Consejo Ejecutivo del Poder Judicial

RESOLUCIÓN ADMINISTRATIVA N° 195 -2007-CE-PJ

Lima, 14 de agosto del 2007

VISTO:

El Oficio N° 1400-2006-GG-PJ cursado por el Gerente General del Poder Judicial, elevando proyecto de Directiva denominada "Metodología para el Desarrollo de Sistemas Informáticos en el Poder Judicial", para su aprobación; y,

CONSIDERANDO:

Que, mediante Resolución Administrativa N° 011-2004-CE-PJ, de fecha 03 de febrero de 2004, se aprobó la Directiva N° 001-2004-CE-PJ "Estandarización del Software de Base en el Poder Judicial";

Que, la Contraloría General de la República por Resolución de Contraloría N° 072-98-CG, expedida el 02 de julio de 1998, aprobó la Norma de Control Interno 500-03 Controles de Datos Fuente, de Operación y de Salida en Sistemas Computarizados;

Que, resulta necesario complementar la labor de estandarización de la producción informática en el Poder Judicial, con la puesta en vigencia de la Metodología de Desarrollo que adoptará la institución en el desarrollo de los sistemas informáticos;

Que, la existencia de normas claras sobre el desarrollo de sistemas informáticos, contribuye a la ejecución de funciones de modo ordenado; además, posibilita el acceso gradual a la tecnología informática, hecho que finalmente coadyuvará a elevar los niveles de eficiencia del servicio de administración de justicia;

Por tales fundamentos, el Consejo Ejecutivo del Poder Judicial, en uso de las atribuciones, de conformidad con el informe del señor Consejero Wálter Cotrina Miñano, en sesión ordinaria de la fecha, por unanimidad;

RESUELVE:

Artículo Primero.- Aprobar la Directiva N° 006-2007-CE-PJ "Metodología para el Desarrollo de Sistemas Informáticos en el Poder Judicial", que en anexo forma parte integrante de la presente resolución.

Artículo Segundo.- Disponer que la Gerencia de Informática del Poder Judicial, proceda a la difusión de la presente resolución a todas las dependencias del Poder Judicial.

Artículo Tercero.- Transcribir la presente resolución al Presidente del Poder Judicial, Gerencia General y Gerencia de Informática del Poder Judicial, para su conocimiento y fines consiguientes.

Regístrese, comuníquese y cúmplase.

SS.

FRANCISCO TÁVARA CÓRDOVA

ANTONIO PAJARES PAREDES

JAVIER ROMÁN SANTISTEBAN

SONIA TORRE MUÑOZ

WÁLTER COTRINA MIÑANO

LUIS ALBERTO MENA NÚÑEZ

DIRECTIVA No. 006 -2007-CE-PJ

METODOLOGIA PARA EL DESARROLLO DE SISTEMAS INFORMATICOS EN EL PODER JUDICIAL

1. OBJETIVOS

- Establecer una Metodología para el desarrollo eficiente de sistemas informáticos que reflejen los lineamientos de política del Poder Judicial, con relación al desarrollo informático institucional con un horizonte de planeación de 5 años.
- Facilitar a la administración y técnicos informáticos, una herramienta de desarrollo homogéneo y acorde con los reales requerimientos para las actividades que desarrollan las dependencias del Poder Judicial.

2. FINALIDAD

- Producción homogénea de sistemas informáticos en el Poder Judicial.

3. ALCANCE

- Todas las Dependencias del Poder Judicial, a nivel nacional.

4. BASE LEGAL

- Resolución Administrativa No. 011-2004-CE-PJ, de fecha 03 de febrero de 2004, que aprueba la Directiva No. 001-2004-CE-PJ, "Estandarización del Software de Base en el Poder Judicial".
- Normas Técnicas de Control 500, aprobada con Resolución de Contraloría No. 072-98-CG de fecha 02 de julio de 1998; emitida por la Contraloría General de la República.
- Resolución Administrativa No. 161-2001-CE-PJ, aprueba el Reglamento de Organización y Funciones de la Gerencia General del Poder Judicial.

5. VIGENCIA

A partir de la fecha de aprobación de la presente Directiva.

6. DISPOSICIONES GENERALES

6.1. La Gerencia de Informática, es responsable de la ejecución de las políticas de desarrollo informático en el Poder Judicial, establecidas en el marco del planeamiento estratégico institucional.

6.2. A efecto de lograr el desarrollo organizado y homogéneo de los sistemas informáticos, el Poder Judicial utilizará el Rational Unified Process (RUP), como Metodología estándar de desarrollo informático.

6.3. Todas las dependencias del Poder Judicial, que requieran de modo particular el desarrollo de aplicativos, estos deberán ser canalizados a la Gerencia de Informática para la respectiva evaluación del requerimiento, previa a la decisión de atención.

6.4. El personal profesional y/o técnico de la Gerencia de Informática, procederá a desarrollar los sistemas informáticos institucionales, de acuerdo a la Metodología RATIONAL UNIFIED PROCESS (RUP).

7. DISPOSICIONES ESPECIFICAS

7.1. Dependencias del Poder Judicial

7.1.1. Los responsables de las distintas dependencias del Poder Judicial, deberán dirigir sus requerimientos por desarrollo de sistemas informáticos, a la Gerencia de Informática de la Gerencia General, para la respectiva evaluación.

7.1.2. Los requerimientos deberán estar alineados a los objetivos y metas institucionales, con relación al horizonte de planeación en orden al desarrollo informático.

7.2. Gerencia de Informática

7.2.1. Recibido el requerimiento, procederá a efectuar la evaluación, con relación a los lineamientos establecidos por el Consejo Ejecutivo.

7.2.2. De considerarlo pertinente, desarrollará el sistema informático, a través de la Sub Gerencia de Desarrollo de Sistemas Informáticos; gestión que hace de conocimiento de la Gerencia General.

Anexo

Directiva "Normas para el Desarrollo de Sistemas Informáticos en el Poder Judicial"

Metodologia

Rational Unified Process (RUP)

Este documento presenta un resumen de *Rational Unified Process* (RUP). Se describe la historia de metodología, características principales y estructura del proceso. RUP es un producto comerci desarrollado y comercializado por Rational Software, una compañía de IBM.

1 Historia

La Figura 1 ilustra la historia de RUP. El antecedente más importante se ubica en 1967 con la Metodologí Ericsson (*Ericsson Approach*) elaborada por Ivar Jacobson, una aproximación de desarrollo basada e componentes, que introdujo el concepto de Caso de Uso. Entre los años de 1987 a 1995 Jacobson fundó l compañía *Objectory* AB y lanza el proceso de desarrollo *Objectory* (abreviación de *Object Factory*).

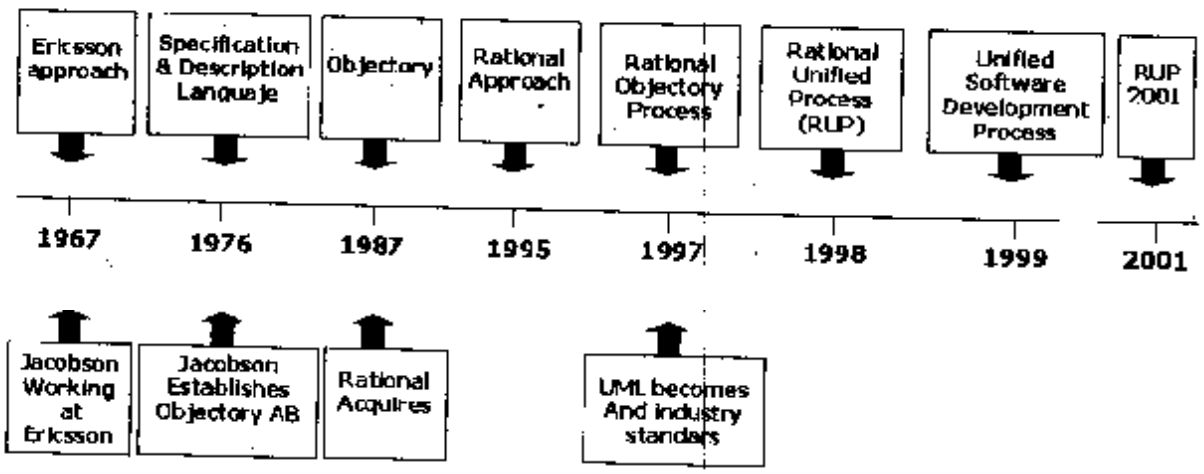

Figura 1: Historia de RUP

Posteriormente en 1995 *Rational Software Corporation* adquiere *Objectory AB* y entre 1995 y 1997 se desarrolla *Rational Objectory Process* (ROP) a partir de *Objectory* 3.8 y del Enfoque Rational (*Rationa Approach*) adoptando UML como lenguaje de modelado.

Desde ese entonces y a la cabeza de Grady Booch, Ivar Jacobson y James Rumbaugh, Rational Software desarrolló e incorporó diversos elementos para expandir ROP, destecándose especialmente el flujo de trabajo conocido como modelado del negocio. En junio del 1998 se lanza *Rational Unified Process*.

2 Características esenciales

Los autores de RUP destacan que el proceso de software propuesto por RUP tiene tres caracteristicas esenciales: está dirigido por los Casos de Uso, está centrado en la arquitectura, y es iterativo e incremental.

2.1 Proceso dirigido por Casos de Uso

Según [2], los Casos de Uso son una técnica de captura de requisitos que fuerza a pensar en términos de importancia para el usuario y no sólo en términos de funciones que seria bueno contemplar. Se define un Caso de Uso como un fragmento de funcionalidad del sistema que proporciona al usuario un valor añadido. Los Casos de Uso representan los requisitos funcionales del sistema.

En RUP los Casos de Uso no son sólo una herramienta para especificar los requisitos del sistema. También guían su diseño, implementación y prueba. Los Casos de Uso constituyen un elemento integrador y una guía del trabajo como se muestra en la Figura 2.

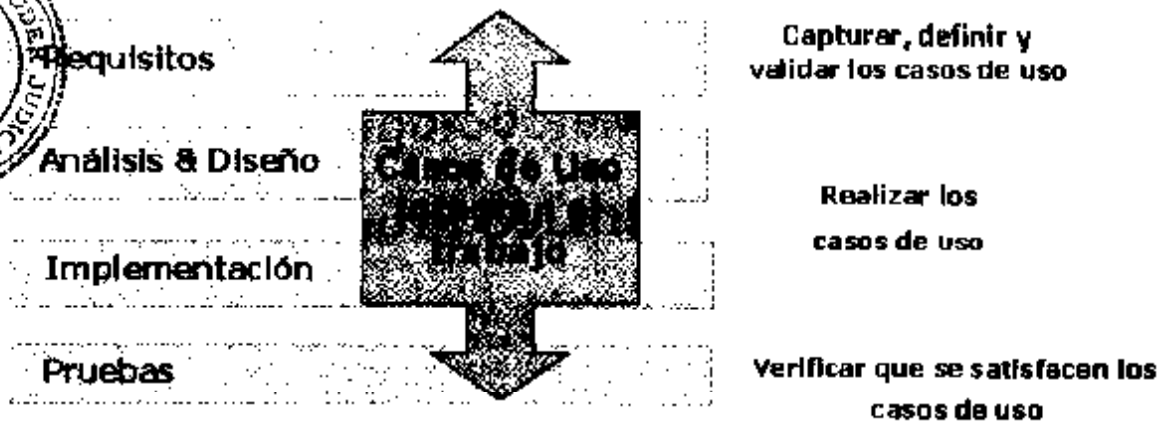

Figura 2: Los Casos de Uso integran el trabajo

Los Casos de Uso no sólo inician el proceso de desarrollo sino que proporcionan un hilo conducto permitiendo establecer trazabilidad entre los artefactos que son generados en las diferentes actividades d proceso de desarrollo.

Como se muestra en la Figura 3, basándose en los Casos de Uso se crean los modelos de análisis y diseñ luego la implementación que los lleva a cabo, y se verifica que efectivamente el producto implement adecuadamente cada Caso de Uso. Todos los modelos deben estar sincronizados con el modelo de Casc de Uso.

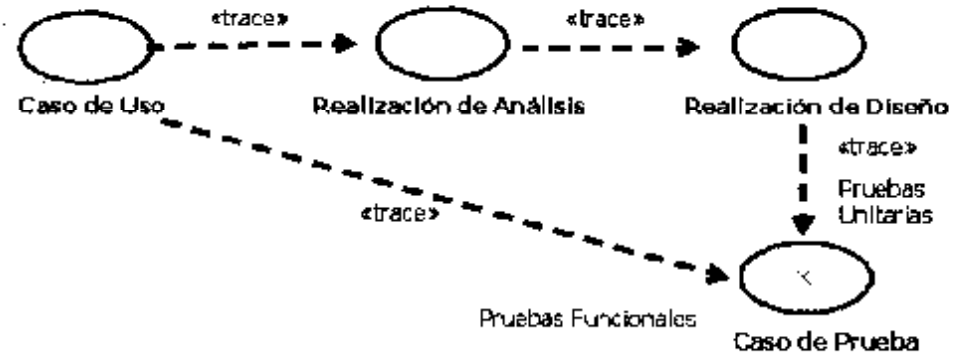

Figura 3: Trazabilidad a partir de los Casos de Uso

2.2 Proceso centrado en la arquitectura

La arquitectura de un sistema es la organización o estructura de sus partes más relevantes, lo que permi tener una visión común entre todos los involucrados (desarrolladores y usuarios) y una perspectiva clara d sistema completo, necesaria para controlar el desarrollo [2].

La arquitectura involucra los aspectos estáticos y dinámicos más significativos del sistema, está relacionac con la toma de decisiones que indican cómo tiene que ser construido el sistema y ayuda a determinar e qué orden. Además la definición de la arquitectura debe tomar en consideración elementos de calidad d sistema, rendimiento, reutilización y capacidad de evolución por lo que debe ser flexible durante todo proceso de desarrollo. La arquitectura se ve influenciada por la plataforma software, sistema operativ gestor de bases de datos, protocolos, consideraciones de desarrollo como sistemas heredados. Muchas c estas restricciones constituyen requisitos no funcionales del sistema.

En el caso de RUP además de utilizar los Casos de Uso para guiar el proceso se presta especial atención establecimiento temprano de una buena arquitectura que no se vea fuertemente impactada ante cambic posteriores durante la construcción y el mantenimiento.

Cada producto tiene tanto una función como una forma. La función corresponde a la funcionalidad reflejac en los Casos de Uso y la forma la proporciona la arquitectura. Existe una interacción entre los Casos de Us

y la arquitectura, los Casos de Uso deben encajar en la arquitectura cuando se llevan a cabo y arquitectura debe permitir el desarrollo de todos los Casos de Uso requeridos, actualmente y en el futur Esto provoca que tanto arquitectura como Casos de Uso deban evolucionar en paralelo durante todo proceso de desarrollo de software.

En la Figura 4 se ilustra la evolución de la arquitectura durante las fases de RUP. Se tiene una arquitectu más robusta en las fases finales del proyecto. En las fases iniciales lo que se hace es ir consolidando arquitectura por medio de *baselines* y se va modificando dependiendo de las necesidades del proyecto.

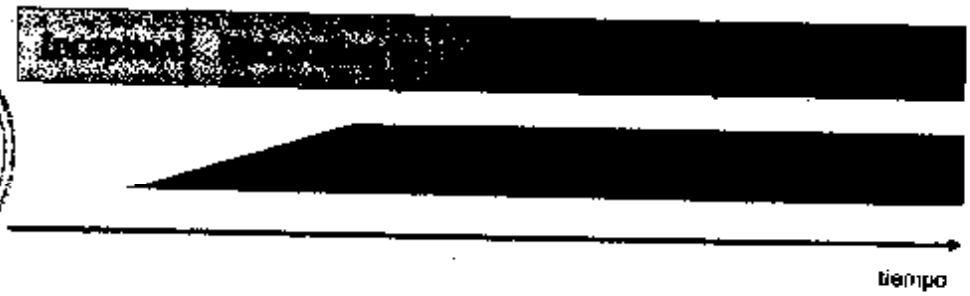

Figura 4: Evolución de la arquitectura del sistema

Es conveniente ver el sistema desde diferentes perspectivas para comprender mejor el diseño por lo que l arquitectura se representa mediante varias vistas que se centran en aspectos concretos del sistema abstrayéndose de los demás. Para RUP, todas las vistas juntas forman el llamado modelo 4+1 de l arquitectura [3], el cual recibe este nombre porque lo forman las vistas lógica, de implementación, d proceso y de despliegue, más la de Casos de Uso que es la que da cohesión a todas.

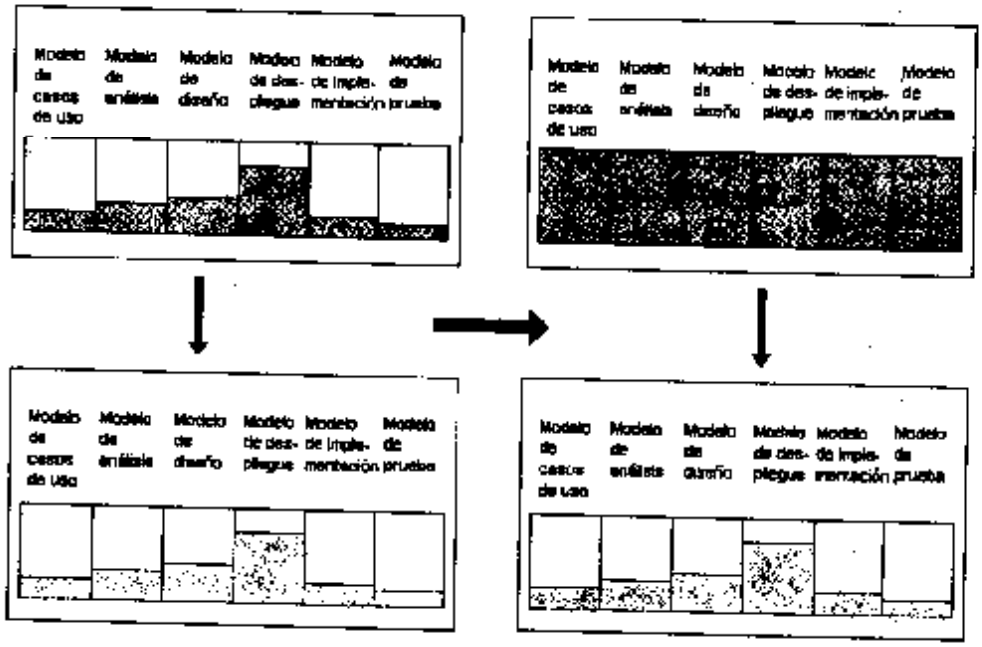

Figura 5: Los modelos se completan, la arquitectura no cambia drásticamente

Al final de la fase de elaboración se obtiene una *baseline*[1] de la arquitectura donde fueron seleccionados una serie de Casos de Uso arquitectónicamente relevantes (aquellos que ayudan a mitigar los riesgos más importantes, aquellos que son los más importantes para el usuario y aquellos que cubran las funcionalidades significativas)

Como se observa en la Figura 5, durante la construcción los diversos modelos van desarrollándose hasta completarse (según se muestra con las formas rellenas en la esquina superior derecha). La descripción de la arquitectura sin embargo, no debería cambiar significativamente (abajo a la derecha) debido a que la

1 Una baseline es una instantánea del estado de todos los artefactos del proyecto, registrada para efectos de gestión de configuración y control de cambios.

mayor parte de la arquitectura se decidió durante la elaboración. Se incorporan pocos cambios a l arquitectura (indicados con mayor densidad de puntos en la figura inferior derecha) [1].

2.3 Proceso iterativo e incremental

Según [1] el equilibrio correcto entre los Casos de Uso y la arquitectura es algo muy parecido al equilibrio d la forma y la función en el desarrollo del producto, lo cual se consigue con el tiempo. Para esto, la estrategi que se propone en RUP es tener un proceso iterativo e incremental en donde el trabajo se divide en parte más pequeñas o mini proyectos. Permitiendo que el equilibrio entre Casos de Uso y arquitectura se vay logrando durante cada mini proyecto, así durante todo el proceso de desarrollo. Cada mini proyecto s puede ver como una iteración (un recorrido más o menos completo a lo largo de todos los flujos de trabaj fundamentales) del cual se obtiene un incremento que produce un crecimiento en el producto.

Una iteración puede realizarse por medio de una cascada como se muestra en la Figura 6. Se pasa por lo flujos fundamentales (Requisitos, Análisis, Diseño, Implementación y Pruebas), también existe un planificación de la iteración, un análisis de la iteración y algunas actividades específicas de la iteración. A finalizar se realiza una integración de los resultados con lo obtenido de las iteraciones anteriores.

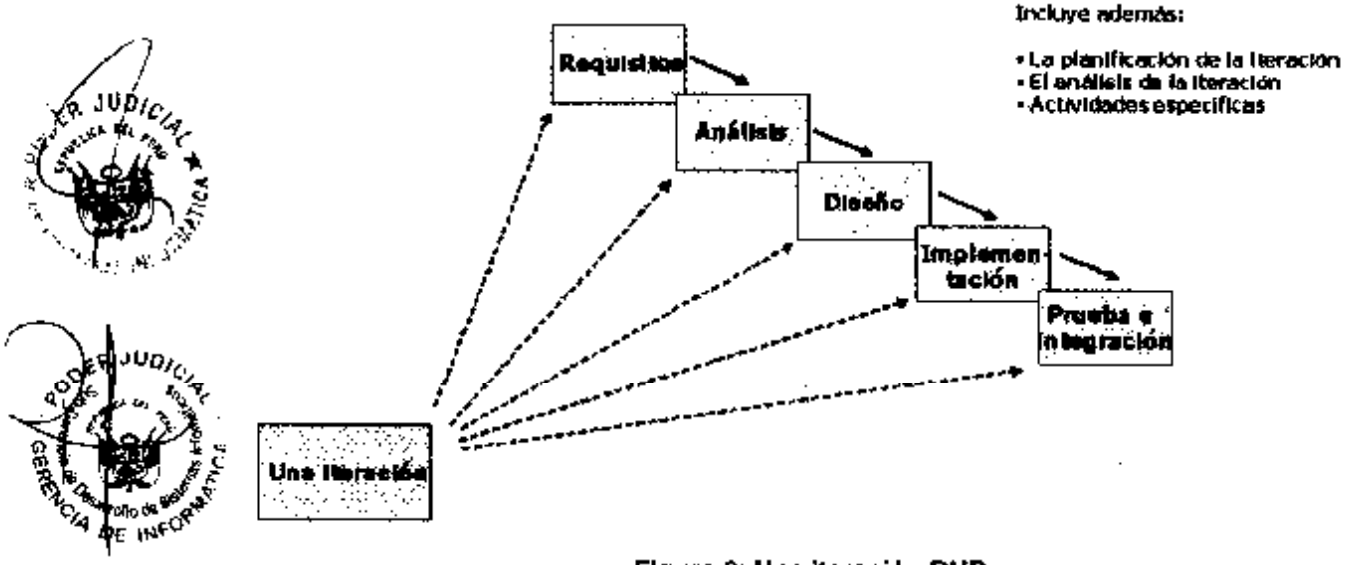

Figura 6: Una iteración RUP

El proceso iterativo e incremental consta de una secuencia de iteraciones. Cada iteración aborda una part de la funcionalidad total, pasando por todos los flujos de trabajo relevantes y refinando la arquitectura. Cad iteración se analiza cuando termina. Se puede determinar si han aparecido nuevos requisitos o ha cambiado los existentes, afectando a las iteraciones siguientes. Durante la planificación de los detalles de l siguiente iteración, el equipo también examina cómo afectarán los riesgos que aún quedan al trabajo e curso. Toda la retroalimentación de la iteración pasada permite reajustar los objetivos para las siguiente iteraciones. Se continúa con esta dinámica hasta que se haya finalizado por completo con la versión actu del producto.

RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en núme variable según el proyecto y en las que se hace un mayor o menor hincapié en los distintas actividades. En l Figura 7 se muestra cómo varía el esfuerzo asociado a las disciplinas según la fase en la que se encuentr el proyecto RUP.

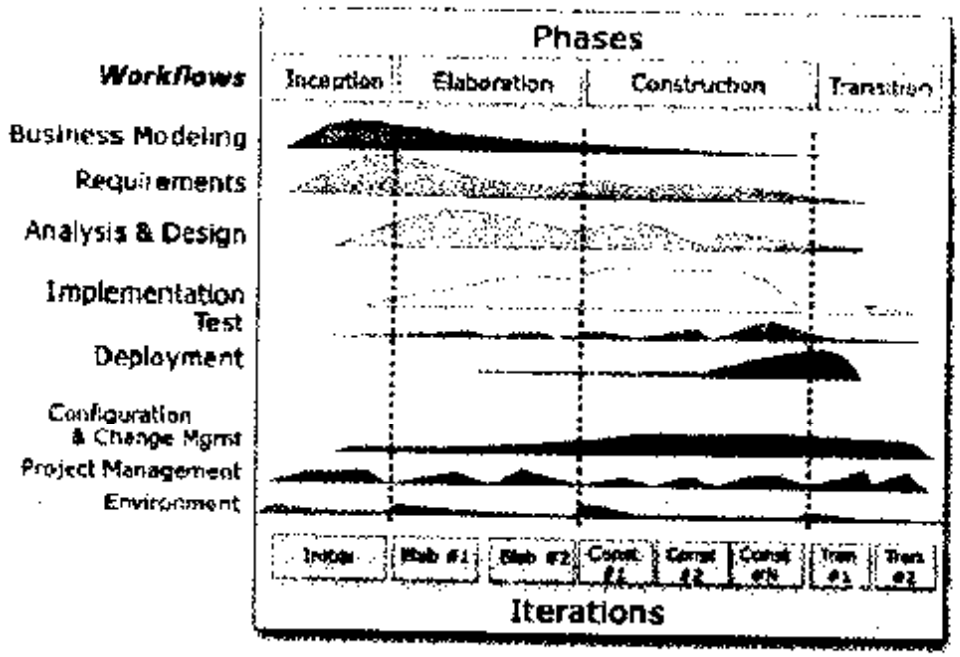

Figura 7: Esfuerzo en actividades según fase del proyecto

Las primeras iteraciones (en las fases de Inicio y Elaboración) se enfocan hacia la comprensión de problema y la tecnología, la delimitación del ámbito del proyecto, la eliminación de los riesgos críticos, y a establecimiento de una *baseline* de la arquitectura.

Durante la fase de Inicio las iteraciones hacen ponen mayor énfasis en actividades modelado del negocio y de requisitos.

En la fase de elaboración, las iteraciones se orientan al desarrollo de la *baseline* de la arquitectura, abarcan más los flujos de trabajo de requerimientos, modelo de negocios (refinamiento), análisis, diseño y una parte de implementación orientado a la *baseline* de la arquitectura.

En la fase de construcción, se lleva a cabo la construcción del producto por medio de una serie de iteraciones.

Para cada iteración se selecciona algunos Casos de Uso, se refina su análisis y diseño y se procede a su implementación y pruebas. Se realiza una pequeña cascada para cada ciclo. Se realizan tantas iteraciones hasta que se termine la implementación de la nueva versión del producto.

En la fase de transición se pretende garantizar que se tiene un producto preparado para su entrega a la comunidad de usuarios.

Como se puede observar en cada fase participan todas las disciplinas, pero que dependiendo de la fase el esfuerzo dedicado a una disciplina varía.

2.4 Otras prácticas

RUP identifica 6 *best practices* con las que define una forma efectiva de trabajar para los equipos de desarrollo de software.

Gestión de requisitos

RUP brinda una guía para encontrar, organizar, documentar, y seguir los cambios de los requisitos funcionales y restricciones. Utiliza una notación de Caso de Uso y escenarios para representar los requisitos.

Desarrollo de software iterativo

Desarrollo del producto mediante iteraciones con hitos bien definidos, en las cuales se repiten las actividades pero con distinto énfasis, según la fase del proyecto.

Desarrollo basado en componentes

La creación de sistemas intensivos en software requiere dividir el sistema en componentes con interface bien definidas, que posteriormente serán ensamblados para generar el sistema. Esta característica en u proceso de desarrollo permite que el sistema se vaya creando a medida que se obtienen o se desarrolla sus componentes.

Modelado visual (usando UML)

UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistem software. Es un estándar de la OMG (http://www.omg.org). Utilizar herramientas de modelado visual facilit la gestión de dichos modelos, permitiendo ocultar o exponer detalles cuando sea necesario. El modelad visual también ayuda a mantener la consistencia entre los artefactos del sistema: requisitos, diseños implementaciones. En resumen, el modelado visual ayuda a mejorar la capacidad del equipo para gestiona la complejidad del software.

Verificación continua de la calidad

Es importante que la calidad de todos los artefactos se evalúe en varios puntos durante el proceso de desarrollo, especialmente al final de cada iteración. En esta verificación las pruebas juegan un papel fundamental y se integran a lo largo de todo el proceso. Para todos los artefactos no ejecutables las revisiones e inspecciones también deben ser continuas.

Gestión de los cambios

El cambio es un factor de riesgo crítico en los proyectos de software. Los artefactos software cambian n sólo debido a acciones de mantenimiento posteriores a la entrega del producto, sino que durante el proces de desarrollo, especialmente importantes por su posible impacto son los cambios en los requisitos. Por otr parte, otro gran desafío que debe abordarse es la construcción de software con la participación de múltiple desarrolladores, posiblemente distribuidos geográficamente, trabajando a la vez en una *release*, y quizás e distintas plataformas. La ausencia de disciplina rápidamente conduciría al caos. La Gestión de Cambios y d Configuración es la disciplina de RUP encargada de este aspecto.

2.5 Estructura del proceso

El proceso puede ser descrito en dos dimensiones o ejes [5]:

Eje horizontal: Representa el tiempo y es considerado el eje de los aspectos dinámicos del proceso. Indic las características del ciclo de vida del proceso expresado en términos de fases, iteraciones e hitos. S puede observar en la Figura 8 que RUP consta de cuatro fases: Inicio, Elaboración, Construcción Transición. Como se mencionó anteriormente cada fase se subdivide a la vez en iteraciones.

Eje vertical: Representa los aspectos estáticos del proceso. Describe el proceso en términos d componentes de proceso, disciplinas, flujos de trabajo, actividades, artefactos y roles.

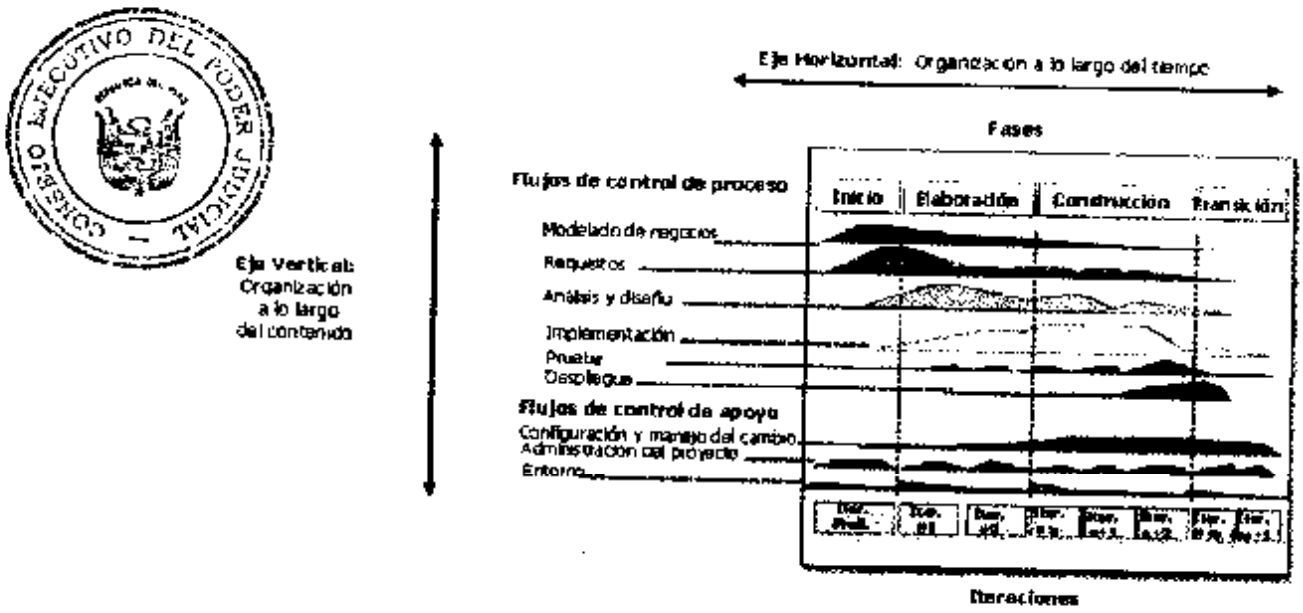

Figura 8: Estructura de RUP

2.5.1. Estructura Dinámica del proceso. Fases e iteraciones

RUP se repite a lo largo de una serie de ciclos que constituyen la vida de un producto. Cada ciclo concluy con una generación del producto para los clientes. Cada ciclo consta de cuatro fases: Inicio, Elaboración Construcción y Transición. Cada fase se subdivide a la vez en iteraciones, el número de iteraciones en cad fase es variable.

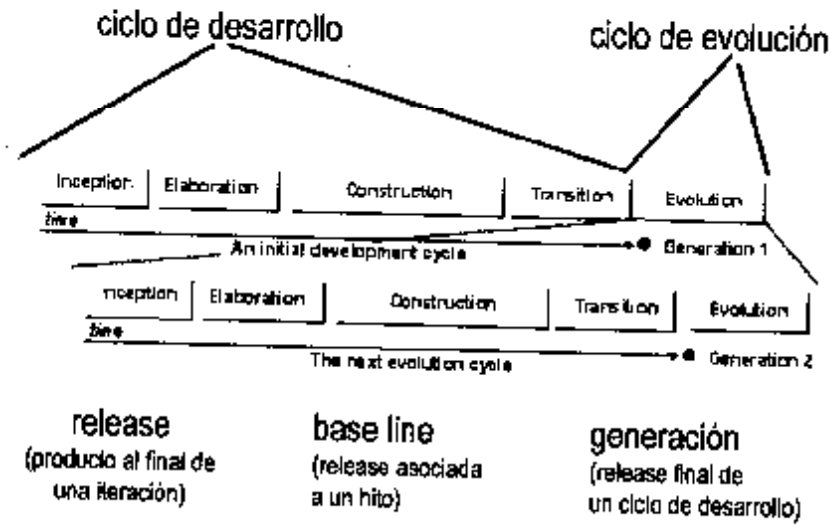

Figura 9: Ciclos, releases, baseline

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 - *Lifecycle Objectives*, Elaboración - *Lifecycle Architecture*, Construcción - *Initial Operational Capability*, Transición - Product Release. Las fases y sus respectivos hitos se ilustran en la Figura 10.

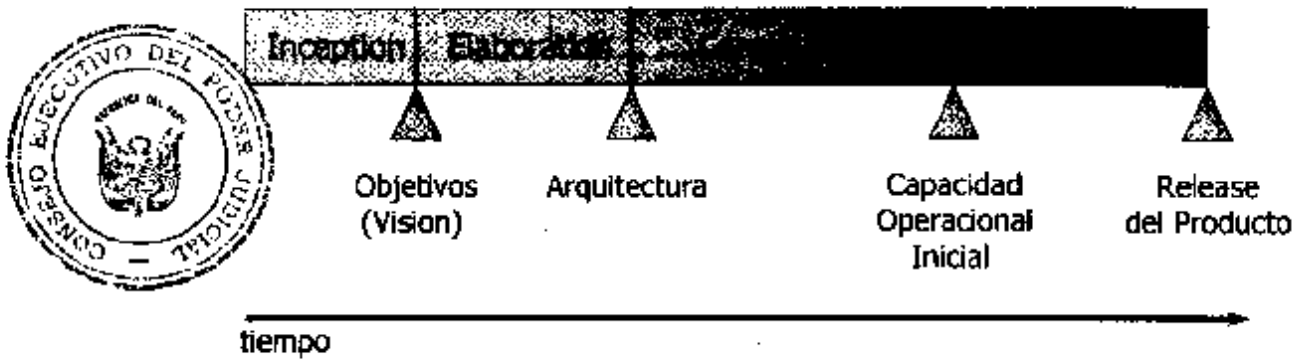

Figura 10: Fases e hitos en RUP

La duración y esfuerzo dedicado en cada fase es variable dependiendo de las características del proyecto Sin embargo, la Figura 11 ilustra porcentajes frecuentes al respecto. Consecuente con el esfuerzo señalado la Figura 12 ilustra una distribución típica de recursos humanos necesarios a lo largo del proyecto.

	Inicio	Elaboración	Construcción	Transición
Esfuerzo	5 %	20 %	65 %	10%
Tiempo Dedicado	10 %	30 %	50 %	10%

Figura 11: Distribución típicas de esfuerzo y tiempo

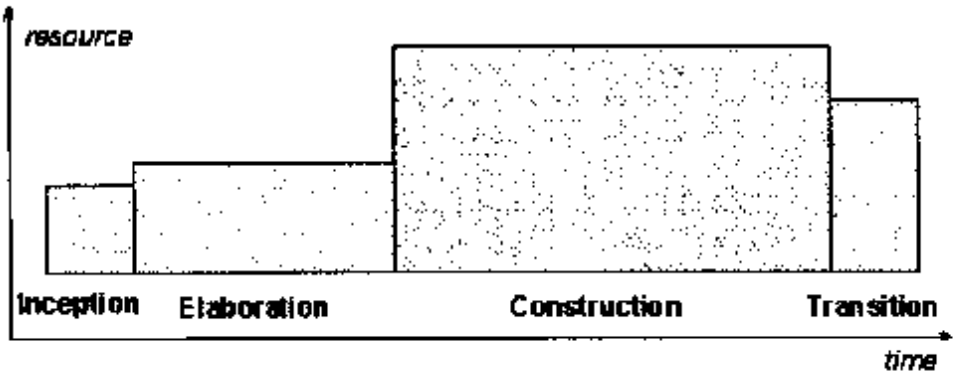

Figura 12: Distribución típica de recursos humanos

Inicio

Durante la fase de inicio se define el modelo del negocio y el alcance del proyecto. Se identifican todos lo actores y Casos de Uso, y se diseñan los Casos de Uso más esenciales (aproximadamente el 20% d modelo completo). Se desarrolla, un plan de negocio para determinar que recursos deben ser asignados proyecto.

Los objetivos de esta fase son [2]:

- Establecer el ámbito del proyecto y sus límites.
- Encontrar los Casos de Uso críticos del sistema, los escenarios básicos que definen funcionalidad.
- Mostrar al menos una arquitectura candidata para los escenarios principales.
- Estimar el coste en recursos y tiempo de todo el proyecto.

Estimar los riesgos, las fuentes de incertidumbre.

Los resultados de la fase de inicio deben ser [5]:

- Un documento de visión: Una visión general de los requerimientos del proyecto, características cla y restricciones principales.
- Modelo inicial de Casos de Uso (10-20% completado).
- Un glosario inicial: Terminología clave del dominio.
- El caso de negocio.
- Lista de riesgos y plan de contingencia.
- Plan del proyecto, mostrando fases e iteraciones.
- Modelo de negocio, si es necesario
- Prototipos exploratorios para probar conceptos o la arquitectura candidata.

Al terminar la fase de inicio se deben comprobar los criterios de evaluación para continuar:

- Todos los interesados en el proyecto coinciden en la definición del ámbito del sistema y la estimaciones de agenda.
- Entendimiento de los requisitos, como evidencia de la fidelidad de los Casos de Uso principales.
- Las estimaciones de tiempo, coste y riesgo son creíbles.
- Comprensión total de cualquier prototipo de la arquitectura desarrollado.
- Los gastos hasta el momento se asemejan a los planeados.

Si el proyecto no pasa estos criterios hay que plantearse abandonarlo o repensarlo profundamente.

Elaboración

El propósito de la fase de elaboración es analizar el dominio del problema, establecer los cimientos de l arquitectura, desarrollar el plan del proyecto y eliminar los mayores riesgos.

En esta fase se construye un prototipo de la arquitectura, que debe evolucionar en iteraciones sucesiva hasta convertirse en el sistema final. Este prototipo debe contener los Casos de Uso críticos identificados e la fase de inicio. También debe demostrarse que se han evitado los riesgos más graves.

Los objetivos de esta fase son [2]:

- Definir, validar y cimentar la arquitectura.
- Completar la visión.
- Crear un plan fiable para la fase de construcción. Este plan puede evolucionar en sucesiva iteraciones. Debe incluir los costes si procede.
- Demostrar que la arquitectura propuesta soportará la visión con un coste razonable y en un tiempo razonable.

Al terminar deben obtenerse los siguientes resultados [5]:

- Un modelo de Casos de Uso completa al menos hasta el 80%: todos los casos y actores identificados, la mayoría de los casos desarrollados.
- Requisitos adicionales que capturan los requisitos no funcionales y cualquier requisito no asociado con un Caso de Uso específico.
- Descripción de la arquitectura software.
- Un prototipo ejecutable de la arquitectura.
- Lista de riesgos y caso de negocio revisados.
- Plan de desarrollo para el proyecto.
- Un caso de desarrollo actualizado que especifica el proceso a seguir.
- Un manual de usuario preliminar (opcional).

En esta fase se debe tratar de abarcar todo el proyecto con la profundidad mínima. Sólo se profundiza en los puntos críticos de la arquitectura o riesgos importantes.

En la fase de elaboración se actualizan todos los productos de la fase de inicio.

Los criterios de evaluación de esta fase son los siguientes:

- La visión del producto es estable.
- La arquitectura es estable.
- Se ha demostrado mediante la ejecución del prototipo que los principales elementos de riesgo han sido abordados y resueltos.
- El plan para la fase de construcción es detallado y preciso. Las estimaciones son creíbles.
- Todos los interesados coinciden en que la visión actual será alcanzada si se siguen los planes actuales en el contexto de la arquitectura actual.

- Los gastos hasta ahora son aceptables, comparados con los previstos.

Si no se superan los criterios de evaluación quizá sea necesario abandonar el proyecto o replanteárselo considerablemente.

Construcción

La finalidad principal de esta fase es alcanzar la capacidad operacional del producto de forma incremental a través de las sucesivas iteraciones. Durante esta fase todos los componentes, características y requisitos deben ser implementados, integrados y probados en su totalidad, obteniendo una versión aceptable del producto.

Los objetivos concretos según [2] incluyen:

- Minimizar los costes de desarrollo mediante la optimización de recursos y evitando el tener que rehacer un trabajo o incluso desecharlo.
- Conseguir una calidad adecuada tan rápido como sea práctico.
- Conseguir versiones funcionales (alfa, beta, y otras versiones de prueba) tan rápido como sea práctico.

Los resultados de la fase de construcción deben ser [5]:

- Modelos Completos (Casos de Uso, Análisis, Diseño, Despliegue e Implementación)
- Arquitectura íntegra (mantenida y mínimamente actualizada)
- Riesgos Presentados Mitigados
- Plan del Proyecto para la fase de Transición.
- Manual Inicial de Usuario (con suficiente detalle)
- Prototipo Operacional – beta
- Caso del Negocio Actualizado

Los criterios de evaluación de esta fase son los siguientes:

- El producto es estable y maduro como para ser entregado a la comunidad de usuario para ser probado.
- Todos los usuarios expertos están listos para la transición en la comunidad de usuarios.
- Son aceptables los gastos actuales versus los gastos planeados.

Transición

La finalidad de la fase de transición es poner el producto en manos de los usuarios finales, para lo que se requiere desarrollar nuevas versiones actualizadas del producto, completar la documentación, entrenar a usuario en el manejo del producto, y en general tareas relacionadas con el ajuste, configuración, instalación y facilidad de uso del producto.

En [2] se citan algunas de las cosas que puede incluir esta fase:

- Prueba de la versión Beta para validar el nuevo sistema frente a las expectativas de los usuarios
- Funcionamiento paralelo con los sistemas legados que están siendo sustituidos por nuestro proyecto.
- Conversión de las bases de datos operacionales.
- Entrenamiento de los usuarios y técnicos de mantenimiento.
- Traspaso del producto a los equipos de marketing, distribución y venta.

Los principales objetivos de esta fase son:

- Conseguir que el usuario se valga por si mismo.
- Un producto final que cumpla los requisitos esperados, que funcione y satisfaga suficientemente a usuario.

Los resultados de la fase de transición son [5]:

- Prototipo Operacional
- Documentos Legales
- Caso del Negocio Completo
- Línea de Base del Producto completa y corregida que incluye todos los modelos del sistema
- Descripción de la Arquitectura completa y corregida
- Las iteraciones de esta fase irán dirigidas normalmente a conseguir una nueva versión.

Los criterios de evaluación de esta fase son los siguientes:

- El usuario se encuentra satisfecho.
- Son aceptables los gastos actuales versus los gastos planificados.

2.5.2. Estructura Estática del proceso. Roles, actividades, artefactos y flujos d trabajo

Un proceso de desarrollo de software define quién hace qué, cómo y cuándo. RUP define cuatro element los roles, que responden a la pregunta ¿Quién?, las actividades que responden a la pregunta ¿Cómo?, l productos, que responden a la pregunta ¿Qué? y los flujos de trabajo de las disciplinas que responde a pregunta ¿Cuándo? (ver Figura 13 y 14) [5].

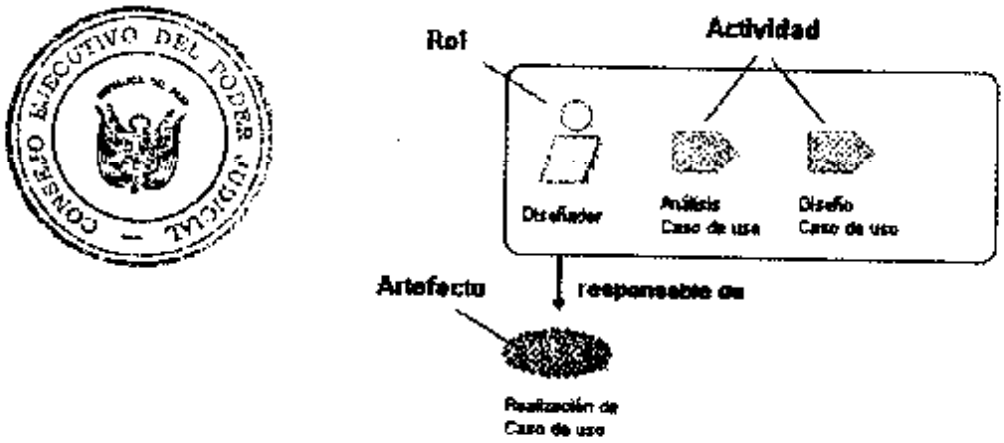

Figura 13: Relación entre roles, actividades, artefactos

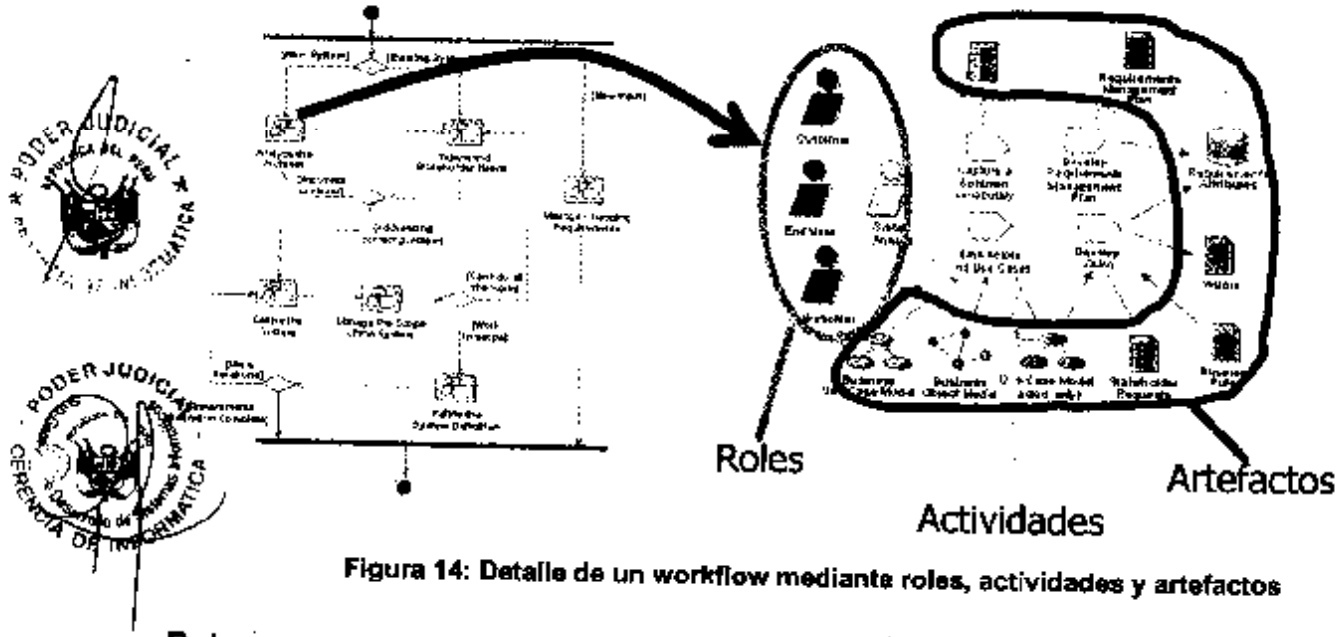

Figura 14: Detalle de un workflow mediante roles, actividades y artefactos

Roles

Un rol define el comportamiento y responsabilidades de un individuo, o de un grupo de individuos trabajando juntos como un equipo. Una persona puede desempeñar diversos roles, así como un mismo rol puede ser representado por varias personas.

Las responsabilidades de un rol son tanto el llevar a cabo un conjunto de actividades como el ser el dueño de un conjunto de artefactos [MMA].

RUP define grupos de roles, agrupados por participación en actividades relacionadas. Estos grupos son [4]:

Analistas:

- Analista de procesos de negocio.
- Diseñador del negocio.
- Analista de sistema.

- Especificador de requisitos.

Desarrolladores:
- Arquitecto de software.
- Diseñador
- Diseñador de interfaz de usuario
- Diseñador de cápsulas.
- Diseñador de base de datos.
- Implementador.
- Integrador.

Gestores:
- Jefe de proyecto
- Jefe de control de cambios.
- Jefe de configuración.
- Jefe de pruebas
- Jefe de despliegue
- Ingeniero de procesos
- Revisor de gestión del proyecto
- Gestor de pruebas.

Apoyo:
- Documentador técnico
- Administrador de sistema
- Especialista en herramientas
- Desarrollador de cursos
- Artista gráfico

Especialista en pruebas:
- Especialista en Pruebas (*tester*)
- Analista de pruebas
- Diseñador de pruebas

Otros roles:
- *Stakeholders.*
- Revisor
- Coordinación de revisiones
- Revisor técnico
- Cualquier rol

Actividades

Una actividad en concreto es una unidad de trabajo que una persona que desempeña un rol puede ser solicitado a que realice. Las actividades tienen un objetivo concreto, normalmente expresado en términos de crear o actualizar algún producto.

Artefactos

Un producto o artefacto es un trozo de información que es producido, modificado o usado durante el proceso de desarrollo de software. Los productos son los resultados tangibles del proyecto, las cosas que va creando y usando hasta obtener el producto final [MMA].

Un artefacto puede ser cualquiera de los siguientes [4]:

- Un documento, como el documento de la arquitectura del software.
- Un modelo, como el modelo de Casos de Uso o el modelo de diseño.
- Un elemento del modelo, un elemento que pertenece a un modelo como una clase, un Caso de Uso o un subsistema.

Flujos de trabajo

Con la enumeración de roles, actividades y artefactos no se define un proceso, necesitamos contar con una secuencia de actividades realizadas por los diferentes roles, así como la relación entre los mismos. Un flujo de trabajo es una relación de actividades que nos producen unos resultados observables. A continuación se dará una explicación de cada flujo de trabajo.

Modelado del negocio

Con este flujo de trabajo pretendemos llegar a un mejor entendimiento de la organización donde se va implantar el producto.

Los objetivos del modelado de negocio son [4]:

- Entender la estructura y la dinámica de la organización para la cual el sistema va ser desarrollad (organización objetivo).
- Entender el problema actual en la organización objetivo e identificar potenciales mejoras.
- Asegurar que clientes, usuarios finales y desarrolladores tengan un entendimiento común de l organización objetivo.
- Derivar los requisitos del sistema necesarios para apoyar a la organización objetivo.

Para lograr estos objetivos, el modelo de negocio describe como desarrollar una visión de la nuev organización, basado en esta visión se definen procesos, roles y responsabilidades de la organización po medio de un modelo de Casos de Uso del negocio y un Modelo de Objetos del Negocio. Complementario estos modelos, se desarrollan otras especificaciones tales como un Glosario.

Requisitos

Este es uno de los flujos de trabajo más importantes, porque en él se establece qué tiene que hace exactamente el sistema que construyamos. En esta línea los requisitos son el contrato que se debe cumpli de modo que los usuarios finales tienen que comprender y aceptar los requisitos que especifiquemos.

Los objetivos del flujo de datos Requisitos es [4]:

- Establecer y mantener un acuerdo entre clientes y otros *stakeholders* sobre lo que el sistema podrí hacer.
- Proveer a los desarrolladores un mejor entendimiento de los requisitos del sistema.
- Definir el ámbito del sistema.
- Proveer una base para la planeación de los contenidos técnicos de las iteraciones.
- Proveer una base para estimar costos y tiempo de desarrollo del sistema.
- Definir una interfaz de usuarios para el sistema, enfocada a las necesidades y metas del usuario.

Los requisitos se dividen en dos grupos. Los requisitos funcionales representan la funcionalidad del sistema Se modelan mediante diagramas de Casos de Uso. Los requisitos no funcionales representan aquello atributos que debe exhibir el sistema, pero que no son una funcionalidad específica. Por ejemplo requisito de facilidad de uso, fiabilidad, eficiencia, portabilidad, etc.

Para capturar los requisitos es preciso entrevistar a todos los interesados en el proyecto, no sólo a los usuarios finales, y anotar todas sus peticiones. A partir de ellas hay que descubrir lo que necesitan y expresarlo en forma de requisitos.

En este flujo de trabajo, y como parte de los requisitos de facilidad de uso, se diseña la interfaz gráfica de usuario. Para ello habitualmente se construyen prototipos de la interfaz gráfica de usuario que se contrastar con el usuario final.

Análisis y Diseño

El objetivo de este flujo de trabajo es traducir los requisitos a una especificación que describe cómo implementar el sistema.

Los objetivos del análisis y diseño son [4]:
Transformar los requisitos al diseño del futuro sistema.
Desarrollar una arquitectura para el sistema.
Adaptar el diseño para que sea consistente con el entorno de implementación, diseñando para el rendimiento.

El análisis consiste en obtener una visión del sistema que se preocupa de ver qué hace, de modo que sólo se interesa por los requisitos funcionales. Por otro lado el diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, en definitiva cómo cumple el sistema sus objetivos.

Al principio de la fase de elaboración hay que definir una arquitectura candidata: crear un esquema inicial de la arquitectura del sistema, identificar clases de análisis y actualizar las realizaciones de los Casos de Uso con las interacciones de las clases de análisis. Durante la fase de elaboración se va refinando esta

arquitectura hasta llegar a su forma definitiva. En cada iteración hay que analizar el comportamiento para diseñar componentes. Además si el sistema usará una base de datos, habrá que diseñarla también obteniendo un modelo de datos.

El resultado final más importante de este flujo de trabajo será el modelo de diseño. Consiste en colaboraciones de clases, que pueden ser agregadas en paquetes y subsistemas.

Otro producto importante de este flujo es la documentación de la arquitectura de software, que captura varias vistas arquitectónicas del sistema.

Implementación

En este flujo de trabajo se implementan las clases y objetos en ficheros fuente, binarios, ejecutables y demás. Además se deben hacer las pruebas de unidad: cada implementador es responsable de probar las unidades que produzca. El resultado final de este flujo de trabajo es un sistema ejecutable.

En cada iteración habrá que hacer lo siguiente:

- Planificar qué subsistemas deben ser implementados y en que orden deben ser integrados, formando el Plan de Integración.
- Cada implementador decide en que orden implementa los elementos del subsistema.
- Si encuentra errores de diseño, los notifica.
- Se prueban los subsistemas individualmente.
- Se integra el sistema siguiendo el plan.

La estructura de todos los elementos implementados forma el modelo de implementación. La integración debe ser incremental, es decir, en cada momento sólo se añade un elemento. De este modo es más fácil localizar fallos y los componentes se prueban más a fondo. En fases tempranas del proceso se pueden implementar prototipos para reducir el riesgo. Su utilidad puede ir desde ver si el sistema es viable desde el principio, probar tecnologías o diseñar la interfaz de usuario. Los prototipos pueden ser exploratorios (desechables) o evolutivos. Estos últimos llegan a transformarse en el sistema final.

Pruebas

Este flujo de trabajo es el encargado de evaluar la calidad del producto que estamos desarrollando, pero no para aceptar o rechazar el producto al final del proceso de desarrollo, sino que debe ir integrado en todo el ciclo de vida.

Esta disciplina brinda soporte a las otras disciplinas. Sus objetivos son [4]:

- Encontrar y documentar defectos en la calidad del software.
- Generalmente asesora sobre la calidad del software percibida.
- Provee la validación de los supuestos realizados en el diseño y especificación de requisitos por medio de demostraciones concretas.
- Verificar las funciones del producto de software según lo diseñado.
- Verificar que los requisitos tengan su apropiada implementación.

Las actividades de este flujo comienzan pronto en el proyecto con el plan de prueba (el cual contiene información sobre los objetivos generales y específicos de las prueba en el proyecto, así como las estrategias y recursos con que se dotará a esta tarea), o incluso antes con alguna evaluación durante la fase de inicio, y continuará durante todo el proyecto.

El desarrollo del flujo de trabajo consistirá en planificar que es lo que hay que probar, diseñar cómo se va a hacer, implementar lo necesario para llevarlos a cabo, ejecutarlos en los niveles necesarios y obtener los resultados, de forma que la información obtenida nos sirva para ir refinando el producto a desarrollar.

Despliegue

El objetivo de este flujo de trabajo es producir con éxito distribuciones del producto y distribuirlo a los usuarios. Las actividades implicadas incluyen:

- Probar el producto en su entorno de ejecución final.
- Empaquetar el software para su distribución.
- Distribuir el software.
- Instalar el software.
- Proveer asistencia y ayuda a los usuarios.
- Formar a los usuarios y al cuerpo de ventas.
- Migrar el software existente o convertir bases de datos

Este flujo de trabajo se desarrolla con mayor intensidad en la fase de transición, ya que el propósito del flu es asegurar una aceptación y adaptación sin complicaciones del software por parte de los usuarios. S ejecución inicia en fases anteriores, para preparar el camino, sobre todo con actividades de planificación, la elaboración del manual de usuario y tutoriales.

Gestión del proyecto

La Gestión del proyecto es el arte de lograr un balance al gestionar objetivos, riesgos y restricciones pa desarrollar un producto que sea acorde a los requisitos de los clientes y los usuarios.

Los objetivos de este flujo de trabajo son:

- Proveer un marco de trabajo para la gestión de proyectos de software intensivos.
- Proveer guías prácticas realizar planeación, contratar personal, ejecutar y monitorear el proyecto.
- Proveer un marco de trabajo para gestionar riesgos.

La planeación de un proyecto posee dos niveles de abstracción: un plan para las fases y un plan para cad iteración.

Configuración y control de cambios

La finalidad de este flujo de trabajo es mantener la integridad de todos los artefactos que se crean en proceso, así como de mantener información del proceso evolutivo que han seguido.

Entorno

La finalidad de este flujo de trabajo es dar soporte al proyecto con las adecuadas herramientas, procesos métodos. Brinda una especificación de las herramientas que se van a necesitar en cada momento, así com definir la instancia concreta del proceso que se va a seguir.

En concreto las responsabilidades de este flujo de trabajo incluyen:

- Selección y adquisición de herramientas
- Establecer y configurar las herramientas para que se ajusten a la organización.
- Configuración del proceso.
- Mejora del proceso.
- Servicios técnicos.

El principal artefacto que se usa en este flujo de trabajo es el *caso de desarrollo* que especifica para e proyecto actual en concreto, como se aplicará el proceso, que productos se van a utilizar y como van a se utilizados. Además se tendrán que definir las guías para los distintos aspectos del proceso, como puede ser el modelado del negocio y los Casos de Uso, para la interfaz de usuario, el diseño, la programación, e manual de usuario.

Referencias

[1] Jacaboson, I., Booch, G., Rumbaugh J., El Proceso Unificado de Desarrollo de Software, 2000 Addison Wesley

[2] Kruchten, P., The Rational Unified Process: An Introduction, 2000 Addison Wesley

[3] Kruchten, P. Architectural Blueprints—The "4+1" View Model of Software Architecture. IEEE Software 12 (6), November 1995, pp. 42-50.

[4] Rational Software Corporation, Product: Rational Software Corporation, 2002

[5] Rational Software Corporation, Rational Unified Process. Best Practices for Software Development Teams, 1998