
5


Proceso
Unificado
Rational
Aplicado

5. Proceso Unificado Rational Aplicado

Rational Unified Process (RUP) es una metodología de desarrollo de software orientado a objeto que establece las bases, plantillas, y ejemplos para todos los aspectos y fases de desarrollo del software. RUP es herramientas de la ingeniería de software que combinan los aspectos del proceso de desarrollo (como fases definidas, técnicas, y prácticas) con otros componentes de desarrollo (como documentos, modelos, manuales, código fuente, etc.) dentro de un framework unificado. RUP establece cuatro fases de desarrollo cada una de las cuales esta organizada en varias iteraciones separadas que deben satisfacer criterios definidos antes de emprender la próxima fase, (Figura 5.1).

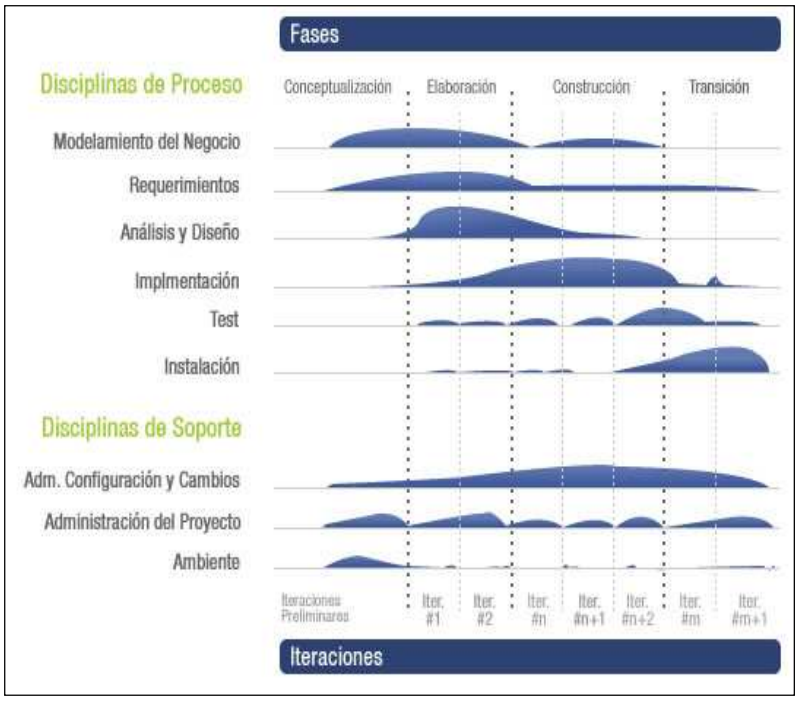

Figura 5.1. Estructura del RUP mostrada en dos dimensiones

5.1 Historia

La Figura 5.2 muestra la historia de RUP. El antecedente más importante se ubica en 1967 con la Metodología Ericsson (Ericsson Approach) elaborada por Ivar Jacobson, una aproximación de desarrollo basada en componentes, que introdujo el concepto de Caso de Uso. Entre los años de 1987 a 1995 Jacobson fundó la

compañía Objectory AB y lanza el proceso de desarrollo Objectory (abreviación de Object Factory).

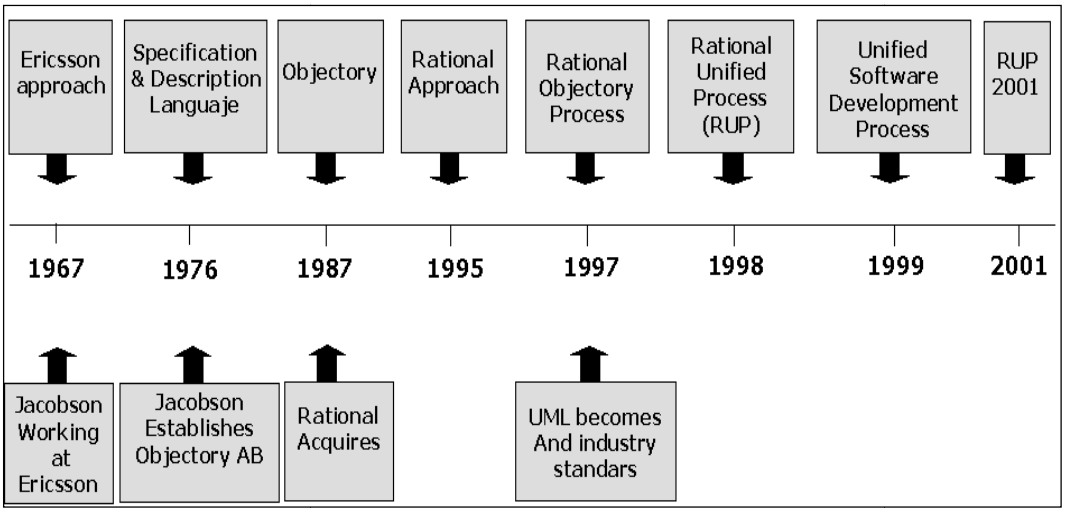

Figura 5.2. Historia de RUP

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 (Rational 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, destacándose especialmente el flujo de trabajo conocido como modelado del negocio. En junio del 1998 se lanza Rational Unified Process.

5.2 Características de la metodología

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

5.2.1 Proceso dirigido a casos de uso

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. No solo inician como se muestra en la Figura 5.3.

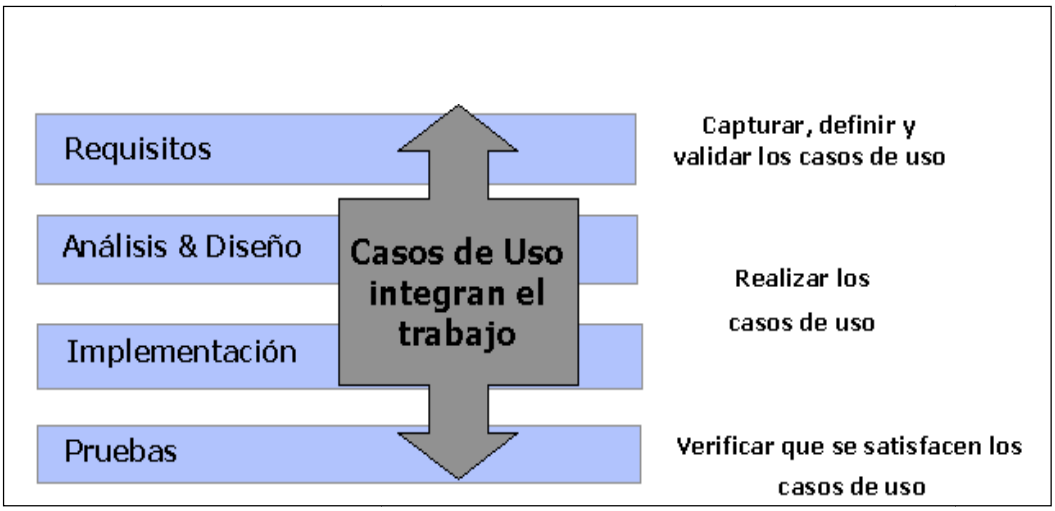

Figura 5.3: Los Casos de Uso integran el trabajo

Los Casos de Uso no sólo inician el proceso de desarrollo sino establecen la transacción entre los distintos artefactos que son generados en las diferentes actividades del proceso de desarrollo. Basándose en los Casos de Uso se crean los modelos de análisis y diseño, posteriormente se genera la implementación que los lleva a cabo, ayuda a verificar la adecuada implementación de cada caso de uso en el producto final.

5.2.2 Proceso centrado en la arquitectura

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

La arquitectura involucra los aspectos estáticos y dinámicos más significativos del sistema, está relacionada con la toma de decisiones que indican la forma de construcción del sistema. La arquitectura se ve influenciada por la plataforma software, sistema operativo, gestor de bases de datos, protocolos, consideraciones de desarrollo como sistemas heredados. Muchas de estas restricciones constituyen requisitos no funcionales del sistema.

RUP presta especial atención al establecimiento temprano de una buena arquitectura que no se vea fuertemente impactada ante cambios posteriores durante la construcción y el mantenimiento.

Existe una interacción entre los casos de uso y la arquitectura, los casos de uso deben encajar en la arquitectura cuando se llevan a cabo y la arquitectura debe permitir el desarrollo de todos los casos de uso requeridos, actualmente y en el futuro. Esto provoca que tanto arquitectura como casos de uso deban evolucionar en paralelo durante todo el proceso de desarrollo de software.

En la Figura 5.4 se muestra la evolución de la arquitectura durante las fases de RUP. Se tiene una arquitectura más robusta en las fases finales del proyecto. En las fases iniciales lo que se hace es ir consolidando la arquitectura por medio de baselines y se va modificando dependiendo de las necesidades del proyecto.

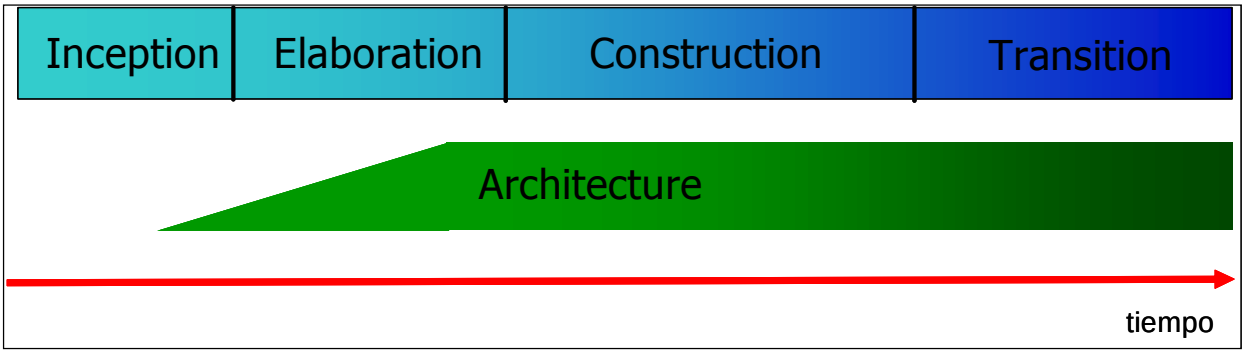

Figura 5.4. Evolución de la arquitectura del sistema

Es conveniente ver el sistema desde diferentes perspectivas para comprender mejor el diseño por lo que la 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 la arquitectura, el cual

recibe este nombre porque lo forman las vistas lógica, de implementación, de proceso y de despliegue, más la de casos de uso que es la que da cohesión a todas.

Durante la construcción, los diversos modelos van desarrollándose hasta completarse. La descripción de la arquitectura sin embargo, no debería cambiar significativamente debido a que la mayor parte de la arquitectura se decidió durante la elaboración, incorporando pocos cambios a la arquitectura, Figura 5.5.

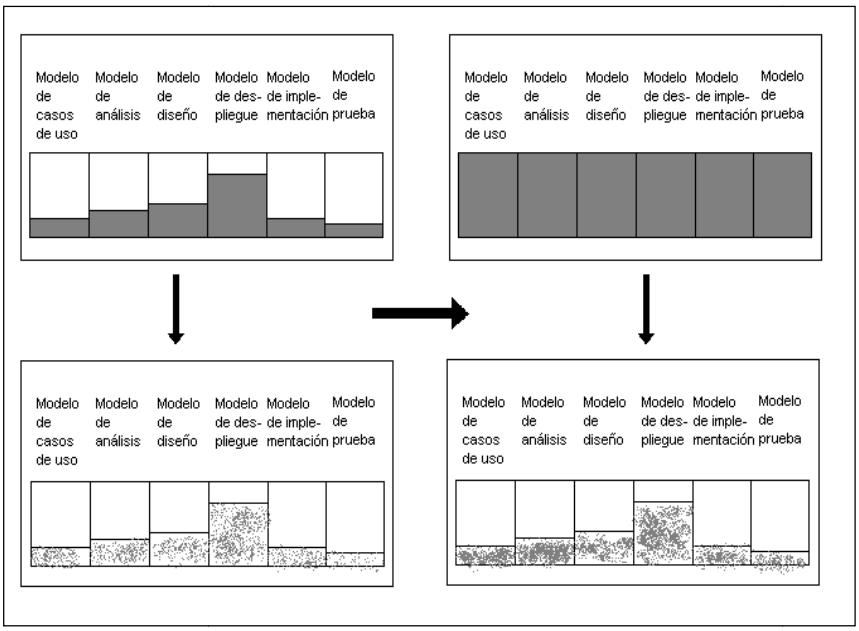

Figura 5.5. Proceso de madurez de la arquitectura

5.2.3 Proceso iterativo e incremental

RUP propone tener un proceso iterativo e incremental en donde el trabajo se divide en partes más pequeñas o mini proyectos, permitiendo generar un equilibrio entre casos de uso y arquitectura. Cada mini proyecto se puede ver como una iteración (un recorrido más o menos completo a lo largo de todos los flujos de trabajo 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 5.6. la cual pasa por los flujos fundamentales (Requisitos, Análisis, Diseño,

Implementación y Pruebas), también existe una planificación de la iteración, un análisis de la iteración y algunas actividades específicas de la iteración. Al finalizar se realiza una integración de los resultados con lo obtenido de las iteraciones anteriores.

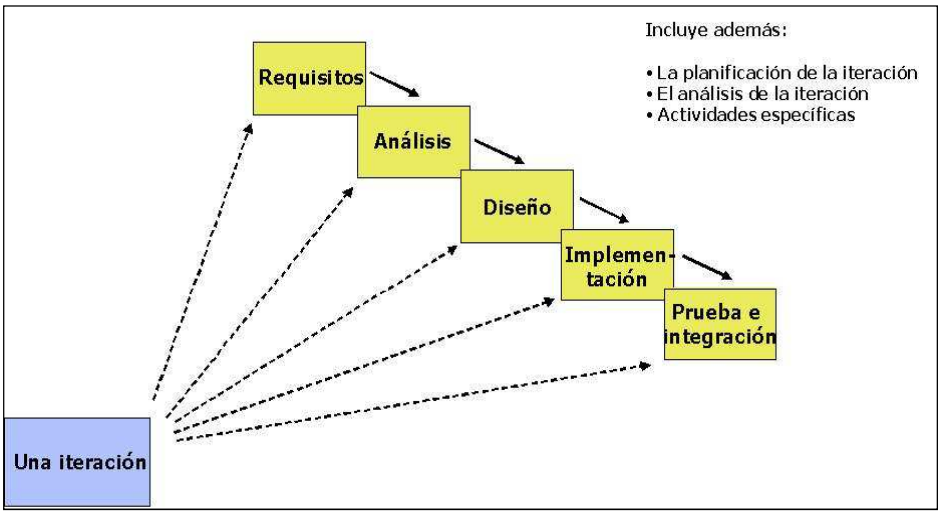

Figura 5.6. Iteración de RUP

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

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 los distintas actividades. En la Figura 5.1 se muestra cómo varía

el esfuerzo asociado a las disciplinas según la fase en la que se encuentre el proyecto RUP.

Fases de Inicio y Elaboración*:* Se enfocan hacia la comprensión del problema y la tecnología, la delimitación del ámbito del proyecto, la eliminación de los riesgos críticos, y al establecimiento de una baseline de la arquitectura.

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

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.

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.

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.

5.3 Mejoras practicas para el desarrollo de software

RUP identifica las seis mejores prácticas con las que define una forma efectiva de trabajar para los equipos de desarrollo de software (Figura 5.7).

- La administración de requerimientos

- El desarrollo iterativo
- La arquitectura basada en componentes
- El modelo visual
- La verificación continua de la calidad
- La administración del cambio

Estas seis prácticas orientan el modelo y con ellas se pretende solucionar muchos de los problemas asociados al software. Adicionalmente hay muchos aspectos de diseño que son bien conocidos, pero que en realidad han sido muy poco implementados en los proyectos de software; estos son: facilidad de uso, modularidad, encapsulamiento y facilidad de mantenimiento. Es necesario entonces definir una arquitectura sólida basada en componentes, para construir mejores y más flexibles soluciones de software para las necesidades organizacionales.

Los cambios en un proyecto no pueden ser detenidos dado que la evolución del entorno de cada organización es continua, pero sí pueden ser administrados de manera que su impacto pueda ser estimado para determinar si dicho cambio se incluye o no y si el proyecto debe ser reajustado.

Cada cambio en el proyecto debe tener especificado cuándo y cómo se va a realizar, quién lo va a hacer y qué productos se ven involucrados en ese cambio. En ese punto es donde el control de cambios y la trazabilidad de los componentes a través de los diversos modelos adquieren una gran importancia.

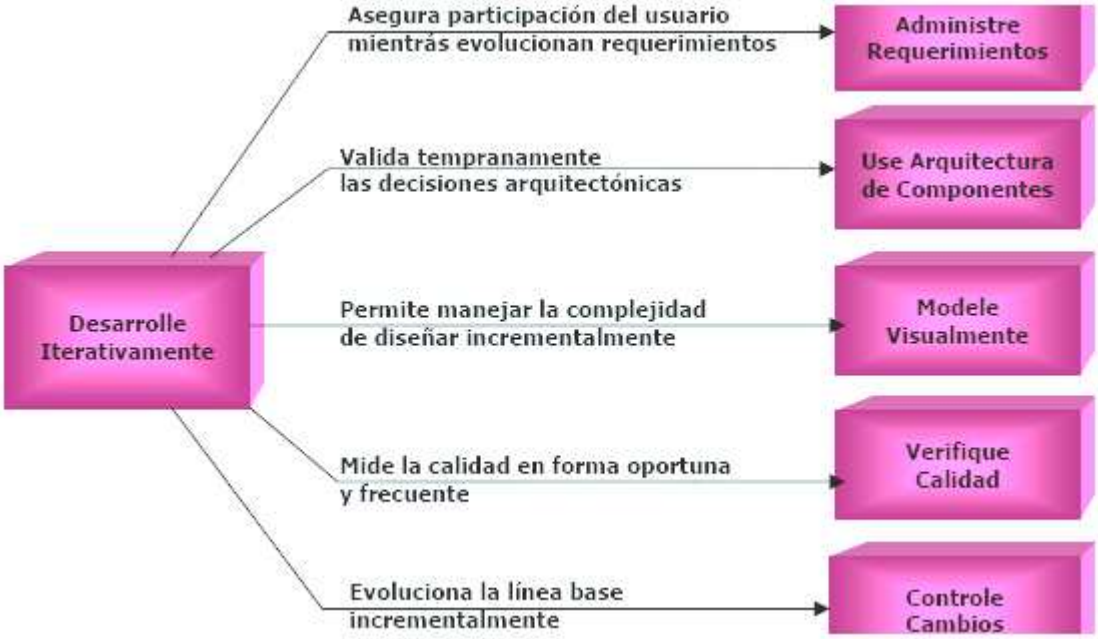

Figura 5.7. Mejores Prácticas de RUP

5.3.1 Gestión de Requisitos

RUP brinda una guía para encontrar, organizar, documentar, y seguir los cambios de los requisitos funcionales y restricciones (Figura 5.8). Utiliza una notación de Caso de Uso para representar los requisitos. Con la finalidad de especificar el comportamiento deseado del sistema (objetivos del usuario), así como describir qué debe de hacer, pero no especifica cómo lo hace y da lugar a un conjunto de posibles escenarios

La gestión de requisitos debe de asegurarse de resolver el problema correcto y construir el sistema correcto. Es necesario evaluar el impacto del cambio en cuestión a las necesidades del negocio y las características del producto (casos de uso). Todos los participantes del proyecto requieren tener acceso a los requisitos

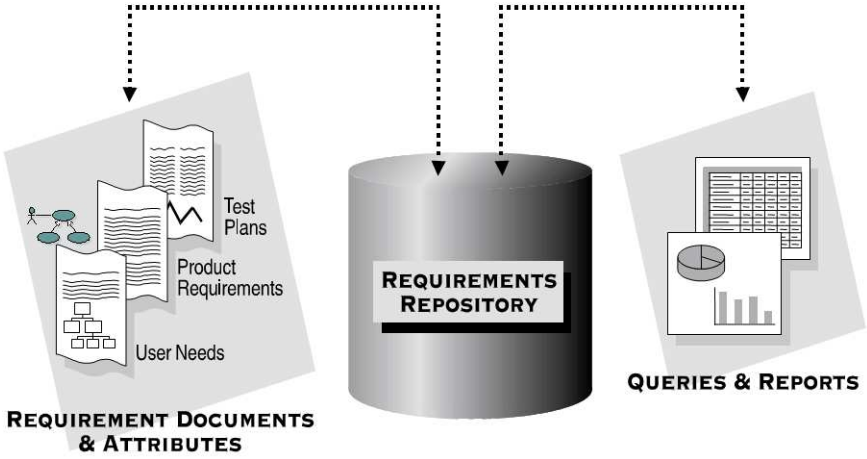

Figura 5.8 Deposito de Requerimientos.

5.3.2 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 (Figura 5.1):

- Cada iteración produce una versión ejecutable del sistema.
- Las primeras iteraciones atacan los riesgos mayores.
- Define y robustece la arquitectura de la aplicación en forma temprana.
- Cada iteración permite la retroalimentación del usuario.
- Prueba desde el principio, verificando desempeño y escalabilidad.
- Entregables bien definidos y delimitados permiten tener metas a corto plazo y no una sola meta a largo plazo.
- El progreso se mide mediante la evaluación de las implementaciones (mediciones reales).

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.

5.3.3 La arquitectura basada en componentes

La clave del éxito es crear arquitecturas duraderas, flexibles al cambio y basadas en componentes (Figura 5.9).

Reuso de todo “artefacto”:

- Arquitectura.
- Casos de Uso, Análisis, Diseño, Implementación y Pruebas
- Modelos de interfaces, modelos de negocio, patrones arquitectónicos, etc.

Reuso de tecnología

- Proceso y automatización
- Proyectos
- Guías

.

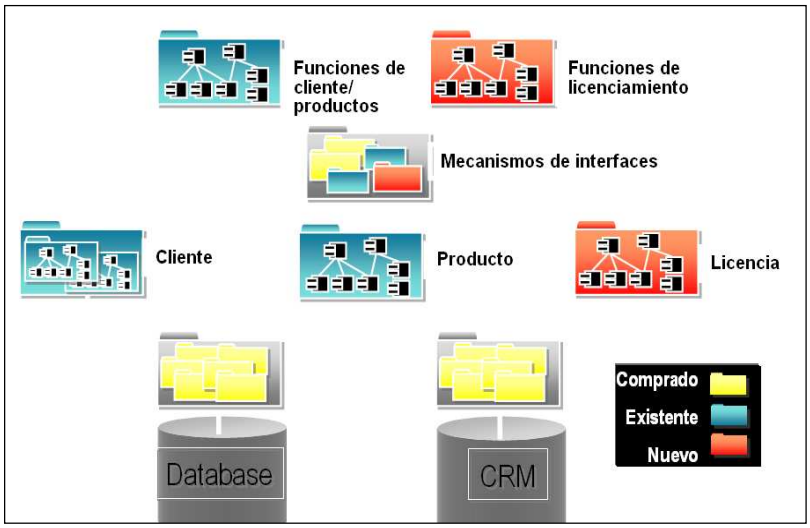

Figura 5.9. Arquitectura basada en Componentes.

5.3.4 Modelo Visual (Usando UML)

El modelado visual también ayuda a mantener la consistencia entre los artefactos del sistema: requisitos, diseños e implementaciones (Figura 5.10). El modelado visual ayuda a mejorar la capacidad del equipo para gestionar la complejidad del

software a modelar el sistema independientemente del lenguaje de implementación y promover la reutilización.

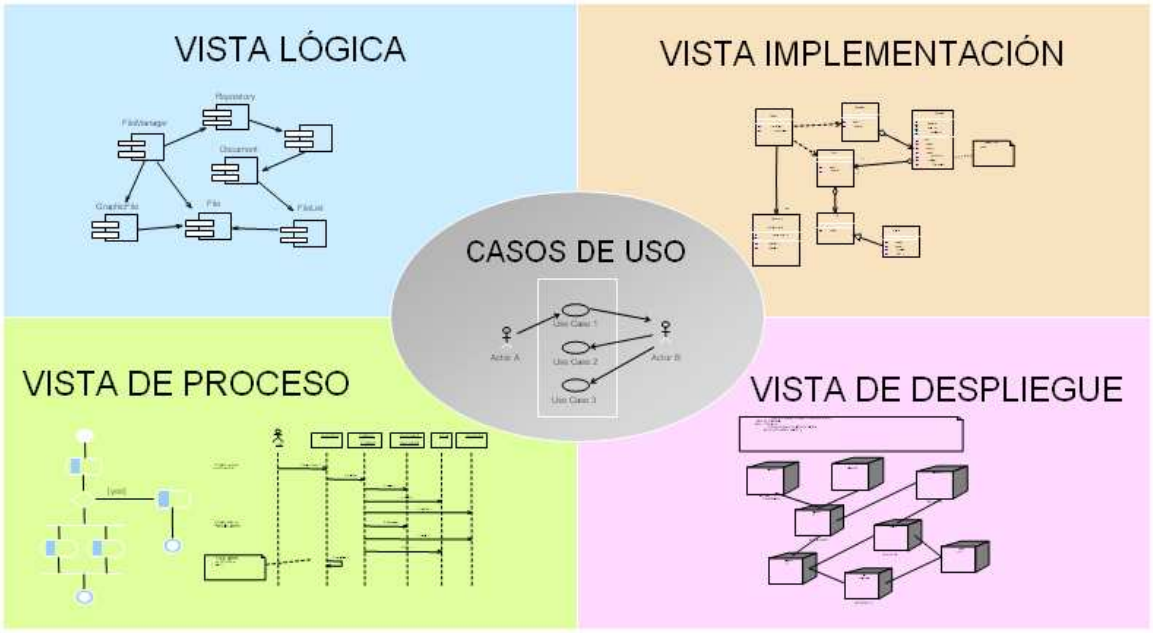

Figura 5.10 Vista de la arquitectura con UML

5.3.5 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 (es de 100 a 1000 veces más costoso encontrar y reparar los problemas del software después del desarrollo).

Se hace una evaluación objetiva del estatus del proyecto, se detecta inconsistencias entre análisis, diseño e implementación. Las pruebas se concentran en los aspectos de mayor riesgo, los defectos se identifican claramente:

- Verificar y probar todo continuamente desde el inicio.
- Toda actividad es verificada: los casos de prueba se definen a partir de: requerimientos, análisis, diseño y codificación.
- Automatizar la verificación de la calidad con herramientas.

Cada actividad incluye su verificación es decir “Todo aquello que hagas, no lo habrás terminado, hasta que hayas verificado que hiciste lo que debías de hacer“ (Figura 5.11).

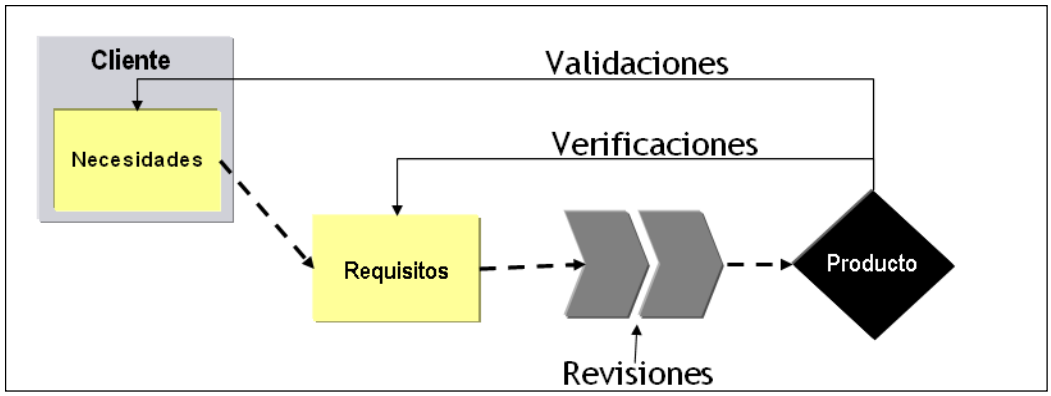

Figura 5.11 Plan de actividades de aseguramiento de la calidad.

5.3.6 Gestión de los cambios

El cambio es un factor de riesgo crítico en los proyectos de software. Los artefactos software cambian no sólo debido a acciones de mantenimiento posteriores a la entrega del producto, sino que durante el proceso de desarrollo, especialmente importantes por su posible impacto son los cambios en los requisitos. Por otra parte, otro gran desafío que debe abordarse es la construcción de software con la participación de múltiples desarrolladores, posiblemente distribuidos geográficamente, trabajando a la vez en una release, y quizás en distintas plataformas. La ausencia de disciplina rápidamente conduciría al caos.

La administración de Configuración y Cambios es:

- Permitir, controlar y monitorear cambios para habilitar un desarrollo iterativo.
- Controlar todos los artefactos de software modelos, código, documentos, etc.
- Administrar todas las versiones, con integración automática a los cambios realizados al software.
- Establecer espacios de trabajos seguros y aislados para cada desarrollador.
- Contar con métricas de estado y avance.

5.4 Estructura del proceso

El proceso puede ser descrito en *dos dimensiones o ejes*:

Eje horizontal: Representa el tiempo y es considerado el eje de los aspectos dinámicos del proceso. Indica las características del ciclo de vida del proceso expresado en términos de fases, iteraciones e hitos. Se puede observar en la Figura 5.1 que RUP consta de cuatro fases: Inicio, Elaboración, Construcción y 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 de componentes de proceso, disciplinas, flujos de trabajo, actividades, artefactos y roles.

5.4.1 Estructura Dinámica

RUP se repite a lo largo de una serie de ciclos que constituyen la vida de un producto. Cada ciclo concluye 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 cada fase es variable (Figura 5.12).

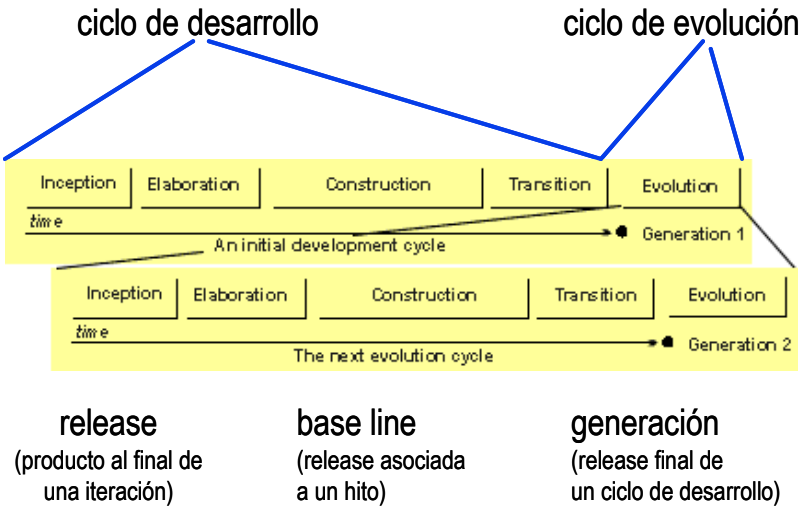

Figura 5.12 Ciclos, releases, baseline

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 - Objetivos, Elaboración - Arquitectura, Construcción - Initial Operational Capability, Transición - Product Release. Las fases y sus respectivos hitos.

La duración y esfuerzo dedicado en cada fase es variable dependiendo de las características del proyecto. Consecuente con el esfuerzo señalado, la Figura 5.13 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 5.13. Distribución típica de esfuerzo y tiempo

5.4.1.1 Fase Inicio

Durante la fase de inicio se define el modelo del negocio y el alcance del proyecto. Se identifican todos los actores y Casos de Uso (CU), y se diseñan los CU más esenciales. Se desarrolla, un plan de negocio para determinar qué recursos deben ser asignados al proyecto.

Los objetivos de esta fase son:

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

Los artefactos o resultados de esta fase deben ser:

- Un documento de visión.
- Un documento de requerimientos.
- Modelo inicial de Casos de Uso.
- Un glosario inicial
- El caso de Negocio.
- Plan del proyecto, mostrando fases e iteraciones.
- Modelo de negocio.
- 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 las estimaciones de agenda.
- Entendimiento de los requisitos, como evidencia de la fidelidad de los Casos de Uso principales.
- Las estimaciones de tiempo, costo 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.

5.4.1.1.1 Documento de visión del sistema

El propósito de este documento es identificar las expectativas y requerimientos generales de los usuarios del sistema a desarrollar, desde una perspectiva de alto nivel. Tiene la finalidad de determinar las fronteras del sistema, la identificación de prioridades y dependencias entre funcionalidades, así como identificar cualquier condición que afecte el éxito del proyecto. Este documento se puede ver en el **Apéndice A.1**

5.4.1.1.3 Documento de Requerimientos

El propósito de este documento es establecer y mantener un acuerdo con los usuarios e involucrados sobre qué es lo que deberá hacer el sistema a desarrollar. Concentra los principales requerimientos a considerar en el desarrollo del sistema. Este documento se puede ver en el **Apéndice A.2.**

5.4.1.1.4 Modelo de negocio

Proporciona una vista global de los procesos de negocio involucrados para el desarrollo del sistema mediante la representación gráfica que muestre el comportamiento de la organización de manera sencilla y comprensible, así como presentar las Áreas de Negocio involucradas y sus diferentes escenarios. Este documento se puede ver en el **Apéndice A.3.**

5.4.1.1.5 Glosario

Este documento tiene la finalidad de proporcionar la definición de los principales términos, acrónimos y abreviaturas utilizados en el sistema. Este documento se puede ver en el **Apéndice A.4**.

5.4.1.2 Fase elaboración

El propósito de la fase de elaboración es analizar el dominio del problema, establecer los cimientos de la 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 sucesivas hasta convertirse en el sistema final.

Los objetivos de esta fase son:

- Definir, validar y cimentar la arquitectura.
- Completar la visión.
- Crear un plan fiable para la fase de construcción. Este plan puede evolucionar en sucesivas iteraciones.
- Demostrar que la arquitectura propuesta soportará la visión con un costo razonable y en un tiempo razonable.

Los artefactos o resultados de esta fase deben ser:

- Un modelo de Casos de Uso completo al menos hasta el 80%:
- 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.
- Plan de desarrollo para el proyecto.
- Un manual de usuario preliminar (opcional).

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.

5.4.1.2.1 Casos de uso

Consiste en narrativas de casos de uso del sistema, indicando los flujos básicos y alternos de las funcionalidades, así como las validaciones y reglas de negocio a implementar. Estos documentos se encuentran en el **Apéndice A.5 y B.1 al B.10.**

5.4.1.2.2 Documento de Arquitectura

Este documento proporciona una visión general completa del sistema, utilizando varias vistas de arquitectura para describir diferentes aspectos del sistema. Tiene la intención de captar y transmitir las decisiones arquitectónicas críticas que se han realizado sobre el Sistema.

Una de las primeras vistas a ser consideradas es la vista de casos de uso ya que estos dirigirán el resto del proyecto, para un sistema de alto grado de concurrencia y distribución las vistas de proceso y deployment deberán ser consideradas desde el inicio ya que podrían tener un impacto sustancial en el sistema. Este documento se encuentra en el **Apéndice B.11.**

5.4.1.2.3 Prototipo

Este prototipo debe implementar los casos de uso críticos identificados en la fase de inicio. También debe demostrarse que se han evitado los riesgos más graves.

5.4.1.2.4 Plan de desarrollo

El propósito del plan de desarrollo es proporcionar la información necesaria para controlar el proyecto. En el se describe el enfoque de desarrollo del software, así como el plan global usado para el desarrollo del sistema. Este documento se encuentra en el **Apéndice B.12.**

5.4.1.3 Fase 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 de esta fase son:

- Minimizar los costos 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 posible.
- Conseguir versiones funcionales (alfa, beta, y otras versiones de prueba) tan rápido como sea posible.

Los artefactos o resultados de esta fase deben ser:

- 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.

5.4.1.3.1 Sistema versión beta

Una versión beta o lanzamiento beta representa generalmente la primera versión completa del sistema, que es probable que sea inestable pero útil para que las demostraciones internas y las inspecciones previas por parte del clientes.

5.4.1.4 Fase 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 al usuario en el manejo del producto, y en general tareas relacionadas con el ajuste, configuración, instalación y facilidad de uso del producto **Apéndice C.3**.

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 al usuario.

Los resultados de la fase de transición son:

- Prototipo Operacional.
- Documento de despliegue.
- Plan de pruebas.
- 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.

5.4.1.4.1 Documento de despliegue

Este documento describe las actividades necesarias para instalar y probar la aplicación, el alcance de este documento es dar una referencia real para el proceso de deployment de la aplicación. Este documento se encuentra en el **Apéndice C.1.**

5.4.1.4.2 Plan de pruebas

El propósito de este documento es definir las actividades de pruebas a cubrir, establece el tipo de pruebas, requerimientos a probar, determinación de la estrategia de prueba y los recursos requeridos para llevarlas a cabo. Este documento se encuentra en el **Apéndice C.2.**

5.4.1.4.3 Prototipo Operacional, versión alpha

Es la versión estable del sistema desarrollado, al prototipo operacional se le aplicarán las actividades indicadas en el documento de plan de pruebas. Este prototipo se convertirá en la versión alfa (lanzamiento productivo) del sistema después de concluir las pruebas con resultados satisfactorios.

5.4.2 Estructura Estática

La estructura estática establece las actividades específicas de cada uno de los integrantes del equipo de desarrollo, así como la forma de realizar estas actividades. RUP define cuatro elementos en la estructura estática (Figura 5.14):

- Roles.
- Actividades.
- Flujo de trabajo.

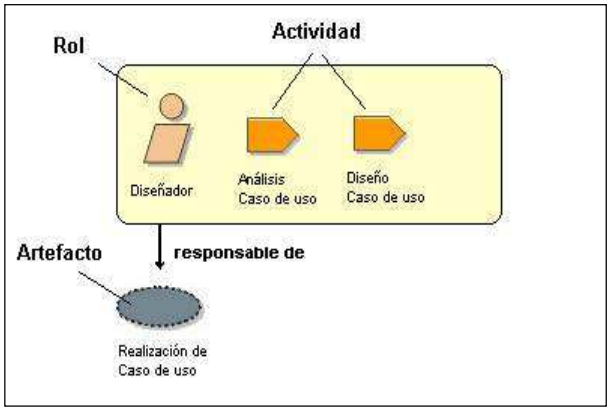

Figura 5.14 Elementos de la estructura estática

5.4.2.1 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.

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

Analistas:

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

Desarrolladores:

- Arquitecto de software.
- 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.

Especialistas en pruebas:

- Especialista en Pruebas (tester).
- Analista de pruebas.
- Diseñador de pruebas.

Otros roles:

- Stakeholders.
- Revisor.
- Coordinación de revisiones.
- Revisor técnico.
- Cualquier rol.

5.4.2.2 Actividades

Una actividad es una unidad de trabajo que es asignado a un rol específico. Las actividades tienen un objetivo concreto, normalmente expresado en términos de crear o actualizar algún producto.

5.4.2.3 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.

5.4.2.4 Flujos de trabajo

Un flujo de trabajo es una relación de actividades que nos producen unos resultados observables. RUP determina los siguientes flujos de trabajo:

- Modelado de negocio.
- Requisitos.
- Análisis y diseño.
- Implementación.
- Pruebas.
- Despliegue.
- Gestión del proyecto.
- Configuración y control de cambio.
- Ambiente.

5.4.2.4.1 Modelado del negocio

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

Los objetivos del modelado de negocio son:

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

El modelo de negocio describe como desarrollar una visión de la nueva organización, basado en esta visión se definen procesos, roles y responsabilidades de la organización por medio de un modelo de Casos de Uso del negocio y un Modelo de Objetos del Negocio. Complementario a estos modelos, se desarrollan otras especificaciones tales como un Glosario.

5.4.2.4.2 Requisitos

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

Los objetivos del flujo de datos Requisitos son:

- Establecer y mantener un acuerdo entre clientes y otros stakeholders sobre lo que el sistema podría 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.

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 (Figura 5.15).

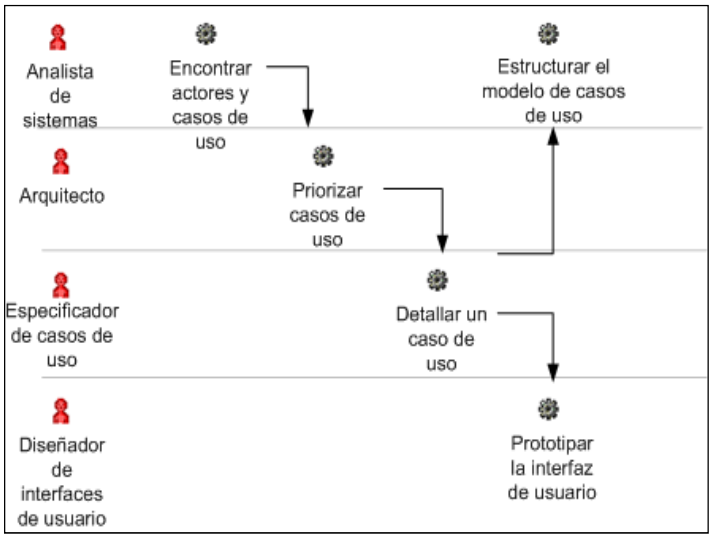

Figura 5.15 Requisitos

5.4.2.4.3 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:

- 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 (Figura 5.16). 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.

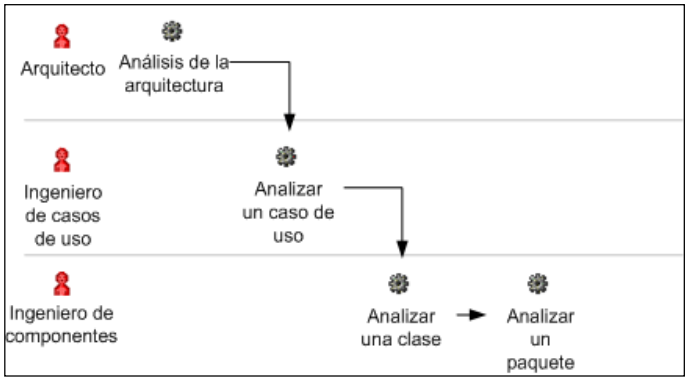

Figura 5.16 Análisis

El resultado final más importante de este flujo de trabajo será el modelo de diseño (Figura 5.17), el cual consiste en colaboraciones de clases, que pueden ser agregadas en paquetes y subsistemas.

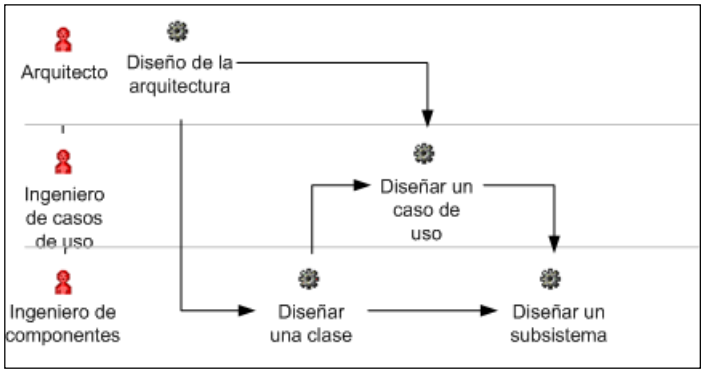

Figura 5.17 Diseño

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

5.4.2.4.4 Implementación

En este flujo de trabajo se implementan las clases y objetos en archivos fuente, binarios, ejecutables y demás. El resultado final de este flujo de trabajo es un sistema ejecutable (Figura 5.18).

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.

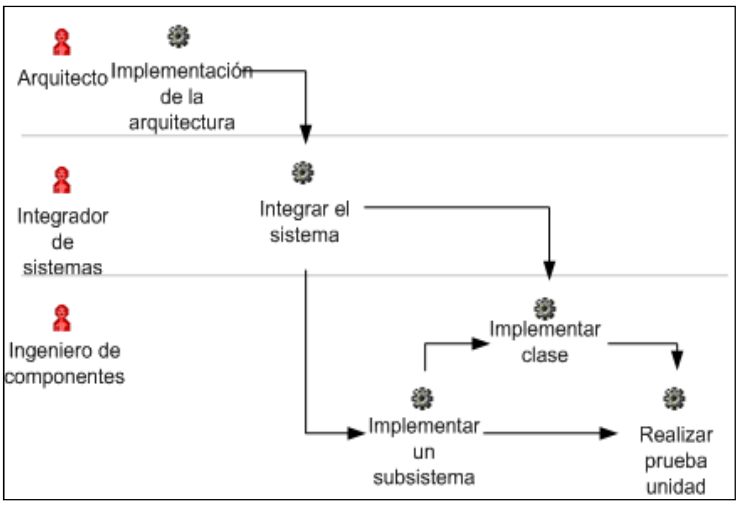

Figura 5.18 Implementación

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.

5.4.2.4.5 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.

Sus objetivos de las pruebas son:

- 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.

El desarrollo del flujo de trabajo consistirá en planificar los requerimientos a 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 (Figura 5.19).

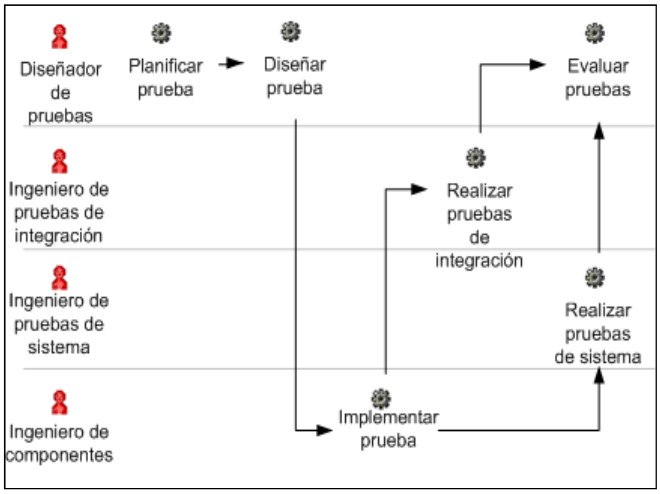

Figura 5.19 Pruebas

5.4.2.4.6 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.

5.4.2.4.7 Gestión del proyecto

La Gestión del proyecto nos permite lograr un balance al gestionar objetivos, riesgos y restricciones para 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 cada iteración.

5.4.2.4.8 Configuración y control de cambios

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

5.4.2.4.9 Ambiente

La finalidad de este flujo de trabajo es dar soporte al proyecto con las adecuadas herramientas, procesos y métodos. Brinda una especificación de las herramientas que se van a necesitar en cada momento, así como 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.