Capítulo 3

Metodologías conducidas por los planes

El Proceso Unificado de Desarrollo de Software 2
Conducido por los planes 4
El flujo de trabajo 6
Las fases y los hitos 6
Las iteraciones 9
El organigrama, los roles y actividades 10
Roles asociados a la organización 11
Roles derivados de la forma de trabajo 12
Los activos como nexo entre los roles 14
Los hitos y entregables 14
Planes 18
Preventa 18
Plan de proyecto 19
Otros planes 20
Activos de trabajo 21
Especificación de Requerimientos de Software 21
Especificación de Arquitectura de Software 21
Conclusión 22
Referencias 22

La Ingeniería de Software, como analizamos en los dos capítulos anteriores, si bien tiene una vida mucho más corta que otras disciplinas ha tenido una evolución acelerada. Los cambios que han sucedido se podrían separar en dos tipos, por un lado los generados por la tecnología y así podemos llamarlos técnicos y otros cuya esencia radica en la forma de trabajar para llevar adelante un proyecto de desarrollo. No estamos diciendo que ambos no estén relacionados, lo están. Se expresan por separado y en este sentido podemos tratarlos como tal, pero su vínculo y la relación entre ellos es más compleja.
En este capítulo analizaremos un aspecto de la Ingeniería de Software que consideramos de importancia, las metodologías conducidas por los planes.
Como explicamos en el capítulo anterior, una metodología no es más que una forma de trabajo. Sin embargo es clave a la hora de organizar las tareas de un grupo de desarrollo. Remarcamos que forma de trabajo de los miembros de un grupo significa en este contexto; las actividades que cada uno realiza, los productos que cada uno genera y en qué momento y cómo estos productos son utilizados por el resto de los miembros. Todo esto conducido por el objetivo de construir un sistema que resuelva un problema determinado. Esta definición es amplia y un tanto ambigua pero nos sirve para introducir diferentes variantes que podrían ser comprendidas en esta definición y así reforzar las diferencias.
Al momento de escribir este libro y con la intención de simplificar esta presentación, podríamos decir que existen dos tipos de metodologías. Unas las conducidas por los planes y otras las ágiles. En este capítulo trataremos las primeras. Cuando decimos que una metodología es conducida por los planes lo que estamos expresando es que la disciplina de los miembros del grupo de desarrollo está basada en la elaboración y seguimiento de planes y documentación en forma previa a desarrollar todas y cada una de las tareas comprendidas en el desarrollo del proyecto.

El Proceso Unificado de Desarrollo de Software

Tomaremos como ejemplo de este tipo de metodología al Proceso Unificado de Desarrollo de Software [1]. En la década de 1970 existían en la industria del desarrollo de software dos estándares instalados de hecho, uno era el lenguaje "C" de programación y el otro era el Desarrollo Estructurado de Sistemas [2] como paradigma de las metodologías llamadas trabajo en Cascada [3] [4]. Estas metodologías, como analizamos en el capítulo anterior, consistían en el desarrollo del proceso en una cascada de tareas. Las dificultades que presentaban estas metodologías radicaban en:

1. Falta de una relación estrecha con los requerimientos en todo el ciclo de vida del proyecto de desarrollo
2. Falta de una especificación estable de la arquitectura del software
3. Una dinámica de trabajo con poca posibilidad de realimentación entre las distintas etapas del proceso de desarrollo

Tanto las metodologías en Cascada como su transición al Proceso Unificado de Desarrollo de Software son un claro ejemplo de la estrecha relación entre los conceptos técnicos y los relacionados con la forma de trabajo. Afirmamos esto debido a que las falencias mencionadas fueron resueltas en parte por la introducción de un nuevo paradigma basado en la programación orientada a objetos y los lenguajes que la implementaron.
Hacia el año 1990 ya existían lenguajes orientados a objetos y una forma de trabajo que no se terminaba de estandarizar pero que era utilizada por los desarrolladores de la época [5] [6]. No fue hasta el año 1999 en que fue publicado el libro citado como [1] cuando comenzó a popularizarse esta nueva forma de trabajo, la cual fue acompañada por la mayor madurez de los compiladores de código C++ [7], lenguaje que se convertiría en estándar y reemplazaría a "C".
Esta metodología propuso resolver las falencias en el proceso de desarrollo citadas para las metodologías en Cascada en tres pilares básicos:

1. Proceso Conducido por los Casos de Uso
2. Proceso Centrado en la Arquitectura
3. Proceso Iterativo e incremental

Los Casos de Uso, como veremos en un próximo capítulo, son una forma de especificar requerimientos funcionales. Esta forma tiene una expresión gráfica que permite realizar diagramas y así abarcar un grupo de funcionalidades asociadas, es decir cohesivas desde el punto de vista del negocio. Además constan de una forma textual que permite detallar dicha especificación. Más aún, las pruebas funcionales basan sus casos de prueba en los casos de uso. Además, verificamos el sistema construido en función de la implementación de la totalidad de los requerimientos, basándonos en los casos de uso. Por esta razón se dice que el proceso es conducido por los casos de uso, ya que están presentes en todo el ciclo de vida.

Al estar inspirada esta nueva metodología en la programación orientada a objetos, toma fuerza y es más fácil de incorporar el concepto de Arquitectura de Software. Esto es debido a que los objetos como concepto poseen identidad y alcance lo que facilita el reconocimiento y agrupamiento de componentes en partes. Estas partes y la manera en que se comunican son a lo que llamaremos Arquitectura de Software. Los lenguajes orientados a objetos a partir de su propiedad de encapsulamiento hicieron posible definir arquitecturas estables para los sistemas en desarrollo. La nueva metodología puso énfasis en la definición temprana de la arquitectura y su prueba con relación a la funcionalidad derivada de los casos de uso que debía soportar.

Por último y por la misma razón, el encapsulamiento aportado por los lenguajes, fue posible trabajar en forma iterativa e incremental. Esto es, ir construyendo pedazos cerrados del software de manera progresiva a lo largo del ciclo de vida del proyecto. De esta manera fue posible analizar en detalle, diseñar, construir, probar e integrar el código que implementa una dada funcionalidad al sistema en desarrollo sin modificar lo ya construido. Esto

permite contar con implementaciones parciales y tempranas del sistema a efectos de aprender el negocio a medida que se desarrolla el sistema. Otra ventaja es achicar los riesgos de forma también temprana, lo cual no era posible en las metodologías previas. Esta imposibilidad era lo que generaba grandes inestabilidades en los proyectos en etapas tardías, al tratar de mitigar algunos riesgos cuando el proyecto estaba cerca de su fecha de finalización (cambios en la arquitectura por cambios en alguna funcionalidad, pruebas integrales a la arquitectura, integración de funcionalidades).
Como veremos en una próxima sección, esta metodología posee una dinámica expresada en fases y un conjunto de productos que los distintos roles generarán en cada momento.
Pero nos interesa analizar el aspecto vinculado a los planes de esta forma de trabajo, porque ese es el objetivo del capítulo.

Conducido por los planes

Cada vez que necesitamos realizar un propósito, conciente o inconcientemente, prolija o desprolijamente, documentado o no documentado realizamos un plan. Este plan no es más que una ayuda o guía en el desarrollo del propósito mencionado. Seguramente incluimos el listado de actividades a realizar y el orden en que las realizaremos, los recursos y objetos que utilizaremos, quienes nos ayudarán, cuándo haremos cada actividad, cómo las haremos y qué obtendremos como resultado, qué entregaremos y a quiénes.
Este plan tiene dos objetivos, el primero es realizar el ejercicio de planificación y el segundo es contar con un documento de referencia para todos los involucrados en el proyecto asociado a nuestro propósito. Si las tareas a realizar son complejas, podemos incluir que acciones y alternativas evaluaremos seguir ante determinados obstáculos a los cuales llamaremos riesgos.
Este proyecto bien podría ser un viaje, una reunión, una expedición, la construcción de una casa o el desarrollo de un proyecto de software. El plan en cualquiera de estos casos sería la guía que consultaríamos al comienzo y durante el tiempo que dure el proyecto. Cada vez que algo no pueda ser llevado adelante como establecimos en el plan, replanificaremos justificando las razones y modificaremos el plan que a partir de ese momento seguiremos. El plan es el que conduce la evolución del proyecto como se esquematiza en la figura 3.1.
No solo el plan se utiliza para llevar adelante el proyecto sino que los participantes elaboran una serie de documentos que junto al plan constituyen los elementos conductores. Todos los involucrados utilizan estos documentos como medio de respaldar, técnicamente y desde la gestión, todos y cada uno de los aspectos relacionados al proyecto. También son utilizados para construir acuerdos de alcances, pruebas de aceptación y formas de trabajo.
Los documentos son vistos como registros válidos de la realización de las actividades asociadas y son la mayoría de las veces entregados a los clientes, de allí su nombre genérico de entregables.

La disciplina de trabajo en esta metodología está representada por la elaboración de dichos documentos y la calidad de los mismos, entendiéndose por calidad su adherencia a estándares predefinidos.
Estos documentos están ligados a los momentos del proyecto y a las actividades realizadas en esos momentos. Estas relaciones las analizaremos en una próxima sección.
Para elaborar el plan que conducirá el proyecto es necesario predecir de antemano los acontecimientos que sucederán durante el desarrollo. Con este fin es que se detectan los posibles riesgos y luego se administran de modo que no afecten el desarrollo predicho y no debamos apartarnos del plan. Con este fin es que se realizan estimaciones de esfuerzo con tiempos de resguardos incluidos de modo que la incertidumbre de estas estimaciones no afecte la planificación y debamos modificar el plan. Este gran trabajo realizado al comienzo de nuestro proyecto con información no del todo certera es una de las críticas que se le hace a estas metodologías y es una de las razones por las que surgieron un par de décadas después las metodologías ágiles que presentaremos en un próximo capítulo.

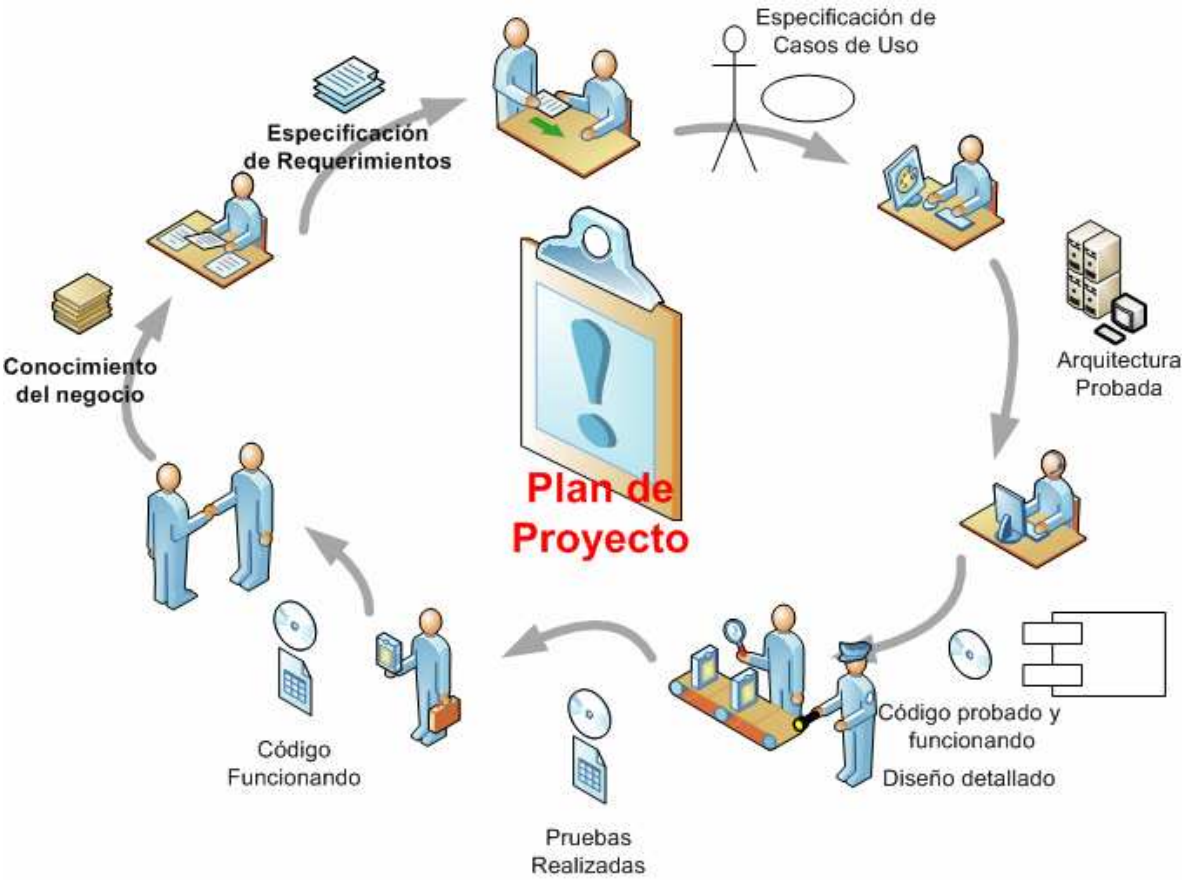

Figura 3.1 Flujo de trabajo en una metodología conducida por los planes

El flujo de trabajo

Las fases y los hitos

El Proceso Unificado de Desarrollo de Software posee cuatro fases. Cada una de ellas termina dando lugar a un hito importante en el ciclo de vida del proyecto. Como se muestra en la figura 3.2 de más abajo estas fases se dan en secuencia, una termina y comienza la próxima. Hemos agregado a las fases originales de esta metodología una fase inicial de preventa en la cual se trabaja a efectos de vender el proyecto de desarrollo. A esta fase nos referiremos cuando tratemos la gestión de los proyectos en un capítulo más adelante. La diferencia sustancial con la secuencia de tareas de las metodologías en cascada es que aquí en cada una de las fases se llevan adelante un conjunto de tareas. Estas tareas son las listadas verticalmente en la figura. La parte coloreada de la figura representa la intensidad con que se realizan cada una de estas tareas a lo largo del ciclo de vida del proyecto. En la parte inferior de la figura se indican las iteraciones de cada fase. Podemos ver que en este caso hubo una única iteración durante la fase de concepción y dos durante la fase de elaboración.

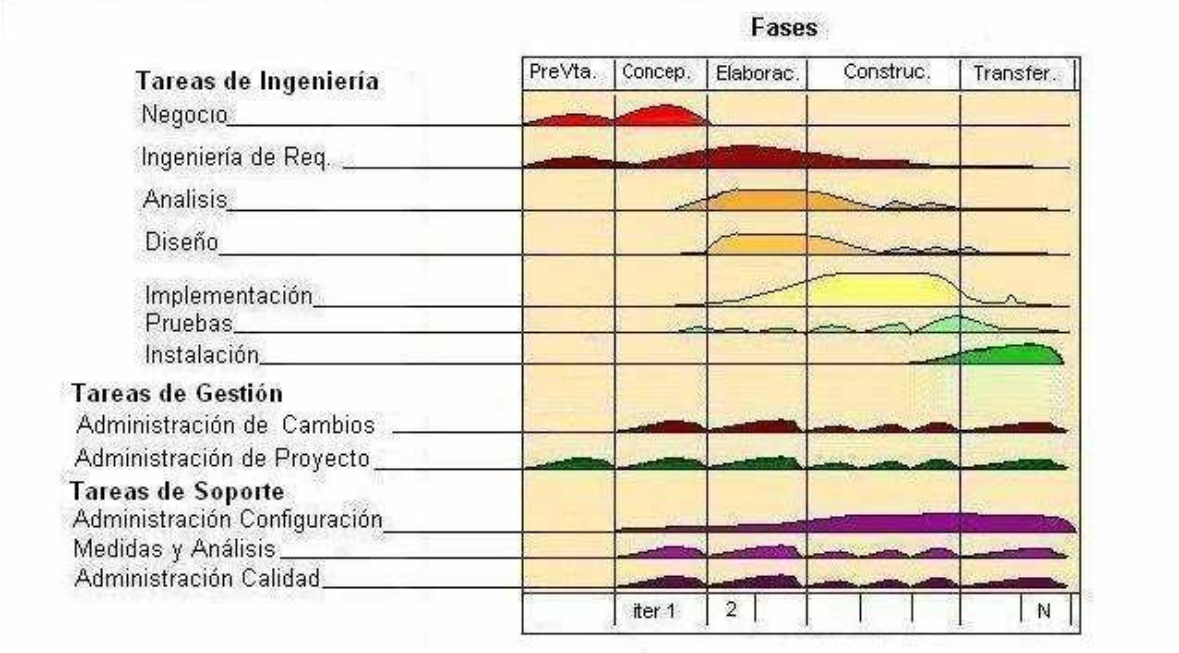

Figura 3.2 – Las fases y tareas del Proceso Unificado de Desarrollo

Nos referiremos luego en detalle a las iteraciones.
Los hitos de finalización de cada fase están ligados al logro de los objetivos de cada fase.
Como se muestra en la tabla 3.1, adaptada de [8], el primer hito es establecer la factibilidad acordada del proyecto a partir de los requerimientos / información relevada durante la fase de concepción. El objetivo para esta fase es achicar los riesgos del negocio, en relación a su comprensión.
El segundo hito, vinculado con el fin de la fase de elaboración, es una alternativa de solución probada. Nos referimos a ella como arquitectura

probada. El objetivo para esta fase es achicar los riesgos derivados de la tecnología en relación con la arquitectura definida para el sistema en desarrollo.
El tercer hito, ligado a la terminación de la fase de construcción es contar con una solución disponible. Se trata aquí de achicar los riesgos asociados a la implementación en tiempo y con los recursos disponibles.
Por último, el hito final está asociado a la terminación de la fase de transferencia que consiste en cerrar el proyecto de desarrollo. El riesgo que se busca achicar es el de la instalación en el ambiente productivo.

Tabla 3.1 – Las fases y los hitos del Proceso Unificado de Desarrollo de Software

Concepción		Elaboración		Construcción		Transferencia	
Foco del riesgo • Negocio	Factibilidad Acordada	**Foco del riesgo** • Arquitectura	Alternativa Probada	**Foco del riesgo** • Logísitca	Solución Disponible	**Foco del riesgo** • Ambiente Productivo	Proyecto Terminado
Preguntas • ¿Estamos construyendo la cosa correcta para el cliente? • ¿Es la solución factible? • ¿Cuánto esfuerzo costará?		**Preguntas** • ¿Sabemos qué estamos construyendo? • ¿Sabemos cómo construirlo? • ¿Estamos de acuerdo en qué construir? • ¿Cuánto dinero costará? • ¿Cuáles son los riesgos técnicos? • ¿Podemos mitigarlos?		**Preguntas** • ¿Lo estamos construyendo? • ¿Estará terminado a tiempo? • ¿Es suficientemente bueno? • ¿Mantenemos nuestros supuestos y decisiones? • ¿Están los usuarios listos?		**Preguntas** • ¿Es aceptable? • ¿Está siendo usado? • ¿Hemos terminado?	
Artefactos claves • Visión • Riesgos • Plan de proyecto • Listado de casos de uso críticos • Modelo de negocios		**Artefactos claves** • Modelo de casos de uso • Especificación de los casos de uso principales • Especificación de la arquitectura • Prototipo de arquitectura • Prueba de arquitectura		**Artefactos claves** • Especificación de casos de uso • Diseño • Código • Pruebas • Resultado de pruebas • Material de capacitación y documentación de usuario		**Artefactos claves** • Instaladores • Convertidores de datos • Listado de últimos defectos y resoluciones	
Salidas • Acuerdo de proyecto		**Salidas** • Arquitectura probada		**Salidas** • Solución probada, documentada y lista para instalar		**Salidas** • Solución instalada y funcionando en forma productiva	

Las iteraciones

Como ya mencionamos, una de las claves de esta metodología es su dinámica iterativa e incremental. En la figura 3.3, adaptada de [8], se muestra este comportamiento. Puede verse la realización de todas las tareas asociadas a la implementación de los casos de uso seleccionados para las iteraciones. Al término de cada una se obtiene un incremento del producto en desarrollo, el cual es evaluado por todos los involucrados en el proyecto.

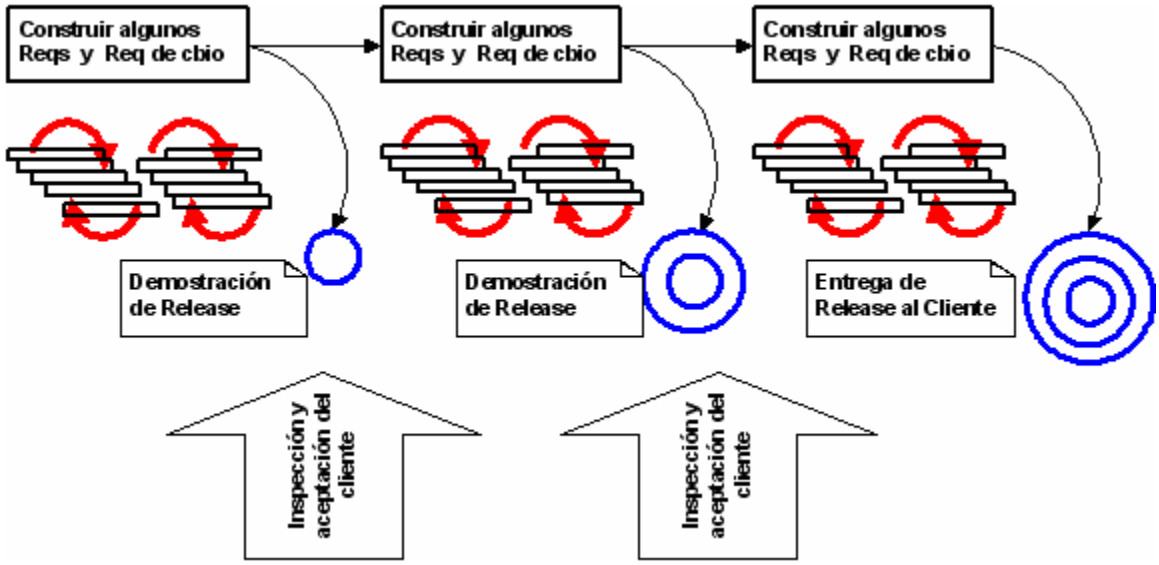

Figura 3.3 – Esquema de la dinámica iterativa e incremental

Se pueden desarrollar iteraciones que incluyan la realización de un número fijado de casos de uso o desarrollar iteraciones de duración fija en las cuales se implementarán la mayor cantidad de casos de uso posible. Esta selección deberá ser priorizada como trataremos en un capítulo próximo. Existen ventajas en esta última alternativa están dadas por la posibilidad de aprender por parte del grupo de desarrollo la propia velocidad de desarrollo y además mantener bajo control la planificación. Es decir, fijar un presupuesto y una planificación y trabajar todo el tiempo en maximizar la cantidad de funcionalidad priorizada que se implementa. En los capítulos dedicados a la gestión de los proyectos nos volveremos a referir a este tema.

El organigrama, los roles y actividades

Hasta los días en que las metodologías de trabajo en Cascada fueron el estándar, los miembros de los grupos de desarrollo ocupaban cargos más que desarrollar roles. El Proceso Unificado de Desarrollo de Software fue innovador y precursor de la visión de los grupos formados por personas que en distintos momentos del ciclo de vida del proyecto pudiesen desarrollar distintos roles [1]. El aspecto de la dinámica iterativa de la metodología hace que se obtengan mejores resultados con miembros especialistas en determinados conocimientos conduciendo la agenda del proyecto en una determinada fase. En la fase siguiente este miembro puede desarrollar otro rol o el mismo pero con otro grado de protagonismo. Esta rotación de roles impone un gran conocimiento, de parte de todos los miembros, de los pormenores del proyecto. También hace que todos los miembros permanezcan en el proyecto durante todo el ciclo de vida y con una carga de trabajo intensa. A diferencia de esto, era común ver en las metodologías en Cascada a personas que participaban de una fase de un proyecto como analistas, por ejemplo, para luego irse a otro proyecto. Esta característica de las metodologías iterativas hizo que se deba modificar la formación profesional de los desarrolladores, fue necesaria una formación muy buena en todos los aspectos del desarrollo de software con especialización en alguna. En un próximo capítulo volveremos a tratar este tema. Como conclusión, se cambiaron de una metodología a otra, el trabajo en cascada realizado por cargos por el trabajo iterativo llevado adelante por roles cambiantes. En la figura 3.8 que sigue se esquematiza esta forma de organización de los grupos de desarrollo.

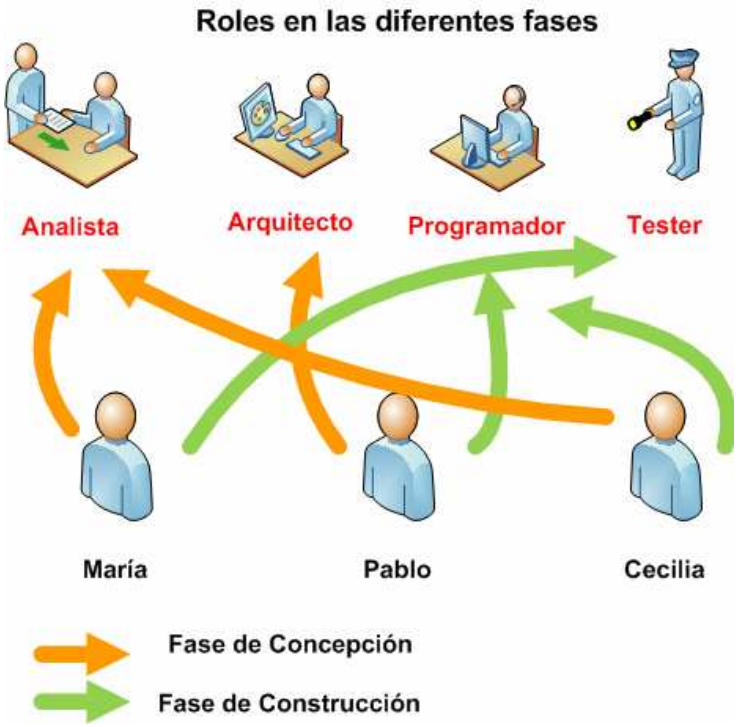

Figura 3.8 – Roles intercambiables en las diferentes fases

En la figura 3.9 que sigue se muestra un organigrama típico de una organización dedicada al desarrollo de software. En ella aparecen los diferentes roles que hacen al funcionamiento de cualquier empresa de este tipo.

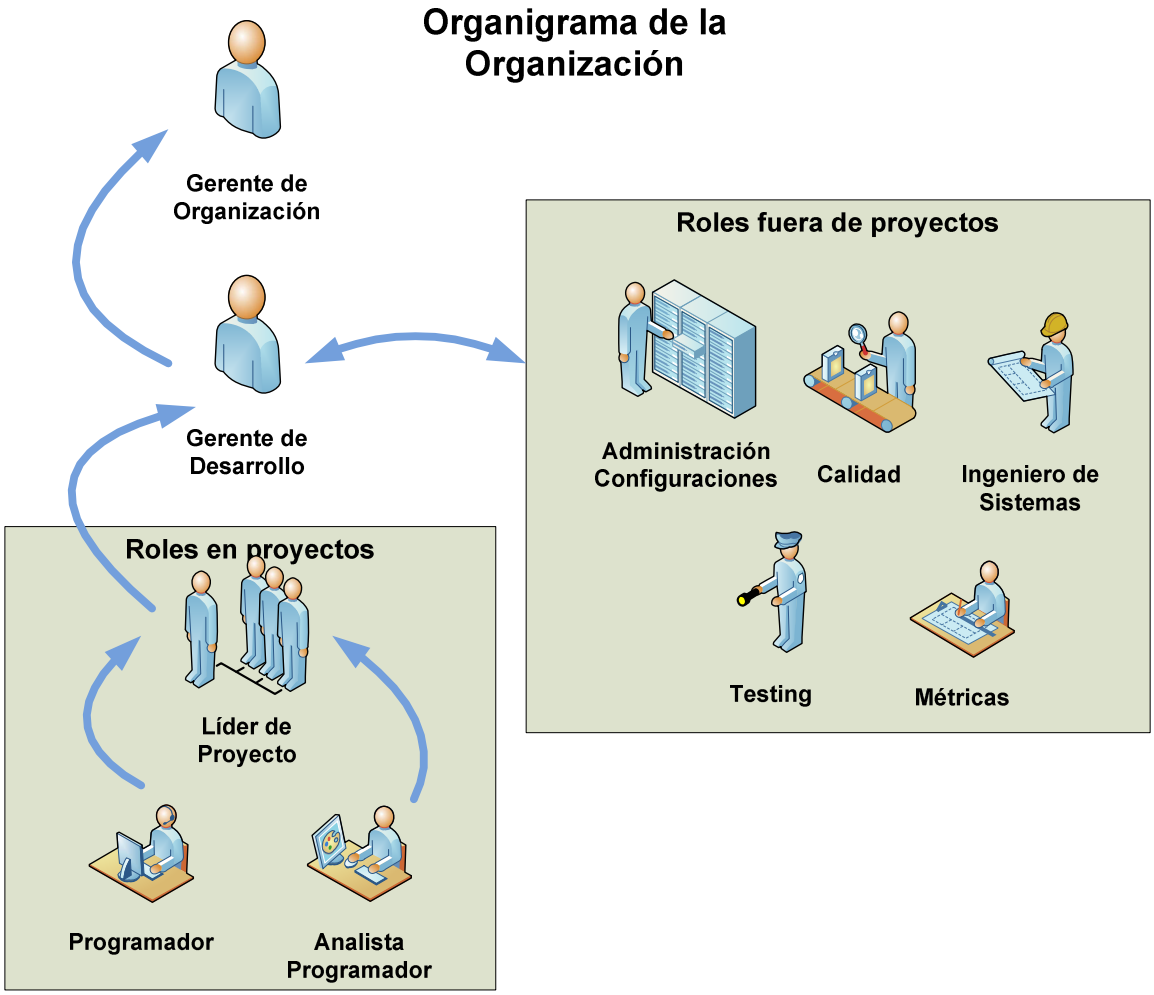

Figura 3.9 - Organigrama de una organización típica de desarrollo de software

Como muestra la figura 3.9 hay roles en una organización de este tipo que funcionan a nivel de la organización y otros a nivel de los proyectos. A continuación los enumeramos y listamos sus actividades típicas.

Roles asociados a la organización

Gerente de Desarrollo: administra el área de desarrollo proveyendo recursos a los proyectos y participando en el seguimiento de todos ellos. Es responsable del área de desarrollo y del cumplimiento de las políticas de la organización.

Gerente de Calidad: gestiona el área de calidad reportando al CEO de la organización. Administra los recursos para el control de calidad en los proyectos de desarrollo. Participa de las definiciones de estándares de calidad.

Responsable de Calidad: desarrollo el control de calidad en los proyectos de desarrollo.

Roles derivados de la forma de trabajo

Líder de Proyecto: gestiona un proyecto de desarrollo acordando con el cliente y facilitando el trabajo al grupo de desarrollo. Es responsable del proyecto y del grupo de desarrollo.

Analista Programador: posee una doble interfaz, una hacia los clientes en el relevamiento de requerimientos y otra hacia el grupo de desarrollo en el análisis de los requerimientos, planificación de pruebas funcionales y participación en la definición de la arquitectura.

Arquitecto: es el responsable técnico del proyecto, define la tecnología a utilizar, el ambiente de desarrollo y la arquitectura del sistema en desarrollo del proyecto.

Programador: realiza diseño de software, codifica y prueba.

Tester: ejecuta y recolecta resultados de las pruebas.

Los roles listados participan en los distintos momentos del ciclo de vida del proyecto realizando diferentes actividades. En la tabla 3.2 que sigue adaptada de [8], se muestran las actividades realizadas en cada fase. En esta tabla dichas actividades están separadas de acuerdo a cómo los roles se relacionan con el desarrollo. Estos enfoques son el problema, la solución y el proyecto. Si bien estos enfoques vinculan directamente al analista con el aspecto del problema, a los roles técnicos con la solución y al líder con el proyecto, todos ellos comparten su visión con los otros roles, aunque con diferente protagonismo.

Tabla 3.2 – Las fases y las actividades del Proceso Unificado de Desarrollo de Software

	Concepción		Elaboración		Construcción		Transferencia	
Problema	• Identificar los requerimientos principales (arquitectura) • Entender el ambiente operacional • Definir el problema • Determinar el valor de resolver el problema	Factibilidad Acordada	• Estabilizar los requerimientos • Detallar los requerimientos críticos • Refinar la visión	Alternativa Probada	• Completar los requerimientos • Construir la documentación • Administrar los cambios de requerimientos • Facilitar la disponibilidad de los usuarios • Solicitar requerimientos de cambios	Solución Disponible	• Llevar adelante las pruebas de aceptación • Entrenar a los usuarios • "Market" la solución • Administrar los cambios en la comunidad de los usuarios • Sugerir mejoras • Informar defectos y deficiencias	Proyecto Terminado
Solución	• Explorar soluciones posibles • Evaluar soluciones alternativas • Proponer la arquitectura		• Probar la arquitectura • Desarrollar prototipos • Describir la arquitectura • Describir las interfaces entre subsitemas y sistemas externos • Seleccionar componentes		• Desarrollar componentes • Probar y evaluar • Refinar la arquitectura • Optimizar el diseño de componentes		• Instalar a los usuarios finales • Prepararse para el mantenimiento • Corregir defectos • Ajustar la aplicación • Contrastar funcionamiento y migrar datos	
Proyecto	• Establecer el alcance del proyecto • Planificar el ciclo de vida • Establecer los costos • Construir el modelo de negocio • Identificar los riesgos críticos		• Monitorear el progreso • Mejorar las estimaciones • Planear el desarrollo • Controlar los riesgos • Establecer la infraestructura del proyecto • Definir las métricas • Dar recursos al proyecto • Refinar el proceso		• Analizar el impacto • Monitorear y controlar el desarrollo • Planear la instalación • Optimizar los procesos • Monitorear los riesgos • Tomar las métricas • Optimizar el uso de recursos • Controlar los costos		• Evaluar el impacto del cambio • Programar cambios • Asignar soporte • Monitorear y controlar instalación • Cerrar el proyecto	

Los activos como nexo entre los roles

Los hitos y entregables

Según dijimos en una sección anterior esta metodología basa su disciplina en los planes y la documentación. Analizaremos entonces como se relacionan los hitos del ciclo de vida con los productos / documentos generados y entregados y cómo están vinculados los involucrados a estos entregables. En la figura 3.4 se presenta un esquema correspondiente a la fase de concepción del proyecto.

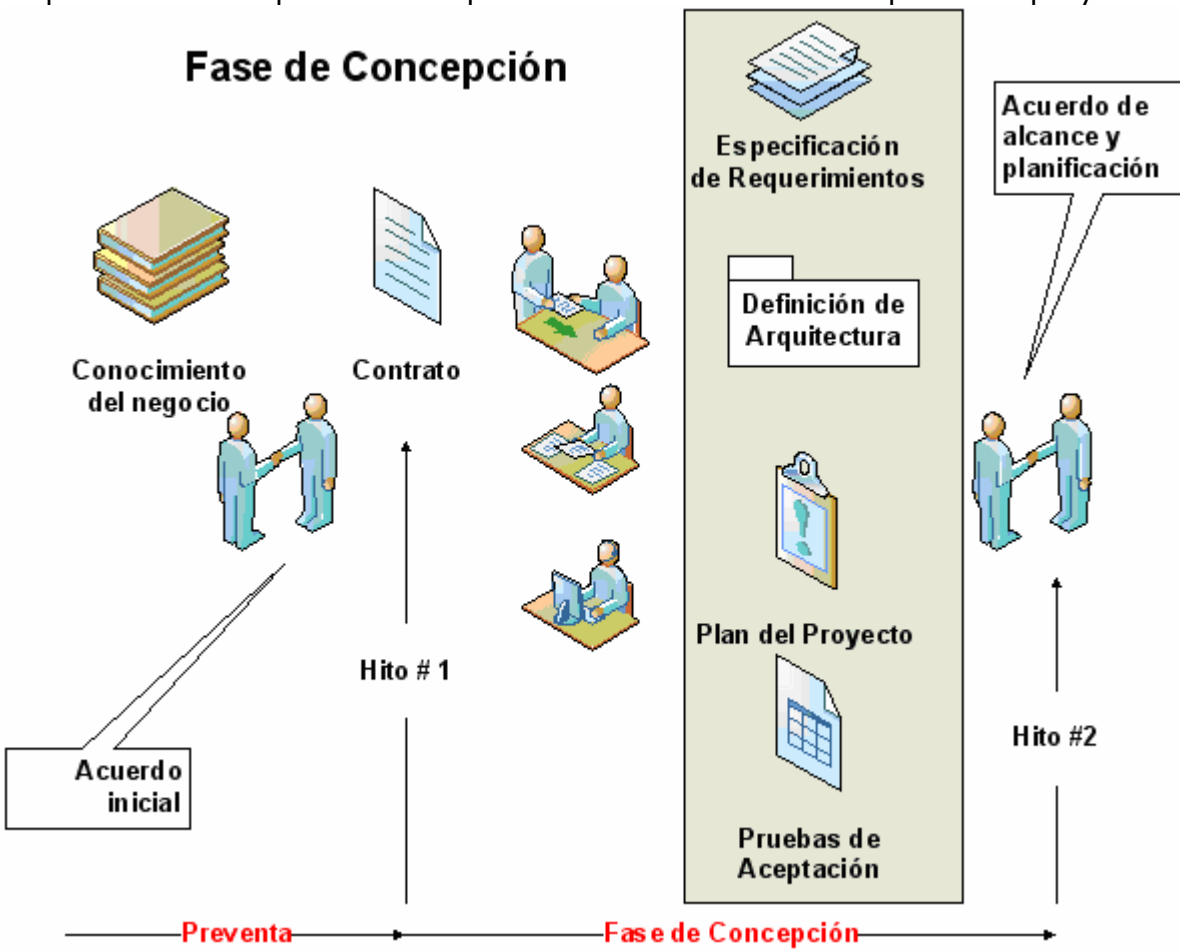

Figura 3.4 – Productos generados y entregables de la fase de concepción

El primer hito del proyecto está dado por la firma del contrato entre la organización cliente y la desarrolladora. El documento ligado a este hito es el contrato correspondiente. Este hito determina el comienzo del proyecto. Los miembros del grupo de desarrollo más el resto de los involucrados desarrollando las actividades que se analizan en una próxima sección construyen los siguientes productos con el objetivo que se indica:

- ERS (Especificación de Requerimientos de Software): definición y acuerdo del alcance del proyecto.

- ⇒ EDA (Especificación de Arquitectura): definición y listado de alternativas de la arquitectura y ambiente de desarrollo.
- ⇒ Plan de Proyecto: establecimiento de una estrategia para el desarrollo del proyecto (estimación de esfuerzo, listado de tareas, recursos necesarios, riesgos a mitigar, cronograma, infraestructura necesaria, capacitaciones necesarias, organigrama y responsabilidades del proyecto, hitos y entregables)
- ⇒ Otros Planes: administración de la configuración y de la calidad del proyectio.

La aprobación / acuerdo de lo definido en estos documentos determina el hito de finalización de esta primera fase y el comienzo de la segunda.

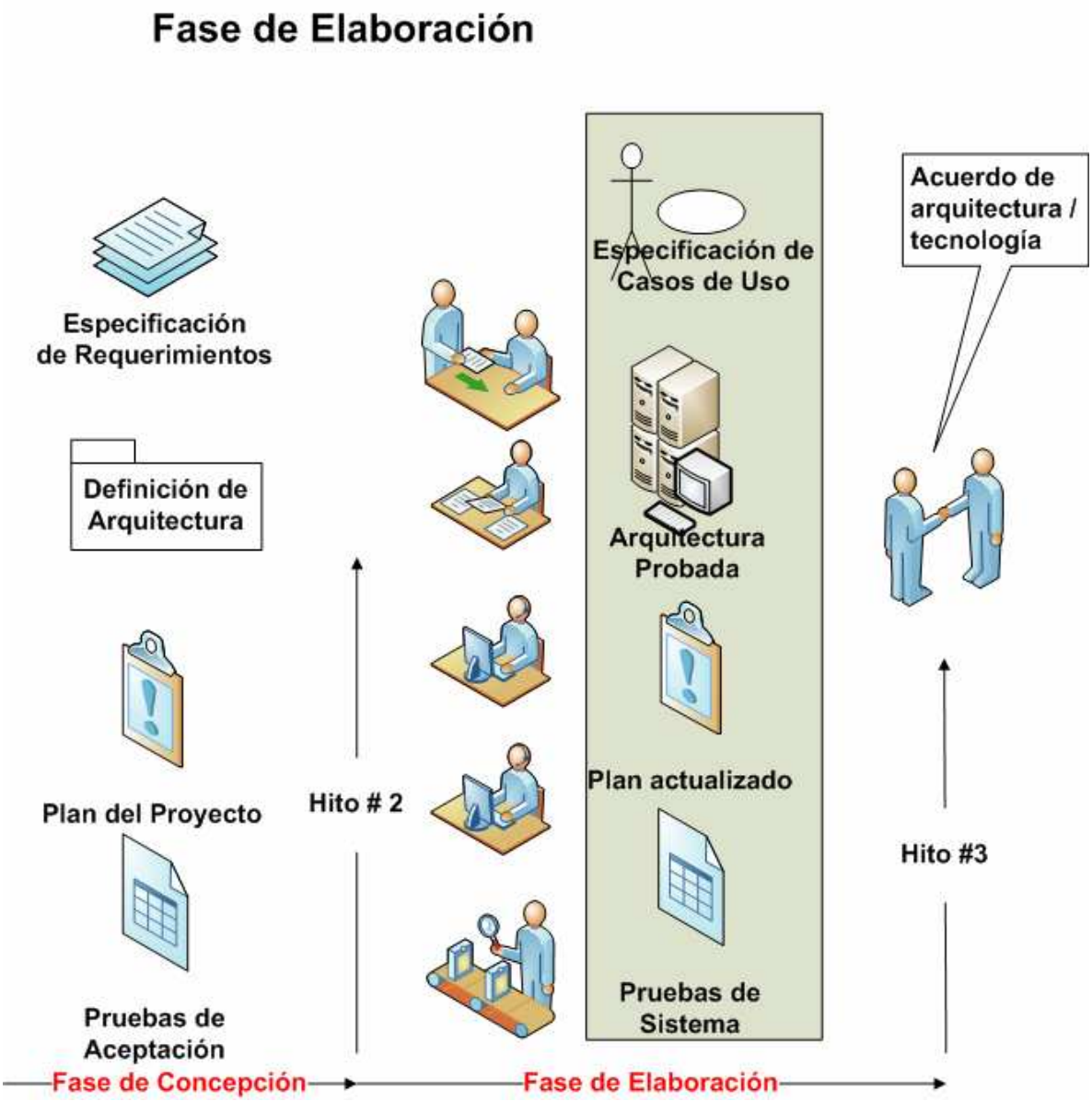

Figura 3.5 – Productos generados y entregables de la fase de concepción

En la figura 3.5 se presenta un esquema similar correspondiente a la fase de elaboración del proyecto. Los miembros del grupo de desarrollo más el resto de los involucrados desarrollando las actividades que se analizan en una próxima sección construyen los siguientes productos con el objetivo que se indica:

- ⇒ ERS (Especificación de Requerimientos de Software): especificación de los casos de uso principales (que afectan a la arquitectura)
- ⇒ EDA (Especificación de Arquitectura): alternativa seleccionada probada.
- ⇒ Plan de Proyecto: estimaciones de esfuerzo adaptadas, iteraciones definidas, cronograma adaptado, riesgos actualizados
- ⇒ Plan de Pruebas: pruebas de sistema realizadas y a realizar.

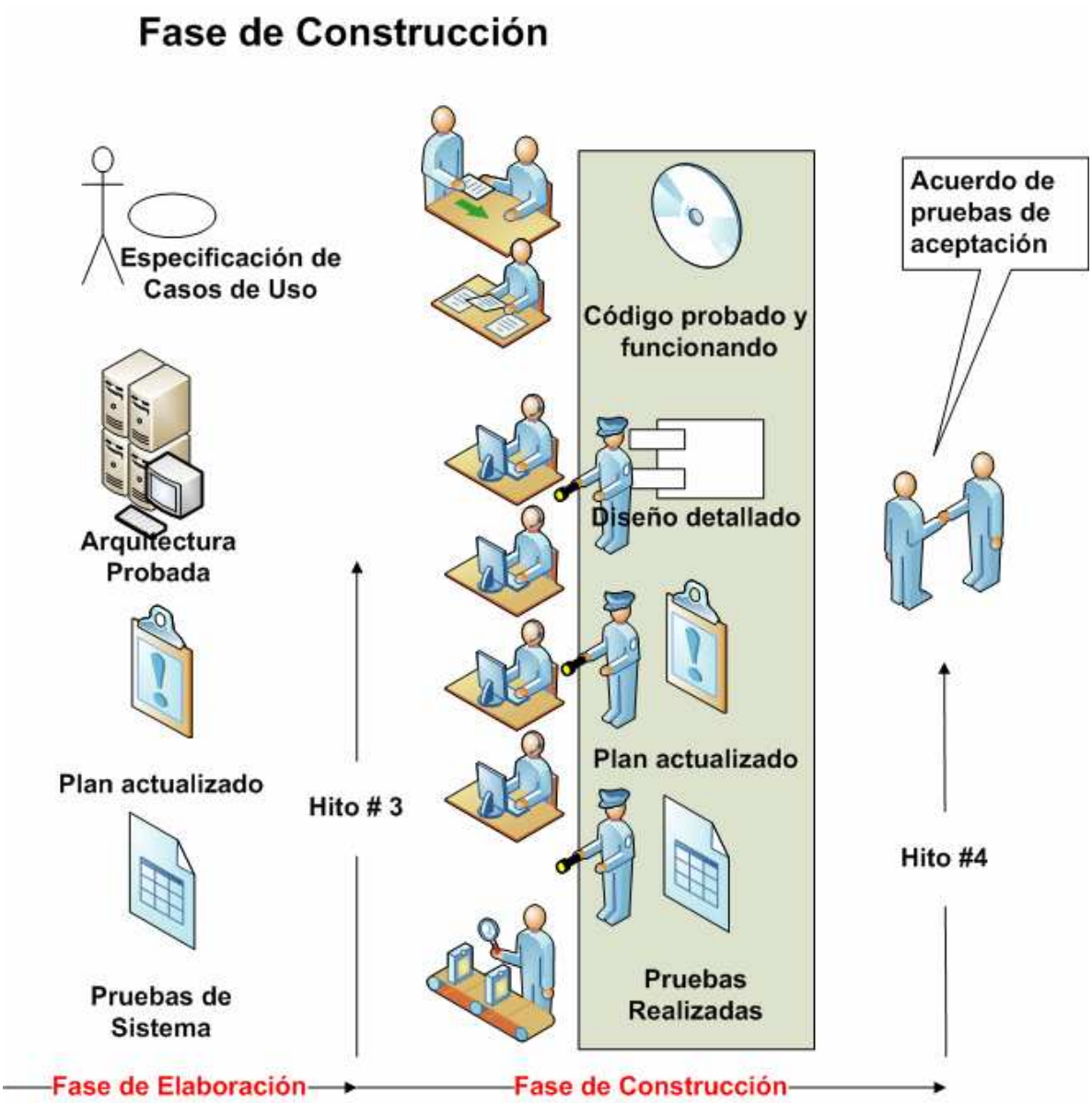

Figura 3.6 – Productos generados y entregables de la fase de elaboración.

En la figura 3.6 se presenta un esquema similar correspondiente a la fase de construcción del proyecto. Los miembros del grupo de desarrollo más el resto de los involucrados desarrollando las actividades que se analizan en una próxima sección construyen los siguientes productos con el objetivo que se indica:

- ⇒ ERS (Especificación de Requerimientos de Software): especificación de todos los casos de uso
- ⇒ EDD (Especificación de Diseño detallado): diseño detallado de los distintos paquetes
- ⇒ Plan de Proyecto: estimaciones de esfuerzo adaptadas, informes de las iteraciones realizadas, cronograma adaptado, riesgos actualizados
- ⇒ Plan de Pruebas: resultados de las pruebas realizadas.

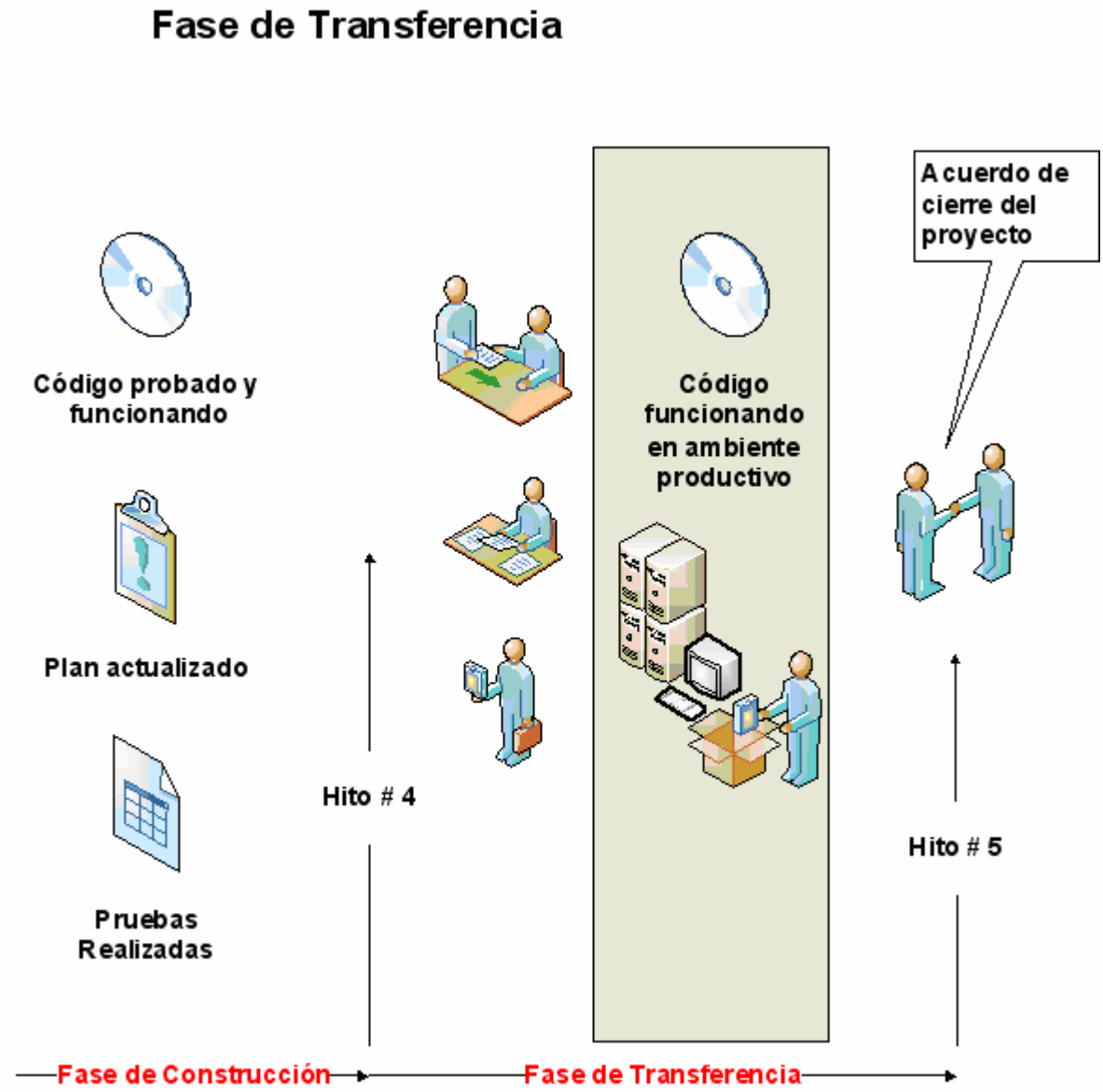

Figura 3.7 – Productos generados y entregables de la fase de transferencia.

En la figura 3.7 se presenta un esquema similar correspondiente a la fase de transferencia del proyecto. Los miembros del grupo de desarrollo más el resto de los involucrados desarrollando las actividades que se analizan en una próxima sección construyen los siguientes productos con el objetivo que se indica:

- ⇒ Plan de Proyecto: informes de seguimiento y cierre del proyecto
- ⇒ Plan de Pruebas: resultados de las pruebas de aceptación.
- ⇒ Guía de instalación y manual de usuario
- ⇒ Copia de código binario

A partir de esta secuencia de imágenes que muestran la forma en que los roles participantes desarrollan su actividades generando productos que son adaptados y / o consumidos por los roles en las fases que siguen queremos enfatizar la disciplina de esta metodología basada en estos productos / documentos intercambiados.
El otro aspecto que caracteriza a esta metodología, como ya lo mencionamos, es que todo este flujo de trabajo es planeado de antemano en los inicios del ciclo de vida del proyecto.

Planes

Como enunciamos en el inicio del capítulo, cuando decimos que esta metodología es conducida por los planes lo que estamos expresando es que la disciplina de los miembros del grupo de desarrollo está basada en la elaboración y seguimiento de planes y documentación en forma previa a desarrollar todas y cada una de las tareas comprendidas en el desarrollo del proyecto.

Preventa

Analicemos la planificación de un proyecto en el que se ha decidido la utilización del Proceso Unificado de Desarrollo de Software como metodología. Todos los temas que siguen, asociados a la planificación, los analizaremos en detalle en los capítulos que tratan la gestión de los proyectos.
Al momento de concretar el acuerdo entre el cliente y la empresa desarrolladora y firmar el contrato correspondiente se contará con la siguiente información:

- ⇒ Una Visión del Proyecto
- ⇒ Una definición preliminar (con información sin relevar y otra sin analizar) del alcance del sistema ha desarrollar
- ⇒ Una estimación preliminar (por basarse en un alcance preliminar) de esfuerzo
- ⇒ Una definición preliminar de la tecnología

- ⇒ Una definición preliminar del grupo de desarrollo
- ⇒ Un presupuesto
- ⇒ Una definición de la forma de trabajo
- ⇒ Una planificación preliminar (duración del proyecto)

Como puede verse esta información es en su mayor parte de carácter preliminar, ya que se ha trabajado un tiempo acotado para elaborar una propuesta técnico / económica por parte de la empresa desarrolladora hacia el cliente. Este corto tiempo, dependiendo del proyecto, puede ser un par de semanas un par de personas. Esta propuesta se conoce con el nombre de anteproyecto en otras ingenierías. Es de fundamental importancia esta fase del proyecto ya que como Steve M´Connell dice en [9] esta es una de las oportunidades en que se evalúa la factibilidad del proyecto. El hecho de acordar su realización implica que ambas partes consideran factible el desarrollo en los términos y condiciones que se expresan en la correspondiente propuesta. No habrá sido trabajo en vano obtener como conclusión de esta tarea que el proyecto no es factible, ya que de no haberse concluido en este momento las pérdidas, para ambas partes, cuando el proyecto estuviese en estado avanzado hubiesen sido mucho más importantes.
Es en este punto donde las metodologías se bifurcan, por un lado las conducidas por los planes y las ágiles por el otro. Básicamente se diferencian en cómo avanzar con el desarrollo del sistema del proyecto en base a esta información preliminar, con bajo grado de certeza, con poca granularidad, sin detalle, relevada en poco tiempo y casi sin analizar.
Frente a esta incertidumbre las metodologías conducidas por los planes eligen relevar y analizar los requerimientos del proyecto en una primera fase de concepción y a partir de esta información dar forma definitiva a la planificación que es volcada en detalle en el plan del proyecto que se elabora al mismo tiempo.
Las metodologías ágiles eligen elaborar una visión de la evolución del proyecto muy poco detallada e ir descubriendo / aprendiendo / adaptando esta visión mientras se desarrollan las iteraciones correspondientes y se va aprendiendo el negocio para el cual el sistema en desarrollo se construye-. Cuando en un próximo capítulo tratemos estas metodologías volveremos sobre este tema.

Plan de proyecto

Un plan de proyecto incluye los siguientes puntos:

1. **Introducción**
    a. Descripción del Proyecto
    b. Objetivos
    c. Documentos Relacionados
        i. Entregados por el Cliente
        ii. Generados por el Proyecto
    d. Destinatarios
    e. Entregables
2. **Organización del Proyecto**
    a. Planificación de la Gestión
        i. Estrategia general

        ii. Supuestos
        iii. Restricciones
        iv. Administración de riesgos
        v. Seguimiento
3. **Planificación Técnica**
    a. Ciclo de vida
    b. Requerimientos
    c. Infraestructura
    d. Organigrama
    e. Documentación a generar
    f. Capacitación a implementar
    g. Administración de Datos
4. **Relación con otros planes**
    a. Plan de CM
    b. Plan de QA
    c. Plan de MA
    d. Acuerdo con Proveedores
5. **Productos de trabajo, cronograma y presupuesto**
    a. Cronograma
        i. Estimaciones de alcance y esfuerzo
        ii. WBS
    b. Presupuesto
6. **Replanificación**
    a. Causas y fundamentación
    b. Alcance y Esfuerzo
    c. Cronograma
    d. Presupuesto
7. **Aprobación del plan**
8. **Acuerdos**
    a. Pruebas de aceptación
    b. Acuerdo del plan
9. **Control de Cambios**

Esta plantilla modelo muestra los aspectos que se documentan del ejercicio de planificación mencionado. En un próximo capítulo trataremos la gestión de los proyectos analizando cada uno de estos aspectos y presentaremos un ejemplo de un proyecto de nuestro caso testigo.

Otros planes

Plan de Administración de la Configuración

Como en todo plan en éste se planean la necesidad y disponibilidad de los recursos materiales, infraestructura, roles involucrados y tiempos del proyecto. En este caso son cubiertos los siguientes aspectos:

Recursos: software de administración del repositorio de software, software de backup y restauración del repositorio, almacenamiento de backup, etc.
Infraestructura: servidor del repositorio de software, conectividad al servidor, etc.
Roles y asignación: administrador de la configuración, administrador de la infraestructura.

Procesos y procedimientos: descripción de los procesos y procedimientos para cada tarea (agregar, cambiar y borrar ítems a la configuración, mantenimiento, auditorías, backup, etc.)
Definición de los ítems de configuración: alcance y modo de identificación de los items.
Tiempos: vinculo de las tareas con los hitos del proyecto de desarrollo.

Plan de Administración de la Calidad
Como en todo plan en éste se planean la necesidad y disponibilidad de los recursos materiales, infraestructura, roles involucrados y tiempos del proyecto. En este caso son cubiertos los siguientes aspectos:

Recursos: software de cálculo de métricas, software de oficina, etc.
Infraestructura: repositorio de documentación de procesos y procedimientos, computadoras personales de uso diario, etc.
Roles y asignación: responsable del área, controladores de la calidad y testers.
Procesos y procedimientos: descripción de los procesos y procedimientos para cada tarea (revisiones de activos y procesos, evaluación y escalamiento de no adherencias, etc.)
Definición de los procesos y procedimientos: definiciones de la organización para su proceso de desarrollo.
Tiempos: vinculo de las tareas con los hitos del proyecto de desarrollo.

Activos de trabajo

Entre los activos de trabajo típicos de esta metodología se encuentran la Especificación de Requerimientos de Software (ERS) y la Especificación de la Arquitectura (EDA). Estos activos relacionados con aspectos técnicos del proyecto se construyen y son utilizados en esta metodología con el objetivo de definir y acordar el alcance del sistema en desarrollo por un lado y los aspectos vinculados a la tecnología por el otro.

Especificación de Requerimientos de Software

Este activo tiene por objetivo documentar qué se construirá, por esta razón se utiliza para definir y acordar el alcance con el cliente a partir de los requerimientos y casos de uso asociados. Documenta aspectos funcionales, no funcionales y del proceso de desarrollo que necesitan ser comunicados a todos los involucrados en el proyecto. Cuando tratemos en un próximo capítulo el trabajo con los requerimientos lo analizaremos.

Especificación de Arquitectura de Software

Este activo tiene por objetivo documentar cómo se construirá el sistema en desarrollo, por esta razón se utiliza para definir y acordar su arquitectura. Documenta cómo los requerimientos funcionales y no funcionales serán implementados utilizando la alternativa tecnológica elegida. Cuando tratemos en un próximo capítulo el trabajo con el diseño de la arquitectura lo analizaremos.

Conclusión

En las metodologías conducidas por los planes, no solo el plan se utiliza para llevar adelante el proyecto sino que los participantes elaboran una serie de documentos que junto al plan constituyen los elementos conductores. Todos los involucrados utilizan estos documentos como medio de respaldar, técnicamente y desde la gestión, todos y cada uno de los aspectos relacionados al proyecto. También son utilizados para construir acuerdos de alcances, pruebas de aceptación y formas de trabajo.
Los documentos son vistos como registros válidos de la realización de las actividades asociadas y son la mayoría de las veces entregados a los clientes, de allí su nombre genérico de entregables.
La disciplina de trabajo en esta metodología está representada por la elaboración de dichos documentos y la calidad de los mismos, entendiéndose por calidad su adherencia a estándares predefinidos.
Estos documentos están ligados a los momentos del proyecto y a las actividades realizadas en esos momentos. Para elaborar el plan que conducirá el proyecto es necesario predecir de antemano los acontecimientos que sucederán durante el desarrollo.

Referencias

[1] *The Unified Software Development Process*, I. Jacobson, G. Booch, J. Rumbaugh, Addison Wesley, 1999.
[2] *Structured Design*, L. Constantine, E. Yourdon, Yourdon Press, 1975.
[3] *Structured Analysis and System Specification*, Tom Demarco, Prentice Hall, 1979.
[4] *Structured System Analysis: Tools and Techniques,* Gane C. and Sarson T., New York: IST, lnc., 1977
[5] *Object Oriented Analysis and Design with Applications*, Grady Booch, Cumming Press, 1992.
[6] *Object oriented Modeling and Design,* J. Rumbaugh et al., Prentice hall, 1991.
[7] *The Design and Evolution of C++, Bjarne Stroustrup*, Addison Wesley, 1994.
[8] *Managing Iterative Software Development Projects,* Kurt Bittner, Ian Spence, Addison-Wesley, 2006.
[9] *Software Project, Survival Guide*, Steve McConnell, Microsoft Press, 1998.