable:** Provee una solución a un problema para un contexto dado. Incluye reglas o instrucciones sobre cómo utilizarlo.



**Concepto:** Resumen de ideas clave asociadas con principios básicos subyacentes. Refieren a tópicos más generales que las directrices y abarcan varios productos de trabajo y/o actividades. Ejemplo: “iterativo e incremental”.



**Definición de Término:** Definición de un término, concepto o idea relevante. Sirve para generar una especie de glosario automático. Se relacionan con Elementos de Contenido mediante su aparición en las descripciones textuales.



**Directriz:** Provee detalle adicional sobre cómo realizar una tarea o grupo de tareas, o provee información adicional, reglas y recomendaciones sobre productos de trabajo. Entre otros, puede incluir detalles sobre: las mejores prácticas y aproximaciones diferentes para hacer un trabajo; cómo usar cierto tipo de producto de trabajo; los subtipos y variantes de un artefacto y su evolución a lo largo del tiempo; habilidades que los roles deben adquirir o mejorar; medidas del progreso o madurez, etc.



**Documentación (Whitepaper):** Versión especial de Concepto que ha sido revisada o publicada externamente y que puede ser leída y comprendida de forma aislada.



**Ejemplo:** Ejemplo de una instancia típica, parcialmente completada, de uno o más productos de trabajo o descripción del escenario en que una tarea debe ser realizada.



**Guía de Herramienta:** Explica el uso de una cierta herramienta en el contexto de cierto trabajo, o de forma independiente.



**Guía para la Estimación:** Indicaciones para estimar el esfuerzo asociado con cierto trabajo, incluyendo consideraciones sobre cómo hacer la estimación y las métricas a utilizar.



**Hoja de Ruta:** Hoja de ruta que describe, en forma de camino lineal, cómo suele llevarse a cabo una actividad o un proceso complejo. Provee información sobre cómo las actividades y tareas se relacionan entre sí a lo largo del tiempo. Sólo pueden estar asociados a Actividades y Procesos.



**Informe:** Plantilla predefinida de un resultado que se obtiene de forma automática mediante alguna herramienta.



**Lista de Comprobación (Checklist):** Identifica una serie de ítems que deben ser completados o verificados



**Material de Soporte:** Comodín para utilizar cuando se está en un caso que no encaja en ninguno de los demás tipos de guías.



**Plantilla:** Establece la tabla de contenidos, secciones, cabeceras y formato estandarizado predefinido de un artefacto (documentos, modelos, etc.). Puede incluir descripciones sobre cómo usar y completar cada parte.



**Práctica:** Manera o estrategia predefinida de hacer un trabajo que tiene un impacto positivo sobre la calidad de un producto de trabajo o de un proceso. Son ortogonales a los métodos y procesos, de forma que una práctica resume aspectos que pueden impactar en diferentes partes de un método o proceso. Ejemplos: “gestionar riesgos”, “verificación continua de la calidad”, “desarrollo centrado en la arquitectura”, “desarrollo basado en componentes”.

## Categorías

Una Categoría es un elemento de contenido, o de proceso, usado para categorizar, es decir, clasificar o agrupar dichos elementos en base a los criterios que desee el ingeniero de procesos. Una categoría puede tener a su vez subcategorías. Esto permite establecer cualquier tipo de jerarquía de agrupamiento de elementos.

Las categorías pueden ser utilizadas para categorizar el contenido basado en criterios del usuario, así como para definir las estructuras de árbol entero de categorías anidadas, permitiendo al usuario navegar y explorar sistemáticamente el contenido y el método de los procesos basados en estas categorías.

Se incluyen 5 tipos predefinidos de categorías:



**Conjunto de Roles:** Sirven para agrupar roles que tienen algo en común (usan técnicas similares, requieren habilidades parecidas, etc.). Ejemplo: Analista englobando a Analista de Sistemas e Ingeniero de Requisitos.



**Disciplina:** Permiten categorizar el trabajo (tareas). Una disciplina es una colección de tareas que están relacionadas con un área principal de esfuerzo dentro de un proyecto completo. Suelen estar basadas en una perspectiva tradicional de proyectos en cascada: requisitos, análisis, diseño, construcción, pruebas, mantenimiento, gestión del proyecto, aseguramiento de calidad, etc.



**Dominio:** Permiten establecer una jerarquía de dominios, para clasificar productos de trabajo, con tantos niveles como se desee. El nivel inferior son productos de trabajo y el resto de niveles son dominios y subdominios. Al ser una jerarquía, un producto de trabajo sólo puede estar asociado con un único dominio. Ejemplos: “modelo”, “código”.



**Herramienta:** A pesar del nombre, no sirve para categorizar herramientas sino guías de herramientas. Por tanto, para asociar herramientas con tareas, roles o productos de trabajo deberá emplearse una categoría personalizada.



**Clase de Producto de Trabajo:** Se incluye por compatibilidad con la versión 1 de SPEM.

Se distingue de Dominio en que un producto de trabajo puede pertenecer a varias clases de producto de trabajo distintas. Ejemplo: El mismo artefacto puede incluirse dentro de “Documento de Análisis” y dentro de “Producto Software”.

Además, los elementos del contenido de método se pueden organizar alternativamente mediante categorías personalizadas. Estas pueden contener cualquier tipo de elemento, proceso, categoría, etc. Son útiles para organizar de forma lógica elementos que no se puedan organizar mediante las categorías estándar, como por ejemplo, agrupar todas las plantillas de productos, agrupar todas las guías que sean directrices, etc.

Las categorías personalizadas permiten organizar el contenido de acuerdo al esquema que queramos, sirviendo como medio eficaz para organizar el contenido para su publicación. El contenido de una categoría personalizada se especifica en la pestaña Asignar del editor de la categoría.

## Procesos

Se distinguen dos etapas a la hora de implementar un proceso o metodología: Primero se completa el Contenido de Método, cuyos conceptos han sido presentados anteriormente, y en segundo lugar, se combinan y reutilizan dichos elementos para ensamblar actividades y procesos.

## Actividades

Una actividad representa una unidad de trabajo general en un proceso. Una actividad puede tener estructura interna formada por agregación de elementos de desglose, que pueden ser de varios tipos, no solo de trabajo: actividades más simples (anidamiento), elementos de método en uso (roles en uso, productos de trabajo en uso), e hitos. La complejidad de la estructura de desglose de una actividad puede variar entre 0 tareas o todo un proceso completo.

Una actividad representa una unidad de trabajo general asignable a realizadores específicos, representados por roles. Puede tener entradas y producir salidas, representadas por Productos de Trabajo.

Las actividades pueden ser de diferentes tipos: Fases, Iteraciones e Hitos.

### Fases



Una Fase representa un periodo significativo en un proyecto, que termina con punto de control de gestión importante, un hito o un conjunto de entregables concluidos. Se incluye en el modelo como una actividad especialmente predefinida, debido a su importancia en la definición de cualquier proyecto.

### Iteraciones



Los grupos de iteración son un conjunto de actividades anidadas que se repiten más de una vez. Representa un elemento de estructuración importante para organizar el trabajo en ciclos repetitivos. El concepto de iteración se puede asociar con normas diferentes en los diferentes métodos.

## Hitos



Un hito es un elemento de división del trabajo que representa un acontecimiento significativo para un proyecto de desarrollo.

Un hito describe un evento significativo en un proyecto de desarrollo, como una decisión importante, la culminación de una prestación, la conclusión de un entregable o la finalización de una fase de proyecto.

Cada fase se concluye con un hito bien definido, un punto en el tiempo en el cual se deben tomar ciertas decisiones críticas y alcanzar las metas clave antes de pasar a la siguiente fase, ese hito principal de cada fase se compone de hitos menores que podrían ser los criterios aplicables a cada iteración. Los hitos para cada una de las fases son:

- *Inicio - Objetivos del ciclo de vida*
- *Elaboración - Arquitectura del ciclo de vida*
- *Construcción - Capacidad operacional*
- *Transición – Lanzamiento del producto*

## Herramientas Sugeridas

Las herramientas CASE (*Computer Aided Software Engineering*, Ingeniería de Software Asistida por Computadora), son un conjunto de diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software, reduciendo el costo de las mismas en términos de tiempo y dinero. Estas herramientas facilitan la automatización del ciclo de vida del desarrollo de sistemas, completamente o en alguna de sus fases.

Seleccionar una herramienta CASE no es una tarea simple. No se puede afirmar que una herramienta es “*mejor*” respecto a otra. Existen numerosos ejemplos respecto al uso de herramientas CASE y las fallas que pueden producirse.

La elección de la herramienta correcta depende de diversos factores. Se deben considerar las características y experiencia del grupo de desarrollo, el tipo de organización en el que está inmerso, y analizar cuidadosamente las alternativas disponibles.

Para los alumnos que recién comienzan a desarrollar sistemas, esta elección puede resultar complicada, debido a la amplia variedad de herramientas disponibles. En este apartado se pretende brindar un acercamiento a diversas herramientas CASE que han resultado útiles al grupo durante el cursado de la carrera y la realización de los Proyectos de Software I y II.

Además, las herramientas presentadas son parte del llamado Software Libre, por lo que el acceso y utilización de las mismas no requiere el pago de ningún tipo de licencia para su uso, pudiendo ser las mismas descargadas de internet gratuitamente.

En el ANEXO IV puede observarse un listado de herramientas sugeridas, junto con sus características principales y enlaces a sus sitios web.

# Herramienta de Acceso a la metodología

## Descripción

La metodología descrita en las secciones anteriores contempla la instanciación del proceso conforme a todos sus artefactos, actividades y tareas, no teniendo la posibilidad de realizar personalizaciones de acuerdo a la magnitud de los diferentes proyectos. A su vez, la metodología por sí misma no presenta facilidades para su acceso y visualización.

En vista de esto, y de la necesidad de organizar el conjunto de plantillas, estándares, planes, etcétera, de forma de permitir un acceso práctico y ágil, se ha desarrollado una herramienta de visualización, a través de la utilización de la herramienta *Eclipse Process Framework Composer* (29).

Como resultado, se ha obtenido un sitio web dinámico, altamente configurable para distintos tipos de proyectos, el cual sirve como guía durante el proceso de desarrollo, facilitando la identificación de las actividades a realizar en cada etapa del proyecto, como así también la obtención de los insumos necesarios para las mismas, de una manera simple e intuitiva.

“Unas buenas especificaciones incrementarán la productividad del programador mucho más de lo que puede hacerlo cualquier herramienta o técnica”

Milt Bryce

## Proceso de desarrollo de la herramienta

La etapa de desarrollo de esta herramienta comenzó luego de que se tuviera el conjunto de plantillas completamente terminado, lo que permitió tener una real dimensión del alcance de la metodología.

Para la gestión de este proyecto se adoptaron las prácticas que se describieron para esta tesis, entre las cuales cabe destacar planificaciones, gestión de riesgos, administración de la configuración y reuniones formales periódicas.

En primer lugar se realizó un análisis de requisitos, consultando con distintos docentes sobre la mejor manera de acceder al material. Luego de haber mantenido algunas entrevistas se decidió utilizar una plataforma web, ya que la misma proporciona la posibilidad de un acceso rápido y sencillo a la información. De la misma manera, se consideró apropiado el alojamiento de la misma en el servidor de la Carrera (1), asegurando un buen acceso dentro del ámbito universitario.

Durante esta etapa se identificó como un requisito de importancia la necesidad de acceder a los artefactos desde distintos enfoques, ya sea determinando qué tareas se deben llevar a cabo en cada etapa, como así también permitir identificar qué tareas son responsabilidad de un rol determinado. El sistema debería considerar éstas y otras posibles formas de visualización.

Antes de continuar con el desarrollo, se realizó un breve estudio de factibilidad que determinó la posibilidad de realizar esta implementación con un bajo riesgo, y que a su vez era de gran importancia para facilitar el uso de la metodología el poder contar con una herramienta de éste tipo.

Se consideraron las distintas alternativas para la implementación de esta herramienta, entre las cuales se tuvo en cuenta la posibilidad de desarrollar un sitio dinámico desde cero, en lenguajes como PHP o Java. También se analizó la utilización de un *framework* de documentación para procesos, entre los cuales se estudiaron *IBM Process Composer* (30) y *Eclipse Process Framework Composer* (29).

Luego de un análisis de las distintas posibilidades, se concluyó en la utilización de Eclipse, dadas sus características de versatilidad, facilidad de modificación, acceso al código fuente y buena adaptación a los requerimientos planteados.

Una de las tareas principales de la etapa de análisis, incluyó el aprender a utilizar el *framework* de desarrollo de Eclipse, como así también comprender la ingeniería de procesos detrás del Meta-Modelo SPEM (19), el cual es la base para la construcción de modelos de proceso con Eclipse EPF.

La etapa de diseño se dedicó a la definición de los distintos componentes, como así también las relaciones entre los mismos. En los anexos se pueden apreciar los distintos componentes de esta herramienta.

Así mismo, dentro de esta etapa se diseñaron las distintas vistas del sitio, considerando los requisitos relevados en las primeras etapas del proyecto. Para lograr esto, se plantearon una serie de prototipos de interfaces, que permitieron depurar la visualización final.

Durante la etapa de implementación no se produjeron mayores inconvenientes. El grupo se dedicó a integrar los distintos insumos previamente elaborados dentro del *framework* de desarrollo. Esto fue posible gracias a que se contó con el tiempo para el estudio de la herramienta y el diseño del sitio.

A continuación se realizaron una serie de pruebas donde se pudo constatar el correcto funcionamiento de la herramienta, y su concordancia con los objetivos planteados para este desarrollo.

Por último, se realizó la instalación de la herramienta de visualización en el Servidor de la Carrera (31), dejándola disponible para el acceso por parte del alumnado, como así también para la descarga de sus fuentes, permitiendo su libre adaptación para proyectos particulares.

En los anexos que acompañan esta memoria, se pueden consultar los documentos generados durante esta etapa.

## Contribución al Proyecto Eclipse

Eclipse Process Framework (EPF) de la fundación Eclipse tiene por objeto producir un marco de trabajo personalizable para un proceso de software, con un contenido guiado de un proceso genérico con sus herramientas. Este proceso fue pensando para una amplia variedad de tipos de proyectos y de estilos de desarrollo.

El framework del proceso tiene dos objetivos:

Proporcionar un marco extensible y herramientas ejemplares para el proceso de ingeniería de software, métodos y procesos, gestión de la biblioteca de software, configuración y publicación de un proceso.

Ofrecer el contenido de un proceso a modo de ejemplo de manera que este sea extensible para un desarrollo de software sin importar el estilo de desarrollo ya sea el mismo iterativo, ágil o cualquier tipo de desarrollo de software.

Al comenzar a utilizar esta herramienta, se observó que no existía una traducción al castellano disponible por lo que, al generar la publicación del sitio, muchos componentes aparecían en inglés.

Esto generaba cierta desprolijidad en la presentación final, por lo que se decidió dedicar un tiempo a la traducción de los archivos XML incluidos en las librerías de EPF Composer, de manera de mejorar la visualización final de nuestro producto.

Para realizar esta tarea fue necesario acceder al código fuente del Framework, lo cual fue posible gracias a que todo el proyecto EPF se encuentra disponible bajo la licencia Pública EPL (32).

Al concluir el proceso de traducción se contactó al grupo de desarrollo y mantenimiento de EPF para ofrecer como aporte al proyecto el módulo de publicación traducido al castellano.



# Epílogo

## Experiencias de uso

Durante el periodo de desarrollo de nuestra tesis, se tuvo la posibilidad de aplicar los conocimientos adquiridos, como así también utilizar algunos de los productos de este trabajo en distintas oportunidades.

Algunas de las plantillas fueron utilizadas como insumo para los trabajos prácticos de asignaturas como Procesos de Desarrollo de Software (1er. Año), Requerimientos de Software (2do. Año) y Análisis y Diseño de Software (2do. Año).

Gracias a esto se observó que la utilización de plantillas resulta práctica para alumnos con poca experiencia y los guía en las tareas que estén llevando a cabo. El uso de plantillas produjo, además, una normalización y mejora en la presentación de los trabajos prácticos de dichas asignaturas.

Además, luego de esta experiencia se modificaron cuestiones de diseño y contenido de nuestro material, gracias a las devoluciones realizadas por alumnos y docentes de las materias.

Por otro lado, se ha tenido la oportunidad de ofrecer la metodología completa para su uso a alumnos avanzados de la carrera Analista de Sistemas, los que se encuentran realizando su Proyecto de Software I para el desarrollo de un Sistema de Control de Acceso mediante un dispositivo biométrico. Dichos alumnos se mostraron entusiastas con la posibilidad de contar con la ayuda de un marco de trabajo para la realización de su proyecto, observando que dispondrían de mayor tiempo para la realización de tareas técnicas específicas de su trabajo.

Por último, se brindó material a los docentes de Análisis y Producción del Discurso para la realización de actividades de interpretación de textos relacionados, con los alumnos de 1er año, ayudándolos además a familiarizarse con la documentación de proyectos de software.

“Dar ejemplo no es la principal manera de influir sobre los demás; es la única manera.”

*Albert Einstein*

## Vigencia del trabajo y perspectiva a futuro

La metodología PSI fue pensada teniendo en consideración los planes actuales de las carreras de sistemas, los cuales están enfocados hacia el desarrollo de software orientado a objetos, sobre la base del proceso unificado y la notación UML, siendo estas perspectivas, además, unas de las perspectivas de mayor adopción en el presente por parte de la comunidad profesional.

Es posible, sin embargo, tomar los insumos presentados en este trabajo y adaptarlos fácilmente a nuevos paradigmas o a enfoques diferentes, ya que la documentación del mismo se encontrará disponible libremente en la Universidad y los distintos componentes pueden



expandirse, simplificarse o modificarse individualmente, para aplicarlos a otros requerimientos.

Como se ha mencionado en apartados anteriores, se tuvo la posibilidad de utilizar parte de los contenidos de esta Tesis durante la realización de trabajos prácticos en asignaturas como Procesos de Desarrollo de Software, Requerimientos de Software y Análisis y Diseño de Software. Para el año lectivo 2010 se prevé tener disponible la totalidad del material a disposición de dichas materias, e incorporar además a las asignaturas de tercer año Gestión de Proyectos y Laboratorio de Desarrollo de Software. De igual manera, será posible su utilización en asignaturas de otras áreas.

El sitio web del proyecto se encuentra alojado en el servidor de las carreras de Sistemas, lo que permitirá el acceso a todos los docentes y alumnos de la carrera, esperando contar además con actualizaciones y mejoras, apoyados en la retroalimentación que se genere con la puesta en práctica.

Por otra parte, será posible su utilización en el marco del ProDAT, donde se aplicará la metodología en el desarrollo de un Sistema de Gestión de la Estructura Orgánica de la Provincia de Santa Cruz que comenzará a desarrollarse durante este año. La vinculación de esta Tesis con el Programa de Asistencia a Terceros podrá permitir además una mejor adaptación de los alumnos avanzados que se incorporen como pasantes en los distintos proyectos, ya que los mismos se encontrarán con un marco de trabajo con el cual se han ido familiarizando a lo largo de toda la carrera.

También, durante 2009 se recibió el visto bueno del Sub Secretario Provincial de informática, Ing. Fernando Igoillo, para plantear una posible transferencia de la metodología a la Dirección Provincial de Informática, donde resultaría de importancia incorporar prácticas de Ingeniería de Software a los desarrollos que allí se realizan.

Por último, se está comenzando con la preparación de *papers* relacionados a la metodología aquí presentada para ser enviados a Congresos como CIESC (33) y JAIIO (34) durante 2010.

## Conclusiones

Esta tesis de grado se basó en la realización de una metodología de desarrollo de software para pequeños y medianos proyectos, pensada para dar soporte tanto a las asignaturas de las carreras de sistemas como a grupos de desarrollo profesionales de la universidad, con el ánimo de acortar la brecha que suele existir entre los conceptos teóricos adquiridos durante el cursado de una carrera universitaria y la realidad del mercado laboral

El desarrollo de esta metodología se basó en la recopilación y análisis de algunas de las metodologías existentes para luego generar un conjunto propio de plantillas, estándares y planes de trabajo de acuerdo a las buenas prácticas de la ingeniería de software y a la experiencia adquirida durante la carrera.

Se trabajó además en la generación de una aplicación para el acceso y visualización de todo el material que engloba la metodología en cuestión, permitiendo que toda la comunidad de alumnos pueda acceder a su contenido.

Gracias a que se tuvo la posibilidad de compartir nuestro trabajo con alumnos de la carrera, se observaron algunos buenos resultados al aplicar una estandarización de procesos en las actividades prácticas de diversas asignaturas permitiendo a los alumnos tomar contacto con un proceso de desarrollo en etapas tempranas de su formación académica.

Por otra parte dados los nuevos planes de estudio de las carreras de sistemas se observa que este trabajo puede colaborar con la articulación entre las distintas áreas de conocimiento dado que se vinculan todas las tareas que se requieren para la generación de un producto software dentro de un marco de trabajo único.

Si bien en el ámbito local es frecuente encontrar dificultades a la hora de intentar aplicar procesos de ingeniería en el desarrollo de productos software, se considera de importancia impulsar desde la Universidad la adopción de estas buenas prácticas en el trabajo del profesional de sistemas.

Como se comentó en apartados anteriores, existe la posibilidad de aplicar esta metodología en distintas asignaturas de la carrera, en desarrollos dentro de la universidad e incluso que la misma sea utilizada en otras organizaciones, lo cual genera grandes expectativas de que este trabajo se mantenga en constante evolución, por lo que anhelamos que esta tesis resulte un pequeño aporte a la estandarización de procesos de desarrollo de software.

# Referencias

---

1. **Universidad Nacional de la Patagonia Austral. Unidad Académica Río Gallegos.** Sitio Web de las Carreras de Sistememas. [En línea] [Citado el: 5 de Diciembre de 2009.] <http://sistemas.uarg.unpa.edu.ar>.
2. Extreme Programming. [En línea] [Citado el: 3 de Enero de 2010.] <http://www.extremeprogramming.org/>.
3. Scrum. [En línea] [Citado el: 23 de Diciembre de 2010.] <http://www.scrumalliance.org/>.
4. Open UP. [En línea] [Citado el: 23 de febrero de 2010.] <http://epf.eclipse.org/wikis/openup/>.
5. **Booch, Grady, Rumbaugh, James y Jacobson, Ivar.** *El Proceso Unificado de Desarrollo de Software*. Madrid : Pearson Educación S.A., 2000. ISBN: 84-7829-036-2.
6. **Scrach, Stephen.** *Ingeniería de Software Clásica y Orientada a Objetos Edición Número 6*. s.l. : MCGRAW-HILL, 2006. ISBN 9789701056363.
7. **Pressman, Roger.** *Ingeniería de Software. Un Enfoque Práctico*. Quinta Edición. s.l. : McGraw Hill, 2002. ISBN: 8448132149.
8. **Whitten, Jeffrey.** *Análisis y Diseño de Sistemas de Información*. s.l. : McGraw-Hill , 1996. SBN: 84-8086-252-1.
9. **Sommerville, Ian.** *Ingeniería de Software 7ma Edicion*. s.l. : Pearson, 2006. ISBN: 8478290745.
10. Laboratorio de Investigación y Formación en Informática Avanzada. [En línea] Facultad de Informática. UNLP. [Citado el: 3 de Diciembre de 2009.] <http://www.lifia.info.unlp.edu.ar>.
11. **Gesto, E.; Prats, M.; Hallar, K.; Sofía, O.** [ed.] UNPA. *Horizon. Sistema de Administración Georeferencial de Información Muestral*. [Proceeding]. Río Gallegos, Santa Cruz : UNPA, Octubre de 2007.
12. RUP. [En línea] [Citado el: 1 de Enero de 2010.] [http://www-01.ibm.com/software/awdtools/rup/?S\\_TACT=105AGY59&S\\_CMP=WIKI&ca=dtl-08rupsite](http://www-01.ibm.com/software/awdtools/rup/?S_TACT=105AGY59&S_CMP=WIKI&ca=dtl-08rupsite).
13. **Humphrey, Watts.** *Introduction to the Personal Software Process*. s.l. : Pearson Education Limited, 2001. ISBN-10: 0201548097.
14. **Humphrey, Watts .** *Introduction to the Team Software Process*. s.l. : Pearson Education Limited, 2000. ISBN: 020147719x.
15. Software Engineering Institute. [En línea] [Citado el: 23 de Enero de 2010.] <http://www.sei.cmu.edu/>.

16. Ready Set. [En línea] [Citado el: 19 de Enero de 2010.] <http://readysset.tigris.org>.
17. Proyecto MeRinde. [En línea] [Citado el: 12 de Diciembre de 2009.] <http://merinde.org.ve>.  
merinde.
18. Modelo de Procesos para la Industria de Software. *MoProSoft*. [En línea] [Citado el: 2 de Enero de 2010.] <http://www.comunidadmoprosoft.org.mx>.
19. SPEM. [En línea] [Citado el: 2 de Enero de 2010.]  
<http://www.omg.org/technology/documents/formal/spem.htm>.
20. Proyecto Tutelkán. [En línea] [Citado el: 9 de Diciembre de 2009.] <http://www.tutelkan.org>.
21. **Association, IEEE Standard**. IEEE Standard for Developing Software Life Cycle Processes - Description. [En línea] 1997. [Citado el: 2 de Diciembre de 2009.]  
[http://standards.ieee.org/reading/ieee/std\\_public/description/se/1074-1997\\_desc.html](http://standards.ieee.org/reading/ieee/std_public/description/se/1074-1997_desc.html).
22. **Object Management Group**. OMG Web Site. [En línea] <http://www.omg.org/>.
23. **Proyecto GNU**. Copyleft. [En línea] [Citado el: 2 de Diciembre de 2009.]  
<http://www.gnu.org/copyleft>.
24. **Weitzenfeld, Alfredo**. *Ingeniería de Software Orientada a Objetos con UML, Java e Internet*. s.l. : Thomson International, 2003. ISBN 970-686-190-4.
25. **Fowler, Scott**. *UML Gota a Gota*. México : Addison Wesley. ISBN: 9684443641.
26. **Booch, Grady, Rumbaugh, James y Jacobson, Ivar**. *El lenguaje Unificado de Modelado*. Madrid : Pearson Educación S.A., 2000. ISBN: 84-7829-028-1.
27. **Larman, Craig**. *UML y Patrones*. Segunda Edición. s.l. : Pretince-Hall, 2003. ISBN: 8420534382 .
28. **Object Management Group**. Software Process Engineering Meta-Model, Version 2.0. [En línea] <http://www.omg.org/technology/documents/formal/spem.htm>.
29. **The Eclipse Foundation**. Eclipse Process Framework Composer. [En línea] [Citado el: 3 de Enero de 2010.] <http://www.eclipse.org/epf>.
30. IBM Composer. [En línea] [Citado el: 23 de Diciembre de 2009.] <http://www-01.ibm.com/software/awdtools/rmc/features/index.html>.
31. **Gesto, E; Hallar, K.; Prats, M**. Metodología de Desarrollo PSI. [En línea] 2010.  
<http://sistemas.uarg.unpa.edu.ar/psi>.
32. **The Eclipse Foundation**. Eclipse Public License - v 1.0. [En línea]  
<http://www.eclipse.org/org/documents/epl-v10.php>.
33. XVIII Congreso Iberoamericano de Educación Superior en Computación, CIESC 2010. [En línea] 18 al 22 de Octubre de 2010. Asunción, Paraguay. <http://www.clei2010.org.py>.

34. 39 Jornadas Argentinas de Informática. *JAIIO*. [En línea] 30 de Agosto al 3 de Septiembre de 2010. UADE, Buenos Aires, Argentina. <http://www.39jaiio.org.ar>.
35. **Zavala, R.** Diseño de un Sistema de Información Geográfica sobre internet. Tesis de Maestría en Ciencias de la Computación. *Universidad Autónoma Metropolitana-Azcapotzalco. México, D.F.* [En línea]  
<http://www.angelfire.com/scifi/jzavalar/apuntes/IngSoftware.html>.
36. **Guerrero, Luciano.** Mejoramiento de Procesos, El modelo IDEAL. [En línea]  
[http://www.geocities.com/SiliconValley/Lab/3629/IDEAL\\_ciclo.pdf](http://www.geocities.com/SiliconValley/Lab/3629/IDEAL_ciclo.pdf).
37. **García, Félix, Piattini, Mario y Pino, Francisco.** *Contribución de los estándares internacionales a la gestión de Procesos de Software*. Universidad de Cauca. Colombia : s.n.
38. Software process engineering systems: models and industry. [En línea] [Citado el: 1 de Marzo de 2010.] <http://herkules.oulu.fi/isbn9514265084/html/x287.html>.
39. **Ruiz, Francisco.** Procesos de Ingeniería del Software. [En línea] [Citado el: 3 de Enero de 2010.] <http://personales.unican.es/ruizfr/is1/doc/teo/02/is1-t02-trans.pdf>.

# Anexo I. Disciplinas

Flujos de trabajo del Proceso	
Modelado del Negocio	
 <b>Describir el Negocio Actual</b>	
	Mantener las Reglas de Negocio
	Analizar la Arquitectura del Negocio
	Encontrar los Actores y los Casos de Uso del Negocio
	Evaluar la Organización Objetivo
 <b>Definir el Negocio</b>	
	Identificar los Procesos del Negocio
	Diseñar las Realizaciones de los Procesos del Negocio
	Refinar los Roles y las Responsabilidades
 <b>Desarrollar el Modelo de Dominio</b>	
	Mantener las Reglas del Negocio
	Detallar las Entidades del Negocio
	Analizar la Arquitectura del Negocio

## Captura de Requerimientos



### Analizar el problema



Establecer la Visión del Sistema



Determinar la Terminología a Utilizar (Glosario)



Determinar los Actores y los Casos de Uso



### Definir el Sistema



Establecer la Visión del Sistema



Desarrollar Especificaciones Suplementarias



Detallar los Casos de Uso



Determinar la Terminología a usar



### Gestionar los Cambios de Requerimientos



Estructurar el Modelo de Casos de Uso



### Entender las necesidades de los Involucrados



Desarrollar Especificaciones Suplementarias



Detallar los Casos de Uso



Establecer la Visión del Sistema



### Gestionar el Alcance del Sistema



Dar Prioridad a los Casos de Uso



Establecer la Visión del Sistema

 <b>Refinar la Definición del Sistema</b>	
	Detallar los Casos de Uso
	Desarrollar Especificaciones Suplementarias
	Detallar los Requerimientos del Software
<b>Análisis y Diseño</b>	
 <b>Definir la Arquitectura Candidata</b>	
	Analizar la Arquitectura
	Analizar los Casos de Uso
 <b>Analizar el Comportamiento del Sistema</b>	
	Identificar los Elementos de Diseño
	Analizar los casos de Uso
	Diseñar la Interfaz del Usuario
	Generar Prototipo de la Interfaz del Usuario
 <b>Refinar la Arquitectura</b>	
	Identificar los Elementos de Diseño
	Describir la Arquitectura en Tiempo de Ejecución
	Describir la Distribución



 <b>Diseñar los Componentes</b>	
	Diseñar Clases
	Diseñar Subsistemas
	Diseñar Casos de Uso
	Diseñar los Elementos Soporte de Prueba
 <b>Diseñar la Base de Datos</b>	
	Diseñar Clases
	Diseñar la Base de Datos
<b>Implementación</b>	
 <b>Estructurar el Modelo de Implementación</b>	
	Estructurar el Modelo de Implementación del Sistema
 <b>Implementar Componentes</b>	
	Analizar el Comportamiento en Tiempo de Ejecución
	Implementar los Elementos de Diseño
	Ejecutar Pruebas a los Elementos y Subsistemas de Implementación
	Revisar el Código
 <b>Integrar el Sistema</b>	
	Integrar el Sistema

## Pruebas



### Definir la Misión de las Pruebas



Identificar la Misión de las Pruebas



Identificar los Motivadores de las Pruebas



Identificar los Objetos de Pruebas



Definir el Enfoque de Pruebas



Acordar la Misión de las Pruebas



Definir Criterios de Aceptación de los Casos de Prueba



### Verificar el Enfoque de las Pruebas



Establecer la Configuración del Ambiente de Pruebas



Definir los Detalles de las Pruebas



Acordar las Pruebas a Realizar



### Probar y Evaluar el Sistema



Definir los Detalles de las Pruebas



Implementar las Pruebas



Ejecutar el Conjunto de Pruebas



Determinar los Resultados de las Pruebas



Analizar Pruebas Fallidas

 <b>Evaluar las Pruebas en Términos de su Misión</b>	
	Determinar los Resultados de las Pruebas
	Evaluar y Defender la Calidad
<b>Implantación</b>	
 <b>Desarrollar Material de Apoyo</b>	
	Elaborar el Plan de Adiestramiento
	Desarrollar los Materiales de Adiestramiento
	Desarrollar Materiales de Apoyo
	Desarrollar Instalador de Componentes
 <b>Planificar la Implantación</b>	
	Definir el Listado de Materiales
	Desarrollar el Plan de Implantación
 <b>Producir Unidad de Implantación</b>	
	Crear Unidad de Implantación
	Elaborar Notas de Lanzamiento

## Flujos de trabajo de Soporte

### Gestión de Configuraciones



#### Gestionar Cambios de Requerimientos



Confirmar Cambios en el Sistema



Enviar Solicitud de Cambio



Revisar Solicitudes de Cambio



Confirmar o Rechazar Cambios de Requerimientos



Programar y Asignar el Trabajo



#### Planear la Configuración del Proyecto y Control de Cambios



Escribir el Plan de Gestión de Configuración



Establecer las Políticas de Gestión de Configuración



Establecer los Procesos de Control de Cambios



#### Registrar y Almacenar los Cambios



Guardar y Registrar Cambios

### Gestión del Proyecto



#### Cerrar-Salir de la Fase















Preparar Cierre-Salida para la Fase



Supervisar los Hitos del Ciclo de Vida

 <b>Cerrar-Salir del Proyecto</b>	
	Preparar Cierre-Salida para el Proyecto
	Evaluar la Aceptación del Proyecto
 <b>Concebir un Nuevo Proyecto</b>	
	Elaborar Solicitud del Sistema
	Desarrollar Términos de Referencia
	Identificar y Evaluar los Riesgos
	Iniciar el Proyecto
 <b>Evaluar el Alcance y los Riesgos del Proyecto</b>	
	Determinar el Alcance del Sistema
	Identificar y Evaluar los Riesgos
 <b>Gestionar Iteración</b>	
	Iniciar Iteración
	Identificar y Evaluar los Riesgos
	Revisar los Criterios de Evaluación de la Iteración

 <b>Monitorear y Controlar Proyecto</b>	
	Organizar Evaluación
	Conducir Evaluación del Proceso de Desarrollo
	Conducir Evaluación del Proyecto
	Programar y Asignar tareas
	Monitorear el Estado del Proyecto
	Solventar Problemas
 <b>Planear el Proyecto</b>	
	Planear las Fases y las Iteraciones
	Definir la Organización del Proyecto y del Personal
	Selección y Contratación del Personal de Desarrollo
	Desarrollar el Plan de Gestión de Riesgos
	Definir los Mecanismos de Monitoreo y Control del Proceso
	Revisar la Planificación del Proyecto
 <b>Planear el Resto de la Iteración Inicial</b>	
	Desarrollar el Plan de Iteración
	Revisar el Plan de Iteración



### Planificar la Próxima Iteración



Desarrollar el Plan de Iteración



Revisar el Plan de Iteración

## Gestión del Ambiente



### Apoyar el Ambiente durante la Iteración



Apoyar el Desarrollo



### Preparar el Ambiente para el Proyecto



Elaborar el Marco de Desarrollo del Proyecto



Determinar los Lineamientos del Proyecto



Adaptar el Marco de Desarrollo para el Proyecto



Preparar las Plantillas para el Proyecto



Seleccionar y Adquirir Herramientas



Configurar las Herramientas



Verificar la Instalación y Configuración de las Herramientas

## Gestión de Calidad



### Concebir el Plan de Gestión de Calidad



Elaborar el Plan de Calidad



Elaborar Estándares, Prácticas y Convenciones



Definir Herramientas y Técnicas



Definir Métricas



Definir Responsables de la Calidad








### Evaluar la Calidad









Generar Plan de Revisión de Calidad



## Anexo II. Roles

Trabajadores		
Imagen	Rol	Descripción
	Analista	Las personas en este rol representan al cliente y los usuarios finales involucrados, obteniendo información desde los <i>stakeholders</i> para entender el problema a ser resuelto y capturar y ajustar las prioridades para los requerimientos.
	Arquitecto	Este rol es el responsable de diseñar la arquitectura del software, la cual incluye tomar las principales decisiones técnicas que condicionan globalmente el diseño y la implementación del proyecto.
	Diseñador	
	Líder del Proyecto	<p>Este rol se encarga de establecer las condiciones de trabajo. Por tal motivo tiene la función de dirigir y asignar recursos, coordina las interacciones con los clientes y usuarios finales, planifica las iteraciones, asigna el trabajo, define la organización del proyecto, establece las prácticas que aseguran la integridad y calidad de los artefactos del proyecto, entre otras responsabilidades.</p> <p>Sinónimos: Administrador del Proyecto</p>
	Programador	<p>La persona en este rol es responsable por desarrollar una parte del sistema, incluyendo diseñar esta para que se ajuste a la arquitectura, posiblemente prototipar la interfaz de usuario y entonces implementar, hacer pruebas unitarias e integrar los componentes que son parte de la solución.</p> <p>Sinónimos: Desarrollador</p>

	Ingeniero de Pruebas	<p>Realizar las pruebas identificadas y definidas previamente, utilizando las instrucciones, métodos y herramientas necesarias.</p> <p>Sinónimos: Probador</p>
	Cliente	<p>Este rol es el responsable de contratar el desarrollo del nuevo sistema.</p>
	Documentador	<p>El objetivo principal del rol de documentador es el de mantener la información generada durante el proceso de desarrollo.</p>
	Gerente de Calidad	
	Rol General	<p>Cualquiera en un equipo puede cumplir este rol para llevar a cabo tareas generales.</p>
	Administrador de la Configuración	<p>Este rol es el responsable de administrar todo lo relacionado con los elementos de configuración del proyecto, es quien crea y administra el servidor de configuración donde todos los demás integrantes del grupo de desarrollo accederán para utilizar los recursos del proyecto.</p>

## Anexo III. Productos de Trabajo

### Requerimientos

	Plantilla	Ejemplo
Especificación de Requerimientos	•	•
Resumen de Entrevista	•	
Reunión de Requerimientos	•	

### Análisis y Diseño

	Plantilla	Ejemplo
Arquitectura	•	•
Modelo de Casos de Uso	•	
◆ Especificación de Caso de Uso	•	•
Modelo de Datos	•	
Modelo de Diseño	•	
◆ Prototipo de Interfaz	•	•
Modelo de Negocio	•	•
Modelo de Visión	•	•

## Implementación

	Plantilla	Ejemplo
Plan de Integración	•	•

## Pruebas

	Plantilla	Ejemplo
Caso de Prueba	•	•
Informe de Evolución de Defectos	•	
Informe de Revisión Técnica Formal	•	•
Informe de Verificación Unitaria	•	
Plan de Pruebas	•	•
♦ Solicitud de Cambios	•	

## Implantación

	Plantilla	Ejemplo
Informe de Consolidación	•	•
Manual de Usuario	•	•
♦ Glosario	•	•
Manual de Instalación	•	•

## Gestión de Calidad

	Plantilla	Ejemplo
Gestión de Riesgos		
◆ Plan de Gestión de Riesgos	•	•
• Informe de Riesgos	•	
• Seguimiento de Riesgos	•	
◆ Riesgos - Anexo I	•	•
Gestión de Configuraciones		
◆ Plan de Gestión de Configuración	•	•
◆ Manejo del Ambiente Controlado	•	
Plan de SQA		
◆ Estándar de Documentación	•	
◆ Estándar de Programación en Java	•	
◆ Informe Final de SQA	•	
◆ Plantilla Base de Formato	•	
◆ Revisión de SQA	•	

## Gestión del Proyecto

	Plantilla	Ejemplo
Plan de Proyecto	•	•
◆ Plan de Estimación	•	
Estimación	•	•
◆ Plan de Iteración	•	
◆ Resumen de Reunión	•	•

## Propuesta

	Plantilla	Ejemplo
Propuesta de Desarrollo	•	
Estudio de Factibilidad	•	

## Anexo IV. Catálogo de Software

### Eclipse

<http://www.eclipse.org>



Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma.

Eclipse dispone de un Editor de texto con resaltado de sintaxis. La compilación es en tiempo real. Tiene pruebas unitarias con JUnit, control de versiones con CVS, integración con Ant, asistentes (wizards) para creación de proyectos, clases, tests, etc., y refactorización.

Asimismo, a través de "plugins" libremente disponibles es posible añadir control de versiones con Subversion e integración con Hibernate.

### Subversion

<http://subversion.tigris.org>



Subversion es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS, el cual posee varias deficiencias. Es software libre bajo una licencia de tipo Apache/BSD y se le conoce también como svn por ser ese el nombre de la herramienta de línea de comandos. Una característica importante de Subversion es que, a diferencia de CVS, los archivos versionados no tienen cada uno un número de revisión independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo.

Subversion puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintos ordenadores. A cierto nivel, la capacidad para que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración. Se puede progresar más rápidamente sin un único conducto por el cual deban pasar todas las modificaciones. Y puesto que el trabajo se encuentra bajo el control de versiones, no hay razón para temer por que la calidad del mismo vaya a verse afectada por la pérdida de ese conducto único.

### Tortoise SVN

<http://tortoisesvn.tigris.org>



TortoiseSVN es un cliente Subversion, implementado como una extensión al shell de Windows. Es software libre liberado bajo la licencia GNU GPL.

## RapidSVN

<http://rapidsvn.tigris.org>



RapidSVN es una plataforma visual para el sistema Subversion escrito en C++. Se utiliza un nuevo marco wxWidgets y también se incluye un cliente de Subversion C++ API en el proyecto

## OpenOffice.org

<http://www.openoffice.org>



Es una aplicación que está siendo usada por distintas dependencias gubernamentales y empresas privadas de todo el mundo para realizar sus documentos de ofimática, tanto por la posibilidad de su uso sin el pago de una licencia, como por la apertura del formato de archivo con que trabaja, que está públicamente documentado.

OpenOffice incluye un procesador de textos, una hoja de cálculo y una herramienta para crear presentaciones, compatibles respectivamente con Word, Excel y PowerPoint.

También es posible generar archivos .PDF (Portable Document Format) de los documentos realizados y grabar en el formato utilizado por Word y Excel.

## Umbrello UML Modeller

<http://uml.sourceforge.net>



Es una herramienta libre para crear y editar diagramas UML, que ayuda en el proceso del desarrollo de software. Fue desarrollada por Paul Hensgen.

Umbrello maneja gran parte de los diagramas estándar UML pudiendo crearlos, además de manualmente, importándolos a partir de código en C++, Java, Python, IDL, Pascal/Delphi, Ada, o también Perl (haciendo uso de una aplicación externa). Así mismo, permite crear un diagrama y generar el código automáticamente en los lenguajes antes citados, entre otros. El formato de fichero que utiliza está basado en XML.

Umbrello se distribuye en el módulo kdesdk de KDE.



## Argo UML



<http://argouml.tigris.org>

Es una aplicación de diagramado de UML escrita en Java y publicada bajo la Licencia BSD open source. Dado que es una aplicación Java, está disponible en cualquier plataforma soportada por Java.

El Magazine de Desarrollo de Software entrega premios anuales a herramientas de desarrollo de software populares en varias categorías. En 2003 ArgoUML fue una de las finalistas en la categoría "Design and Analysis Tools". ArgoUML recibió un premio "runner-up"(revelación), derrotando a muchas herramientas comerciales.

## IzPack



<http://izpack.org/>

Es un generador de Instaladores de software para los archivos fuentes de la plataforma Java. Genera instaladores que caben en los archivos JAR.

Por esta característica los instaladores de IzPack son multiplataforma y funcionan en cualquier sistema operativo IzPack es liberado bajo los términos de la versión liberal Apache License 2.0.

## MySQL



<http://www.mysql.com/>

Es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. Desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009. MySQL se desarrolla como software libre en un esquema de licenciamiento dual.

## Apache

<http://httpd.apache.org>



El servidor HTTP Apache es un servidor web HTTP de código abierto para plataformas Unix (Solaris, BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual.

El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation.

Apache presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

## Notepad++

<http://notepad-plus.sourceforge.net>



Notepad++ es un editor de texto y de código fuente libre con soporte para varios lenguajes de programación para Microsoft Windows.

Gracias a su velocidad, puede convertirse en una alternativa al bloc de notas. Con la implementación de navegación por pestañas, moverse entre los archivos de texto abiertos es más cómodo.

Por defecto incluye la extensión TextFX que proporciona muchas opciones de transformación de texto.

## Filezilla

<http://filezilla-project.org>



Es un cliente FTP, gratuito, libre (GNU) y de código abierto. Sustenta FTP, SFTP y sobre SSL. Inicialmente sólo diseñado para funcionar bajo Windows, desde la versión 3.0.0, gracias al uso de wxWidgets, es multiplataforma, estando disponible además para otros sistemas operativos, entre ellos Linux, Solaris, FreeBSD y MacOS X.

Las principales características son el Site Manager (Administrador de sitios), Message Log (Registro de mensajes), y Transfer Queue (Cola de transferencia).

## Firefox



<http://www.mozilla.com/es-ES>

Es un navegador web libre descendiente de Mozilla Application Suite, desarrollado por la Corporación Mozilla, la Fundación Mozilla y un gran número de voluntarios externos

Firefox es un navegador multiplataforma y está disponible en varias versiones de Microsoft Windows, Mac OS X, GNU/Linux y algunos sistemas basados en Unix. Su código fuente es software libre, publicado bajo una triple licencia GPL/LGPL/MPL.

## 7-Zip



<http://www.7-zip.org>

7-Zip es un programa libre para la compresión de datos para sistemas Microsoft Windows (con interfaz gráfica de usuario), DOS y Linux.

Por defecto el programa utiliza el nuevo formato de archivo 7z, también libre, (con extensión .7z). Este formato usa los métodos de compresión LZMA y PPMd (más adecuado para textos), desarrollados por su autor, y puede aplicar un filtro a los ejecutables para aumentar su compresibilidad. Los archivos 7z pueden ser sólidos, a diferencia de los zip, lo que mejora la compresión de conjuntos de archivos pequeños.

## The Gimp



<http://www.gimp.org>

GIMP (GNU Image Manipulation Program) es un programa de edición de imágenes digitales en forma de mapa de bits, tanto dibujos como fotografías. Es un programa libre y gratuito. Está englobado en el proyecto GNU y disponible bajo la Licencia pública general de GNU.

La primera versión de GIMP se desarrolló inicialmente en sistemas Unix y fue pensada especialmente para GNU/Linux. Existen versiones totalmente funcionales para Windows, para Mac OS X, y se incluye en muchas distribuciones GNU/Linux. También se ha portado a otros sistemas operativos, haciéndolo el programa de manipulación de gráficos disponible en más sistemas operativos. Se le puede considerar como una alternativa firme, potente y rápida a Photoshop para muchos usos, aunque no se ha desarrollado como un clon de él y posee una interfaz bastante diferente.

GIMP está disponible en varios idiomas, entre ellos: español, alemán, inglés, catalán, gallego, euskera, francés, italiano, ruso, sueco, noruego, coreano, neerlandés y en otras lenguas adicionales.

## OpenProj

OPENPROJ™

<http://openproj.org>

OpenProj es una aplicación gratuita y de código abierto para la gestión de proyectos y tareas avanzado que destaca por su usabilidad.

OpenProj está programado en Java y se distribuye bajo los terminos de CPAL license. Se encuentra disponible para distintas plataformas con instaladores .deb .rpm .dmg .msi tar.gz y .zip, los dos últimos de plataforma independiente. Además cuenta con las siguientes funciones adicionales:

- Visualización del diagrama de Red.
- Visualización del diagrama (*WBS – Work Breakdown Structure*)
- Visualización del diagrama (*RBS – Resouce Breakdown Structure*)
- Visualización de un informe que el programa genera automáticamente.
- Histograma
- Gráficos generales

## MySQL Workbench

<http://wb.mysql.com>



Una base de datos puede llegar a ser muy compleja. Diseñar de antemano su esquema no sólo sirve para crear nuevas bases, sino también para documentar una existente o migrar otra a MySQL.

MySQL Workbench es la herramienta oficial de MySQL para el diseño visual de esquemas de bases de datos. En la pestaña MySQL Model se especifica la estructura física en tablas y vistas, con multitud de parámetros definibles en un panel inferior.

La parte más espectacular de MySQL Workbench es el editor de diagramas. Los elementos pueden arrastrarse al lienzo desde el catálogo o añadirse usando la caja de herramientas lateral. MySQL Workbench podrá exportar el diagrama como imagen o documento PDF, así como generar un script SQL CREATE o ALTER.

La versión de código abierto de MySQL Workbench dispone de todo lo necesario para el diseño de bases de datos, pero excluye herramientas tan jugosas como la validación del modelo o la sincronización en vivo.

## Dia

[http://dia-installer.de/index\\_en.html](http://dia-installer.de/index_en.html)



Dia es un editor de diagramas que sirve para crear todo tipo de diagramas: UML, entidad relación, diagramas de flujos, diagramas de red, diagramas eléctricos y muchos más.

El mismo es sencillo de usar y permite guardar los diagramas dibujados en los formatos EPS, SVG, XFIG, WMF y PNG. También puede cargar y guardar los diagramas en formato XML.

Dia tiene funciones muy similares al programa Microsoft Visio, la diferencia más notable entre estos dos programas es que Dia es gratis.

## ZETA TEST

<http://www.zeta-test.com>



Zeta Test es un ambiente de administración de pruebas integrado que le permite ejecutar pruebas de caja-negra, caja-blanca, pruebas de regresión o pruebas de administración de cambio de programas de aplicación.

Zeta Test ayuda a planificar, ejecutar, registrar, verificar y documentar las pruebas, para luego evaluar los resultados de las mismas.

Zeta Test es gratuito, libre de cargos para uso personal y para uso en el campo de instituciones educativas, así como para ser utilizado en el contexto de Código Libre.

## REM



[http://www.lsi.us.es/descargas/descarga\\_programas.php?id=3](http://www.lsi.us.es/descargas/descarga_programas.php?id=3)

REM (REquirements Management) es una herramienta experimental gratuita de Gestión de Requisitos diseñada para soportar la fase de Ingeniería de Requisitos de un proyecto de desarrollo software de acuerdo con la metodología definida en la Tesis Doctoral "Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información", presentada por Amador Durán en septiembre de 2000.