RUP

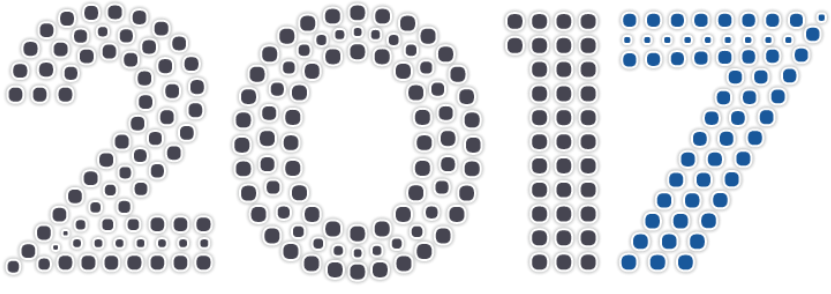
Rational Unified Process

Curso: ALGORITMICA II

Docente: ING. CORONEL CASTILLO, ERIC GUSTAVO

Integrantes:

- Díaz Núñez, Ivan
- Laura Fuerte, Daniel
- Ricra Chauca, Yordin
- Romucho Huaytan, Elena



INTRODUCCIÓN

En la actualidad, la utilización de metodologías para el desarrollo de aplicaciones es casi imposible omitirla, debido a la gran necesidad de control de variables que conlleva el mismo desarrollo, y para la ordenada elaboración de las aplicaciones, por lo tanto, seguir metodologías y estándares nos llevan a estar en competitividad en todo momento. Es de suma importancia conocer el modo como se interrelacionan metodologías con estándares y herramientas siguiendo un único propósito, el cual consiste en la elaboración de aplicaciones de manera eficiente, ordenada y con el menor número de defectos. La metodología RUP nos proporciona disciplinas en las cuales se encuentran artefactos con lo cual se podrá contar con guías para poder documentar e implementar de una manera fácil y eficiente, todas las guías para un buen desarrollo, todo esto dentro de las respectivas fases con las cuales cuenta.

INDICE

RESUMEN

Las metodologías y estándares utilizados en un desarrollo de software nos proporcionan las guías para poder conocer todo el camino a recorrer desde antes de empezar la implementación, con lo cual se asegura la calidad del producto final, así como también el cumplimiento en la entrega del mismo en un tiempo estipulado. Es de suma importancia elegir la metodología adecuada, así como las herramientas de implementación adecuadas, es por ello que la metodología RUP basada en UML nos proporciona todas las bases para llevar al éxito la elaboración del software, para ello la utilización de la herramienta RUP para el desarrollo rápido de aplicaciones. Este trabajo consta de siete capítulos, los cuales se describen a continuación: En el capítulo uno se abarcará la explicación de la metodología RUP con sus bases en el UML, las partes que la conforman, su funcionalidad; con lo cual podremos observar la interrelación entre ambos y la importancia de su uso en el desarrollo de aplicaciones. En el capítulo dos se abarcará lo que son las dimensiones de RUP dentro de ello específica los casos de uso y el proceso modelado a una arquitectura y los procesos iterativos. En el capítulo tres se abarca lo que son las fases y sus planeaciones los esfuerzos a los flujos de trabajo y a los de su respecto. En el capítulo cuatro se abarca lo que son las disciplinas como modelado de negocio, requerimientos, análisis, etc. En el capítulo cinco se abarca a lo que son las organizaciones del RUP como actores, artefactos y grados de artefacto. En el capítulo seis se trata de lo que es el Lenguaje Unificado de Modelado (UML) que se extiende hasta los casos de uso y diagramas y más ámbito de desarrollos de software. Y como final el capítulo siete se abarca a lo que son las metodologías de diseño y Análisis de software con RUP y UML.

ANTECEDENTE HISTORICO

El antecedente más importante se ubica en 1967 con la Metodología Ericsson (Ericsson Approach) elaborada por Ivar Jacobson, unaaproximación de desarrollo basada en componentes, que introdujo el concepto deCaso de Uso. Entre los años de 1987 a 1995 Jacobson fundó la compañía ObjectoryAB y lanza el proceso de desarrollo Objectory (abreviación de Object Factory).

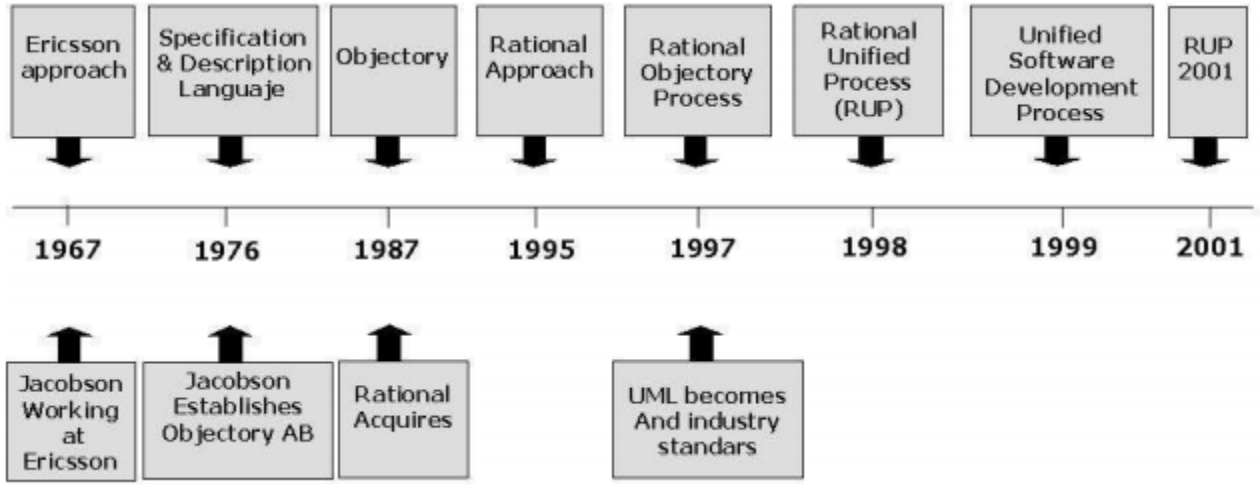


Figura 1: Historia de RUP

Posteriormente en 1995 Rational Software Corporation adquiere Objectory AB y entre1995 y 1997 se desarrolla Rational Objectory Process (ROP) a partir de Objectory 3.8 y del Enfoque Rational (RationalApproach) adoptando UML como lenguaje de modelado. Desde ese entonces y a la cabeza de Grady Booch, Ivar Jacobson y JamesRumbaugh, Rational Software desarrolló e incorporó diversos elementos para expandir RUP, destacándose especialmente el flujo de trabajo conocido como modelado delnegocio. En junio del 1998 se lanza RationalUnifiedProcess.

METODOLOGIA RUP

METODOLOGÍA PURA

Es una metodología cuyo fin es entregar un producto de software. Se estructura todos los procesos y se mide la eficiencia de la organización.
Es un proceso de desarrollo de software el cual utiliza el lenguaje unificado de modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.
El RUP es un conjunto de metodologías adaptables al contexto y necesidades de cada organiza

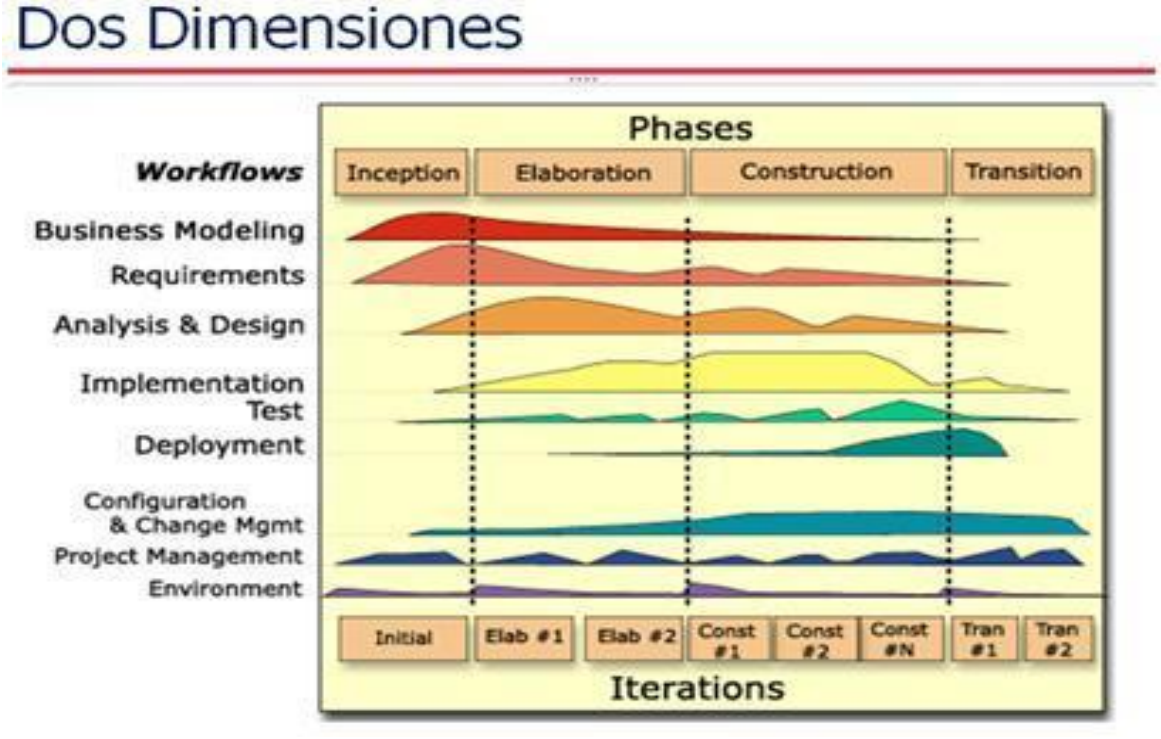
Principales características

- Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo)
- Pretende implementar las mejores prácticas en Ingeniería de Software.
- Desarrollo iterativo
- Administración de requisito
- Uso de arquitectura basada en componentes
- Control de cambios
- Modelado visual del software
- Verificación de la calidad del software

En esta metodología lo que se pretende es el desarrollo de un software, en el cual se aplicara el PSP y el CMMI en todos sus fases, que están en la realización de los procesos

El RUP es un producto de Rational (IBM). Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Incluye artefactos (que son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, etc.) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso).

- ✓ **CICLO DE VIDA.**



- ✓ ***Esfuerzo en actividades según fase del proyecto***

  El ciclo de vida RUP es una implementación del Desarrollo en espiral. Fue creado ensamblando los elementos en secuencias semi-ordenadas. El ciclo de vida organiza las tareas en fases e iteraciones.

  RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias

iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en las distintas actividades.

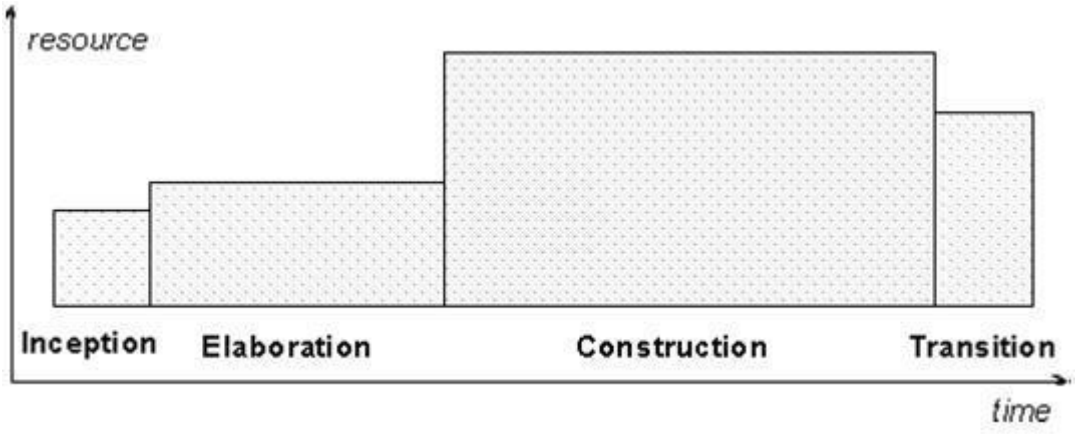
Fases del ciclo de vida del RUP:

1. Fase de Inicio: Esta fase tiene como propósito definir y acordar el alcance del proyecto con los patrocinadores, identificar los riesgos asociados al proyecto, proponer una visión muy general de la arquitectura de software y producir el plan de las fases y el de iteraciones posteriores.

2. Fase de elaboración: En la fase de elaboración se seleccionan los casos de uso que permiten definir la arquitectura base del sistema y se desarrollaran en esta fase, se realiza la especificación de los casos de uso seleccionados y el primer análisis del dominio del problema, se diseña la solución preliminar.

3. Fase de Desarrollo: El propósito de esta fase es completar la funcionalidad del sistema, para ello se deben clarificar los requerimientos pendientes, administrar los cambios de acuerdo a las evaluaciones realizados por los usuarios y se realizan las mejoras para el proyecto.

4. Fase de Cierre: El propósito de esta fase es asegurar que el software esté disponible para los usuarios finales, ajustar los errores y defectos encontrados en las pruebas de aceptación, capacitar a los usuarios y proveer el soporte técnico necesario. Se debe verificar que el producto cumpla con las especificaciones entregadas por las personas involucradas en el proyecto.



La metodología RUP tiene 6 principios clave:

1. Adaptación del proceso: El proceso debe adaptarse a las características de la organización para la que se está desarrollando el software.

2. Balancear prioridades: Debe encontrarse un balance que satisfaga a todos los inversores del proyecto.

3. Colaboración entre equipos: Debe haber una comunicación fluida para coordinar requerimientos, desarrollo, evaluaciones, planes, resultados, entre otros.

4. Demostrar valor iterativamente: Los proyectos se entregan, aunque sea de una forma interna, en etapas iteradas. En cada iteración se evaluará la calidad y estabilidad del producto y analizará la opinión y sugerencias de los inversore

5. Elevar el nivel de abstracción: Motivar el uso de conceptos reutilizables.

6. Enfocarse en la calidad: La calidad del producto debe verificarse en cada aspecto de la producción.

Disciplina de desarrollo de RUP

Determina las etapas a realizar durante el proyecto de creación del software.

- **Ingeniería o modelado del negocio:** Analizar y entender las necesidades del negocio para el cual se está desarrollando el software.
- **Requisitos:** Proveer una base para estimar los costos y tiempo de desarrollo del sistema.
- **Análisis y diseño:** Trasladar los requisitos analizados anteriormente a un sistema automatizado y desarrollar una arquitectura para el sistema.
- **Implementación:** Crear software que se ajuste a la arquitectura diseñada y que tenga el comportamiento deseado.
- **Pruebas:** Asegurarse de que el comportamiento requerido es correcto y que todo lo solicitado está presente.
- **Despliegue:** Producir distribuciones del producto y distribuirlo a los usuarios.

Disciplina de soporte RUP

Determina la documentación que es necesaria realizar durante el proyecto.

- **Configuración y administración del cambio:** Guardar todas las versiones del proyecto.
- **Administración del proyecto:** Administrar los horarios y recursos que se deben de emplear.
- **Ambiente:** Administrar el ambiente de desarrollo del software.
- **Distribución:** Hacer todo lo necesario para la salida del proyecto.

Elementos del RUP

- **Actividades:** Procesos que se han de realizar en cada etapa/iteración.
- **Trabajadores:** Personas involucradas en cada actividad del proyecto.
- **Artefactos:** Herramientas empleadas para el desarrollo del proyecto. Puede ser un documento, un modelo, un elemento del modelo.

Artefactos

RUP en cada una de sus fases (pertenecientes a la estructura estática) realiza una serie de artefactos que sirven para comprender mejor tanto el análisis como el diseño del sistema (entre otros). Estos artefactos (entre otros) son los siguientes:

Inicio:

- Documento Visión
- Especificación de Requerimientos

Elaboración:

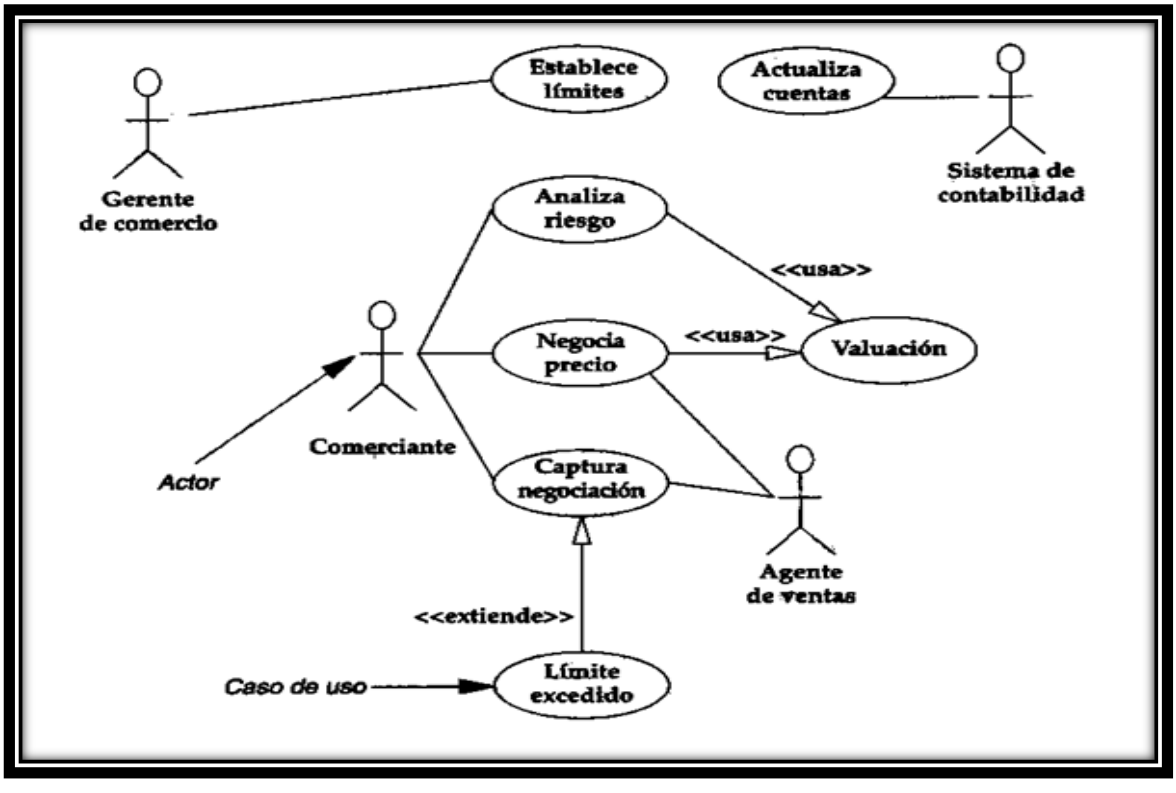
- Diagramas de caso de uso: Es una descripción de los pasos o las actividades que deberán realizarse para llevar a cabo algún proceso. Los personajes o entidades que participarán en un caso de uso se denominan actores.

Un caso de uso debe:

- Describir una tarea del negocio que sirva a una meta de negocio.
- Tener un nivel apropiado del detalle.
- Ser bastante sencillo como que un desarrollador lo elabore en un único lanzamiento.

Situaciones que pueden darse:

- Un actor se comunica con un caso de uso (si se trata de un actor primario la comunicación la iniciará el actor, en cambio sí es secundario, el sistema será el que inicie la comunicación).
- Un caso de uso extiende otro caso de uso.
- Un caso de uso utiliza otro caso de uso.



Construcción:

- Documento Arquitectura que trabaja con las siguientes vistas:

VISTA LÓGICA:

DIAGRAMA DE CLASES:

✓ En Lenguaje Unificado de Modelado (UML) es un tipo de **diagrama** de estructura estática que describe la estructura de un sistema mostrando las **clases** del sistema, sus atributos, operaciones (o métodos), y las relaciones entre los objetos.

MODELO E-R (SI EL SISTEMA ASÍ LO REQUIERE):

✓ Un **modelo entidad-relación** o **diagrama entidad-relación** (a veces denominado por sus siglas en inglés, **E-R** *"Entity relationship"*; en español **DER**: "Diagrama de Entidad-Relación") es una herramienta para el modelado de datos que permite representar las entidades relevantes de un sistema de información así como sus interrelaciones y propiedades.

VISTA DE IMPLEMENTACION:

DIAGRAMA DE SECUENCIA

✓ Es un tipo de **diagrama** usado para modelar interacción entre objetos en un sistema según UML.

DIAGRAMA DE ESTADOS

- ✓ Una transición simple es una relación entre dos **estados** que indica que un objeto en el primer **estado** puede entrar al segundo **estado** y ejecutar ciertas operaciones, cuando un evento ocurre y si ciertas condiciones son satisfechas.

DIAGRAMA DE COLABORACIÓN

- ✓ Un diagrama de comunicación es también un diagrama de clases que contiene roles de clasificador y roles de asociación en lugar de sólo clasificadores y asociaciones. Los roles de clasificador y los de asociación describen la configuración de los objetos y de los enlaces que pueden ocurrir cuando se ejecuta una instancia de la comunicación. Cuando se instancia una comunicación, los objetos están ligados a los roles de clasificador y los enlaces a los roles de asociación.

VISTA CONCEPTUAL

MODELO DE DOMINIO

- ✓ En la resolución de problemas e ingeniería de software, es un **modelo** conceptual de todos los temas relacionados con un problema específico. En él se describen las distintas entidades, sus atributos, papeles y relaciones, además de las restricciones que rigen el **dominio** del problema.

VISTA FISICA

MAPA DE COMPORTAMIENTO A NIVEL DE HARDWARE.

El mapa del comportamiento pretende precisamente clasificar a las personas en cuatro cuadrantes dependiendo de Cómo pensamos y de Cómo actuamos.

En el Cómo pensamos hemos de generar dos dimensiones:

- ✓ **Estructurado:** supone que la persona que dispone de esta forma de pensar es metódica y acude siempre a esquemas mentales y métodos muy definidos.
- ✓ **Desestructurado:** es su contrario. Aquí se piensa de una forma poco metódica, caótica y sin un patrón aparente marcado.

En el Cómo actuamos, las dimensiones son:

- ✓ **Programada:** en este caso, las actuaciones que se llevan a cabo responden a una programación y planificación clara realizada por esa persona.
- ✓ **Impulsiva:** es su contrario también. Aquí las actuaciones se producen de forma intuitiva y sin responder a una planificación previa.

CONCLUSIONES

Se puede concluir que, el RUP, como herramienta colaboradora en el desarrollo de software, aumenta la visión de desarrollo del mismo, es decir, el RUP es una herramienta que permite prever los cambios que un software pueda tener de acuerdo a los requerimientos y avance social que se tenga, brindando objetivos más amplios y visión de requerimientos global.

Visto desde su punto más simple, el RUP es aquel método que da cabida al cambio en las etapas del desarrollo de software, no siguiendo al pie de la letra los requerimientos, sino, por el contrario, mostrando otros campos que mejoren y optimicen el desarrollo del mismo.

RECOMENDACIONES

1. la metodología RUP cuenta con un enfoque disciplinado en la asignación de tareas y responsabilidades dentro de una organización del desarrollo, con la cual se puede mantener una fácil administración de este proceso.

2. Para obtener un máximo control de variables que conlleva un desarrollo de aplicaciones y poder mantener una ordenada implementación de éstas, es importante seguir metodologías y estándares que nos lleven a estar en competitividad en todo momento.

3. Al implementar un Desarrollo Rápido de Aplicaciones de un Sistema particular, es importante la utilización de Patrones, los cuales ya tienen una funcionalidad general y han sido predefinidos, y así contar con una base consistente y previamente elaborada para la implementación del Software.

BIBLIOGRAFÍA

1. Grupo Isaias Carrillos Perez. (2008) Metodología del desarrollo de software.New York:Editorial Edit and write.

2. IBM, Rational Software. (2003). Rational Rapid Developer, Technical Overview.EE.UU:IBM publications, World Wide Web.

3. ITSA (2008). Metodologías De Desarrollo De Software Canada:Editorial Canada Pen.

4. Jacaboson, I., Booch, G., Rumbaugh J. (2000) Proceso Unificado de Desarrollo deSoftware.New York:Editorial Mc Graw Hill.

5. Kruchten, P. (1995). Architectural Blueprints—the “4+1” View Model of SoftwareArchitecture.IEEE Software. 6. Marcos, E. (2005). “Investigación en Ingeniería del Software vs. DesarrolloSoftware”, Grupo KYBELE,Universidad Rey Juan Carlos.

7. Morgan, Stanley (2001). Sr. Business Analyst. New York:Editorial Pretince Hall.

8. Pressman, Roger. (2003) Ingenieria de Software 6ta edic.New York:Editorial McGraw Hill.

9. RUP/Easy. (2004). GUÍA METODOLÓGICA DE DESARROLLO DE SISTEMAS

10. Schmuller, Joseph. (2000) Aprendiendo UML en 24 horas.México:Editorial Prentice all.

11. Sommerville, I. (2005) Ingeniería del software, 7ª edición, PearsonAddison.España:Editorial Wesley.

12. Wasserfallmodell, Entstehungskontext, Markus Rerych, und Wirkungsforschung,(2007). TU-Wien de Gestaltungs- del für de Institut. Tenido acceso en línea 28 denoviembre.