Guía a Rational Unified Process

Alejandro Martínez y Raúl Martínez
Escuela Politécnica Superior de Albacete – Universidad de Castilla la Mancha
e-mail: alexmv@ono.com, m_m_raul@ono.com

Resumen

Este trabajo consiste en una introducción al Proceso Unificado de Rational (RUP). Primero se ve en que principios se basa, luego se trata su estructura desde dos puntos de vista: las cuatro fases y los nueve flujos de trabajo. Para terminar se llega a las conclusiones.

1. Introducción

Este documento es una pequeña guía al Proceso Unificado de Rational. En ningún caso esto será suficiente para entender completamente el RUP, los autores recomendamos encarecidamente leer [1] y otros libros y documentos que tratan el tema con mayor profundidad.

Entonces, ¿para qué vale este documento? En primer lugar, pretendemos que sirva de introducción a aquellos que no saben nada de RUP. Dedicar un rato a hojear este documento debe dar las ideas principales, con la ventaja añadida de que esto está en castellano. El segundo objetivo es el de servir de referencia rápida cuando se está aplicando el RUP para desarrollar un proyecto.

La primera sección trata sobre las bases teóricas sobre las que se fundamenta el RUP. De entre ellas cabe destacar que se trata de un proceso iterativo e incremental. Como se verá, el desarrollo del proyecto se hace en iteraciones, cada una de ellas conteniendo trabajo en varios flujos de trabajo. A su vez, las iteraciones se organizan en cuatro fases.

La sección tres trata de las fases y lo que se hace en cada una de ellas, así como de los productos (Artifacts) que se deben obtener al finalizarlas.

La sección cuatro es la más extensa y en ella se desarrollan los flujos de trabajo, sus objetivos y, nuevamente, los productos que deben obtenerse de ellos.

Para terminar, en la sección cinco obtenemos algunas conclusiones y en la sección seis se encuentra la bibliografía utilizada.

2. Bases teóricas

El Proceso Unificado de Rational es un proceso de ingeniería del software. Proporciona un acercamiento disciplinado a la asignación de tareas y responsabilidades en una organización de desarrollo. Su proposito es asegurar la producción de software de alta calidad que se ajuste a las necesidades de sus usuarios finales con unos costos y calendario predecibles. [1]

En definitiva el RUP es una metodología de esarrollo de software que intenta integrar todos los aspestos a tener en cuenta durante todo el ciclo de vida del software, con el objetivo de hacer abarcables tanto pequeños como grandes proyectos software. Ademas Rational proporciona herramientas para todos los pasos del desarrollo asi como documentación en linea para sus clientes.

Las características principales de RUP son:

- **Guiado/Manejado por casos de uso:** La razón de ser de un sistema software es servir a usuarios ya sean humanos u otros sistemas; un caso de uso es una facilidad que el software debe proveer a sus usuarios. Los casos de uso reemplazan la antigua especificación funcional tradicional y constituyen la guía fundamental establecida para las actividades a realizar durante todo el proceso de desarrollo incluyendo el diseño, la implementación y las pruebas del sistema.
- **Centrado en arquitectura:** La arquitectura involucra los elementos más significativos del sistema y está influenciada entre otros por plataformas software, sistemas operativos, manejadores de bases de datos, protocolos, consideraciones de desarrollo como sistemas heredados y requerimientos no funcionales. Es como una radiografía del sistema que estamos desarrollando, lo suficientemente completa como para que todos los implicados en el desarrollo tengan una idea clara de qué es lo que están construyendo, pero lo suficientemente simple como para que si quitamos algo una parte importante del sistema quede sin especificar. Se representa mediante varias vistas que se centran en aspectos concretos

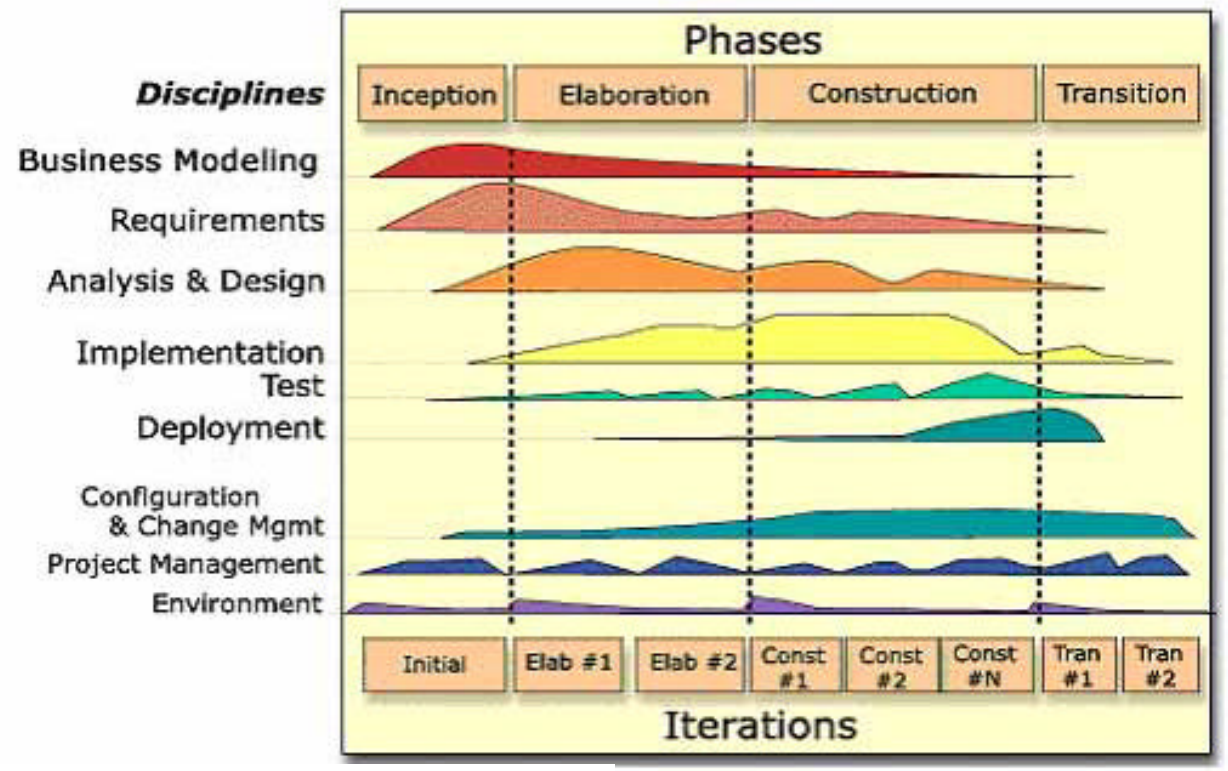

Figura 1: Fases, iteraciones y disciplinas

del sistema, abstrayéndose de lo demás. Todas las vistas juntas forman el llamado modelo 4+1 de la arquitectura, recibe este nombre porque lo forman las vistas lógica, de implementación, proceso y despliegue, más la de casos de uso que es la que da cohesión a todas. Como resumen de las mismas recomiendo dar un vistazo a la Figura 5-1 de la página 87 de [1].

- **Iterativo e Incremental:** Para hacer más manejable un proyecto se recomienda dividirlo en ciclos. Para cada ciclo se establecen fases de referencia, cada una de las cuales debe ser considerada como un miniproyecto cuyo núcleo fundamental está constituido por una o más iteraciones de las actividades principales básicas de cualquier proceso de desarrollo. En concreto RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en numero variable según el proyecto y en las que se hace un mayor o menor hincapié en los distintas actividades. En la Figura 1 tenemos un ejemplo de la distribución del trabajo.

Además de estas características principales según [5] cabe destacar las siguientes:

- **Desarrollo basado en componentes:** La creación de sistemas intensivos en software requiere dividir el sistema en componentes con interfaces bien definidas, que posteriormente serán ensamblados para generar el sistema. Esta característica en un proceso de desarrollo permite que el sistema se vaya creando a medida que se obtienen o que se desarrollan y maduran sus componentes.
- **Utilización de un único lenguaje de modelado:** UML es adoptado como único lenguaje de modelado para el desarrollo de todos los modelos.
- **Proceso Integrado:** Se establece una estructura que abarque los ciclos, fases, flujos de trabajo, mitigación de riesgos, control de calidad, gestión del proyecto y control de configuración; el proceso unificado establece una estructura que integra todas estas facetas. Además esta estructura cubre a los vendedores y desarrolladores de herramientas para soportar la automatización del proceso, soportar flujos individuales de trabajo, para construir los diferentes modelos e integrar el trabajo a través del ciclo de vida y a través de todos los modelos.

La estructura estática del proceso unificado se define en base a cuatro elementos, que son: los roles(antes workers), que responde a la pregunta ¿quién?, las actividades (activities), que responden a la pregunta ¿cómo?, los

productos (artifacts), que responden a la pregunta ¿qué?, y los flujos de trabajo (workflows), que responden a la pregunta ¿cuándo?. La definición de estos términos que se nos hace en [8] es:

- **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. En la Figura 2 se puede observar la relación entre los tres conceptos.
- **Actividades:** Una actividad de un trabajador en concreto es una unidad de trabajo que una persona que desempeñe ese rol puede ser solicitado a que realice. Las actividades tienen un objetivo concreto, normalmente expresado en terminos de crear o actualizar algún producto.
- **Productos:** Un producto o artefacto es un trozo de información que es producido, modificado o usado por un proceso. Los productos son los resultados tangibles del proyecto, las cosas que va creando y usando hasta obtener el producto final.

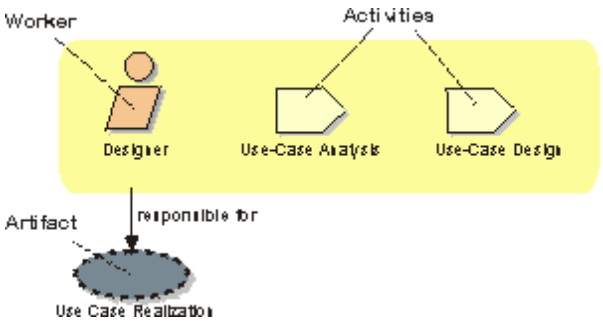

Figura 2: Roles, actividades y artefactos.

- Flujos de trabajo: La mera enumeración de rolos, actividades y artefactos no define un proceso, necesitamos definir la secuencia de actividades realizadas por los diferentes roles, así como la relación entre los mismos, que nos producen unos resultados observables. El RUP define varios flujos de trabajo distintos, entre los que distingue entre dos grupos, los de proceso, y los de apollo. Las distintas iteraciones a realizar consistirá en la ejecución de estos flujos de trabajo con una mayor o menos intensidad dependiendo de la fase e iteración en la que nos encontremos, como ya vimos un poco más arriba y que clarificabamos con la Figura 1.

En los apartados siguientes dada la naturaleza de este documento, daremos de lado a los roles, y trataremos las actividades de manera poco precisa, centrandonos en las generalidades de cada flujo de trabajo y los productos que tienen que dar como resultado.

3. Las fases

Como ya se ha visto en el apartado anterior, el RUP se divide en cuatro fases, las cuales pasaremos a ver con más detalle.

En la tabla 1 encontramos un resumen de los principales productos de RUP y en que momento deben iniciarse y terminarse. Para estos productos y otros existen plantillas pregeneradas en [7]. La tabla resumen de [6] también proporciona una buena visión de conjunto de lo que hay que hacer en RUP. Por último en la página 41 de [1] está la figura 3-3 que muestra las relaciones de los productos de la tabla 1.

3.1 Inicio

Antes de iniciar un proyecto es conveniente plantearse algunas cuestiones: ¿Cuál es el objetivo? ¿Es factible? ¿Lo construimos o lo compramos? ¿Cuánto va a costar? La fase de inicio trata de responder a estas preguntas y a otras más. Sin embargo no pretendemos una estimación precisa o la captura de todos los requisitos. Más bien se trata de explorar el problema lo justo para decidir si vamos a continuar o a dejarlo, ver [4]. Generalmente no debe durar mucho más de una semana.

Como dice en [1], los objetivos 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 coste en recursos y tiempo de todo el proyecto.
- Estimar los riesgos, las fuentes de incertidumbre.

Los productos de la fase de inicio deben ser:

- Visión del negocio: Describe los objetivos y restricciones a alto nivel.
- Modelo de casos de uso.

Flujo	Productos	Inicio	Elaboración	Construcción	Transición
Administración del Proyecto	Plan de desarrollo	I	R	R	R
	Caso de negocio	I			
	Lista de riesgos	I	R	R	R
Requisitos	Modelo de casos de uso	I	R		
	Vision	I	R		
	Especicifación adicional	I	R		
	Glosario	I	R		
Análisis y Diseño	Modelo de diseño		I	R	
	Documentación de la arquitectura SW		I		
Implementación	Modelo de implementación		I	R	R
Test	Plan de test		I	R	
Despliegue	Plan de despliegue				I

Tabla 1. Principales productos en RUP. I = inicio, R = refinamiento.

- Especificación adicional: requisitos no funcionales.
- Glosario: Terminología clave del dominio.
- Lista de riesgos y planes de contingencia.
- El caso de negocio (business case). Para más detalles ver el flujo de modelado del negocio.
- Prototipos exploratorios para probar conceptos o la arquitectura candidata.
- Plan de iteración para la primera iteración de la fase de elaboración.
- Plan de fases.

No todos los productos son obligatorios, ni deben completarse al 100%, hay que tener en cuenta el objetivo de la fase de inicio.

En [4] encontramos los síntomas de que no se ha entendido la fase de incio:

- Dura más de unas pocas semanas.
- Se intentan definir todos los requisitos.
- Se espera que las estimaciones o los planes sean muy precisos.
- Definir la arquitectura completamente, en lugar de refinarla en la fase de elaboración.
- No se definen el caso de negocio o la visión.
- Los nombres de la mayoría de los casos de uso o actores no se han definido.
- Todos los casos de uso se escriben con detalle.

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.
- Entiendimiento los requisitos, evidenciado por la fidelidad de los casos de uso principales.
- Las estimaciones de tiempo, coste y riesgo son creibles.
- 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.

3.2 Elaboración

Cómo se indica en el capítulo 5 de [1], 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 elimirar los mayores riesgos.

Cuando termina esta fase se llega al punto de no retorno del proyecto: a partir de ese momento pasamos de las relativamente ligeras y de poco riesgo dos primeras fases, a afrontar la fase de construcción, costosa y arriesgada. Es por esto que la fase de elaboración es de gran importancia.

En esta fase se construye un prototipo de la arquitectura, que debe evolucionar en iteraciones sucesivas hasta convertirse en el sistema final. Este prototipo debe contener los casos de uso críticos identificados en la fase de

incicio. También debe demostrarse que se han evitado los riesgos más graves, bien con este prototipo, bien con otros de usar y tirar.
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. 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 productos:

- 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.
- 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 espicifica el proceso a seguir.
- Posiblemente un manual de usuario preliminar.

La forma de aproximarse a esta fase debe ser 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 todas los productos de la fase de inicio el glosario, el caso de negocio, el ROI (Return Of Invest), etcétera.
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 creibles.
- 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.

3.3 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 todas los componentes, características y requisitos, que no lo hayan sido hecho hasta ahora, han de ser implementados, integrados y testeados, obteniendose una versión del producto que se pueda poner en manos de los usuarios(una versión beta).
El énfasis en esta fase se pone controlar las operaciones realizadas, administrando los recursos eficientemente, de tal forma que se optimicen los costes, los calendarios y la calidad.
Los objetivos concretos según [1] 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 rapido como sea practico.
- Conseguir versiones funcionales (alfa, beta, y otras versiones de prueba) tan rapido coo sea practico.

Los productos de la fase de construccion según [5] 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

3.4 Transición

La finalidad de la fase de transición es poner el producto en manos de los usuarios finales, para lo que tipicamente se requerirá 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 usabilidad del producto.

En concreto en [1] se citán algunas de las cosas que puede incluir esta fase:

- Testeo de la versión Beta para validar el nuevo sistema frente a las espectativas de los usuarios.
- Funcionamiento paralelo con los sistemas legados que estan siendo sustituidos por nuestro proyecto.
- Conversión de las bases de datos operacionales.
- Entrenamiento de los usuarios y tecnicos 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 al usuario.

Los productos de la fase de transición según [5] son:

- 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. Las actividades a realizar durante las iteraciones dependerán de su finalidad, si es corregir algún error detectado, normalmente será suficiente con llevar a cabo los flujos de trabajo de implementación y test, sin embargo, si se deben añadir nuevas características, la iteración será similar a la de una iteración de la fase de construcción.

La complejidad de esta fase depende totalmente de la naturaleza del proyecto, de su alcance y de la organización en la que deba implantarse.

4. Los flujos de trabajo

En RUP se definen nueve flujos de trabajo distintos, separados en dos grupos.

Los flujos de trabajo de 'ingeniería' son los siguientes:

- Modelado del negocio.
- Requisitos.
- Análisis y diseño.
- Implementación.
- Test.
- Despliegue.

Los flujos de trabajo de apoyo son:

- Administración del proyecto.
- Configuración y control de cambios.
- Entorno.

Aunque los nombres de los flujos de trabajo de ingeniería recuerden a las etapas de una metodología en cascada, en RUP como ya hemos visto las fases son distintas, y estos flujos de trabajo serán visitados una y otra vez a lo largo de todo el proceso.

En algunos flujos se explican productos importantes. Para ver como usar las plantillas ver el anexo 1. El ejmplo 1 se puede obtener en [11] y el 2 en [12].

4.1 Administración del Proyecto

El objetivo de la administración de un proyecto es conseguir equilibrar el completar los objetivos, administrar el riesgo y superar las restricciones para desarrollar un producto que sea acorde a los requisitos de los usuarios. Como se ve en [3], para conseguir esto el flujo de trabajo se centra en tres aspectos:

- Planificar un proyecto iterativo y cada iteración particular.
- Administrar el riesgo.
- Monitorizar el progreso del proyecto a través de métricas.

La planificación de un proyecto debe acometerse en dos niveles de abstracción: un plan de "grano grueso" para las fases y un plan de "grano fino" para cada iteración.

El plan de desarrollo (o plan de fases) debe contener las fechas esperadas para los hitos principales, cuándo se tendrá la arquitectura, cuándo estará la primera versión beta, etcétera. Estas fechas coincidirán, generalmente, con el final de las fases. También debería tener una previsión de las necesidades de personal y medios, así como fechas de hitos menores, sólo si se conocen. Este plan debe obtenerse temprano en la fase de inicio y no debe ir más allá de una o dos páginas. Debe actualizarse siempre que sea necesario.

Debe realizarse un plan de iteración por cada iteración, como cabría suponer. Este plan se elabora hacia la segunda mitad de la iteración, lo que significa que en un momento dado habrá dos planes activos: el de la iteración en curso y el de la próxima, que es construido en ésta. En este plan se detallarán fechas importantes para la iteración: compilaciones importantes, revisiones o llegada de componentes.

La administración del riesgo consiste en ocuparse de las incógnitas de un proyecto, las cuestiones que pueden llevarlo a pique. En concreto hay que identificar los riesgos, típicamente en la fase de inicio, y hacerles frente. Para ello trataremos de evitarlos, transferirlos (leáse quitarnoslos de encima) o asumirlos. En este último caso habrá que tratar de mitigar el riesgo y definir un plan de contingencia por si el riesgo se convierte en un problema real. En definitiva la administración del riesgo consistirá en gestionar una lista de riesgos.
Monitorizar un proyecto es importante para mantenerlo bajo control. Tenemos que "medir" para ver como de bien nos ajustamos a los planes, la calidad requerida y los requisitos. También es necesario para planificar de forma precisa y ver cuál es el comportamiento del proyecto frente a cambios. Tomar métricas no es gratis, así que hay que justificar por qué se mide.
En este flujo de trabajo también se obtiene el caso de negocio (business case). Consiste en el contexto del negocio, criterios de éxito del proyecto (como por ejemplo, ser pagados) y una previsión de financiera (gastos, salarios, etcétera). Si esperamos vender el sistema, también tendrá que haber una aproximación a los beneficios que obtendremos: el ROI (Return Of Investment).

✓ **Producto**: Plan de desarrollo.
Ya hemos visto cuales son los objetivos de este producto. Ahora veremos con detalle uno, así como las partes de que se compone la plantilla.
La sección 1 se comenta en el anexo 1. La sección 2 pretende proporcionar una breve visión de conjunto del proyecto, sus objetivos, restricciones, los entregables que se van a producir (los productos) y que evolución se espera.
La sección 3 trata de la organización de la gente que desarrollará el proyecto y sus interacciones con el exterior. En el ejemplo se consideran cuatro roles: gestor del proyecto, analistas, desarrolladores y testeadores.
La sección 4 es la más importante, trata de la gestión del proyecto propiamente. En el ejemplo vemos como se ha obtenido un plan de fases en el que se indica el número de iteraciones previstas. Después se detalla que se va a hacer en cada fase y los hitos que se espera obtener.
Como se puede ver el ejemplo carece de las secciones 5, 6 y 7. Esto se debe a que contendrían planes para proyectos de gran envergadura, como aseguramiento de la calidad, infraestructuras, etcétera.

✓ **Producto**: Plan de iteración.
Este procducto trata la planificación de grano fino. La sección 2 contiene el plan. En el ejemplo 1 se refieren a unos diagramas de Gantt creados con Microsoft Project. Estos diagramas dan una previsión detallada del tiempo asignado a cada tarea. Una forma menos formal de planificar sería poner sólo la fecha de fin, como en el ejemplo 2.
En la sección 3 se reseñan todos los recursos humanos, financieros o de otra índole necesarios para completar la iteración.
La sección 4 es una lista de los casos de uso implicados en esta iteración. Si se desea puede justificarse su elección.
Por último está la sección 5 con los criterios de evaluación de la iteración. En los ejemplos no está la sección, mal hecho por su parte.

✓ **Producto**: Lista de Riesgos.
Este producto sólo tiene una sección además de la primera: la lista de riesgos propiamente dicha. Para cada uno hay que indicar: su magnitud, una descripción, su impacto (donde afecta), indicadores para monitorizarlo, una estrategia para mitigarlo y el plan de contingencia por si el riesgo se hace real.
En los ejemplos se ven dos formas expresar cada riesgo: en forma de tablas o con texto puro. Que cada cual elija la que la más le guste, pero si se va a hacer algo raro, conviene explicarlo antes, como en el ejemplo 1.

4.2 Modelado del negocio
Con este flujo de trabajo pretendemos llegar a un mejor entendimiento de la organización donde vamos a implantar nuestro producto. Los principales motivos para esto son los siguientes: asegurarnos de que el producto será algo útil, no un obstáculo; conseguir que encaje de la mejor forma posible en la organización; y tener un marco común para los desarrolladores, los clientes y los usuarios finales. Este flujo de trabajo no será siempre necesario. Si sólo añadimos funcionalidad que no verán los usuarios directamente, no hará falta.
Para modelar el negocio usaremos las mismas técnicas que para modelar software, facilitando que ambas partes entiendan los modelos. En

concreto tendremos *casos de uso de negocio*, *actores de negocio*, etcétera. Los diagramas también tendrán su equivalente *de negocio*.
Dependiendo del tipo de software que estemos construyendo, el modelado del negocio cambiará. No se trata de modelar la organización de arriba abajo, sólo la parte que nos toca. En concreto en [1] se pueden encontrar seis escenarios para escoger el más apropiado o hacernos una idea de qué hay que modelar.
En la fase de inicio habrá que hacer una valoración del negocio. Tras hacerla decidiremos cómo vamos ha hacer el modelado del negocio: por ejemplo, ver en que escenario de los seis estamos.
Como ya dije usaremos una extensión de UML para modelar el negocio. En concreto habrá modelos de casos de uso de negocio y modelos de objetos de negocio. Con los primeros capturaremos QUÉ se hace y QUIÉN lo hace. Con los segundos veremos CÓMO se hace.
Una gran ventaja de ésta aproximación al modelado del negocio es que es un forma clara y concisa de mostrar las dependencias entre el negocio y el sistema que estamos construyendo.

4.3 Requisitos

Este es uno de los flujos de trabajo más importantes, porque en él se establece QUÉ es lo que tiene que hacer exactamente el sistema que construyamos. En esta línea los requisitos son el contrato que debemos cumplir, de modo que los usuarios finales tienen que comprender y aceptar los requisitos que especifiquemos.
Tal como indica [2], los requisitos se dividen en dos grupos. Los requisitos funcionales son las cosas que el sistema puede hacer, su funcionalidad. Se modelan mediante diagramas de casos de uso. Los requisitos no funcionales representan aquellos atributos que debe exhibir el sistema, pero que no son una funcionalidad específica. Por ejemplo requisitos de usabilidad, fiabilidad, eficiencia, portabilidad, etcétera.
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 usabilidad, se diseña la interfaz gráfica de usuario. Para ello habitualmente se construyen prototipos de la GUI que se contrastan con el usuario final. Además ésta es una buena forma de explicitar requisitos (Usuario: "¿Dónde está el botón de calcular la sobrecarga de TLA?" Analista: "¿La qué?").
En definitiva, en este flujo de trabajo hay que analizar el problema, comprender las necesidades de los interesados y expresarlas en forma de requisitos; construir diagramas de casos de uso para los requisitos funcionales, los no funcionales describirlos textualmente en especificaciones suplementarias. Además hay que gestionar los cambios en los requisitos a lo largo de todo el proceso.
Dentro de éste flujo, en la fase de inicio hay que crear la Visión. Se trata de un documento que de una vista general del núcleo de los requisitos del proyecto, características clave y restricciones principales.
Algunos dominios de negocio pueden ser algo enrevesados al principio, por ejemplo si hay que crear una aplicación para una base aérea. Por este motivo puede ser de gran ayuda construir un Glosario que recoja la terminología usada a lo largo del proyecto o la organización.

- ✓ **Producto**: Modelo de Casos de Uso.

Este producto se obtiene con la plantilla de Especificación de Casos de Uso. En teoría habría que hacer un documento por caso de uso, y así puede hacerse. Sin embargo en los ejemplos se optó por fusionar todos los casos de uso en un documento.
Por cada caso de uso hay que dar una pequeña descripción. A continuación hay que describir el flujo de eventos del caso. Primero se destaca el flujo principal y después vienen los alternativos. Si una alternativa es simple, puede ponerse con el flujo principal. Si un caso es complejo, puede ponerse figuras explicativas, diagramas UML o lo que se necesite.
Lo siguiente es poner que requisitos especiales hay, si los hay. Luego vienen las precondiciones y las postcondiciones. Finalmente los puntos de extensión.
Si se opta por juntar todas las especificaciones de casos de uso convendrá hacer una primera sección como en el ejemplo 1, describiendo los actores, las relaciones de los casos de uso y mostrando los diagramas.

- ✓ **Producto**: Especificación Adicional.

En este producto se especifican todos los requisitos no funcionales de nuestro sistema. Como ya se dijo, se trata de atributos que no dan funcionalidad, por ejemplo lo fácil que es de usar o si cumple con una normativa concreta.

La plantilla para este producto está dividida en secciones que indican distintos tipos de requisitos no funcionales. Nosotros tendremos que ir simplemente colocando nuestros requisitos en la sección adecuada y borrar las que no vayamos a usar. Del mismo modo podemos crear secciones nuevas si las necesitamos.
En el ejemplo 2 se optó por obviar esta clasificación y poner todos los requisitos juntos. En mi opinión las secciones añaden claridad y sirven de recordatorio para no dejarnos nada, pero si hay pocos requisitos, pueden ser un estorbo.

✓ **Producto**: Visión.
El propósito de la visión es reunir, analizar y definir necesidades y características del sistema a alto nivel.
La sección 2 pretende posicionar el problema que da lugar al sistema. Para ello se describe el problema, que oportunidad para hacer negocio hay y que lugar en el mercado ocupará el sistema como producto. Si no pretendemos vender nada, bastará con describir el problema, como en el ejemplo 2.
La sección 3 pretende dar a conocer todos los interesados en el sistema y los usuarios finales. Hay que indicar los problemas que cada uno ve que deben ser resueltos. No hay que poner las peticiones concretas sino el trasfondo de cada interesado. Si el producto se va a vender, hay que poner un estudio de la población objetivo. El ejemplo puede aclarar bastante este punto.
Las secciones 4 y 5 dan una visión de conjunto del producto y sus capacidades. Algunos puntos de la plantilla sólo son apropiados si se pretende vender el producto.
Las secciones restantes dan un mayor detalle sobre el producto. En principio, para un proyecto pequeño, podrían no ser necesarios tantos apartados si se puede escribir un texto breve y claro que describa lo mismo. Sin embargo es buena idea mirarse todas las secciones para no olvidar nada.

✓ **Producto**: Glosario.
En el glosario se recoge el vocabulario propio del dominio del sistema, y que dependiendo del proyecto pueden ser términos muy especializados. Además puede usarse para definir un diccionario informal de tipos de datos. El núcleo de este producto es la sección 2 donde se va introduciendo a modo de diccionario, y normalmente por orden alfabético, la definición de los diferentes conceptos. En las sección 3 se definen aquellos estereotipos(especificaciones de UML) que no son los predefinidos por RUP o UML y que pueden o deben ser usados en el proyecto. Ni el ejemplo 1 ni el 2 han hecho uso de este último apartado.

4.4 Análisis y diseño
El objetivo de este flujo de trabajo es traducir los requisitos a una especificación que describe cómo implementar el sistema.
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.
Como se dice en [1] el diseño debe ser suficiente para que el sistema pueda ser implementado sin ambigüedades. De hecho, cuando la precisión del diseño es muy grande, la implementación puede ser hecha por un generador automático de código.
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 software, que captura varias visiones arquitectónicas del sistema.

✓ **Producto**: Modelo de Diseño.
No se dispone de una plantilla para este producto. Es sencillamente la estructuración de los distintos diagramas y modelos que se tengan referentes a la parte de diseño del sistema. En el ejemplo 1 la información se ha estructurado en cuatro apartados, en el primero se muestran los paquetes que forman el sistema y sus

relaciones, en el segundo se muestra lo que contiene cada paquete, y en el tercero y cuarto se muestra una vista lógica del sistema, general en el tercer apartado y detallada en el cuarto, mediante diagramas de clases de diseño.

- ✓ **Producto**: Documentación de la Arquitectura Software.

En este documento se da una descripción de la arquitectura del sistema, véase el apartado de características principales del RUP en el apartado de *Bases teóricas* de este trabajo. En el apartado 2 del documento se anticipan las vistas que van a ser necesarias para la descripción, así como la representación escogida para las mismas. En la sección 3 se describen los requerimientos y objetivos del sistema que sean de influyan en la arquitectura del mismo. A partir de aquí viene un apartado por cada vista (Casos de uso**,** lógica**,** proceso**,** despliegue, e implementación) que se incluya en el documento. Así como una vista optativa adicional del almacenamiento de los datos persistentes. Se termina con una sección para describir los requerimientos en tiempo y espacio, y otra con una explicación de cómo contribuye la arquitectura a garantizar la calidad del software.

4.5 Implementación

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

En cada iteración habrá que hacer lo siguiente:

- Planear que 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 testean los subsitemas 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 evolucionarios. Estos últimos llegan a transformarse en el sistema final.

- ✓ **Producto**: Modelo de implementación.

La plantilla para este producto no la proporciona UPEDU (ver anexo I), que consiste en una visión general lo que tiene que ser implementado, y un apartado para cada iteración (que coincide con la plantilla de RUP del plan de integración de constructos) con los componentes y subsistemas a implementar durante esa iteración, así como de los resultado software (builds) que se han de obtener y el testeo que se ha de realizar sobre ellos (para lo que se puede hacer referencia al plan de test).

4.6 Test

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. Como se dice en [1] : "El papel del testeo no es asegurar la calidad, pero sí evaluarla, y proporcionar una realimentación a tiempo, de forma que las cuestiones de calidad puedan resolverse de manera efectiva en tiempo y coste. "

Los principales aspectos a ser evaluados en un producto software son la Fiabilidad (resistente a fallos), la Funcionalidad (hace lo que debe) y el Rendimiento (lleva a cabo su trabajo de manera efectiva). Los tests pueden hacerse a diferentes niveles dependiendo del objetivo de los mismos, a saber: Test de unidad (se testean las unidades mínimas por separado, y normalmente se hace durante la implementación misma), de integración (varias unidades juntas), de sistema (sobre la aplicación o sistema completo) y de aceptación (realizado sobre el sistema global por los usuarios o terceros). Dentro de cada uno de estos niveles podemos distinguir distintos tipos de test.

A la representación de lo que será testeado y cómo debe de hacerse es a lo que se le llama el modelo de test. Incluye la colección de casos de test, procedimientos de test, scripts, resultados esperados...

Las actividades de este flujo comienzan pronto en el proyecto con el plan de test (el cual contiene información sobre los objetivos generales y específicos del testeo 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 testear, 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.

✓ **Producto**: Plan de Test.

El RUP diferencia entre un Plan de Test global, donde se describen los objetivos y mecanismos que se van a utilizar para el proyecto en general, así como un plan de test especifico para cada iteración donde se especifica que elementos se deben testear, cuales son los objetivos que se persiguen con esas pruebas, y la aproximación a utilizar para conseguir esos objetivos. Incluyendo también una estimación de los recursos necesarios para llevarlos a cabo.

4.7 Configuración y gestión de cambios

La finalidad de este flujo de trabajo es mantener la integridad de todos los artefactos que se crean en el proceso, asi como de mantener información del proceso evolutivo que han seguido.
Las causas por las que la evolución de los artefactos puede causar problemas según [8] son:

- **Actualización simultanea**: Se da cuando dos personas trabajan por separado sobre el mismo artefacto a la vez, el último en hacer las modificaciones sobreescribe lo hecho por el primero.
- **Notificación limitada**: Cuando un problema ha sido resuelto en un artefacto compartido por varios roles y algunos de ellos no son notificados del cambio.
- **Multiples versiones**: Cuando se trabaja con diferentes versiones del producto al mismo tiempo en diferentes flujos de trabajo, pueden surgir problemas si los cambios no son convenientemente monitorizados y propagados.

La configuración y gestión de cambios cubre tres funciones interdependientes:

- La **gestión de la configuración**, que maneja la estructura del producto, la identificación de los elementos, configuraciones validas de los mismos versiones, versiones y espacios de trabajo.
- **Gestión de las peticiones de cambio**, que coordina el proceso de modifivar artefactos de una manera consistente.
- **Métricas y status,** que se encarga de extraer información para la correcta administración del proyecto de las herramientas que soportan las dos funciones anteriores.

4.8 Entorno

La finalidad de este workflow es dar soporte al proyecto con las adecuadas herramientas, procesos y metodos. Es decir tener a punto las herramientas que se vayan a necesitar en cada momento, así como definir la instancia concreta de proceso unificado 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.
- Sercicios técnicos.

El principal artefacto que se usa en este flujo de trabajo es el *caso de desarrollo* que especifica para el proyecto actual en concreto, como se aplicará el proceso unificado, que productos se van a utilizar y como van a ser utiliados. Ademas se tendrán que definir las lineas guia (los pasos concretos y políticas a seguir) para los distintos aspectos del proceso, como pueden ser el modelado del negocio y los casos de uso, para la interfaz de usuario, el diseño, la programación, el manual de usuario, ...
Las actividades que se deben llevar a cabo durante este flujo de trabajo según [6] son:

- Preparar el entorno para el trabajo.
- Preparar el entorno para una iteración.
- Preprarar las lineas de guia para una iteración.
- Dar soporte al entorno durante la iteración.

4.9 Despliegue

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

- Testear el producto en su entorno de ejecución final.
- Empaquetar el software para su distribución.
- Distribuir el software.
- Instalar el sofware.
- 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 proposito tanto del flujo como de la fase es asegurar una aceptación y adaptación sin complicaciones del software por parte de los usuarios. Aunque la ejecución de este flujo de trabajo debe empiezar en fases anteriores, para preparar el camino, sobre todo con actividades de planificación, pero tambien con la elaboración del manual de usuario, tutoriales, ... Dado el amplio rango de aplicaciones que se pueden dar y sus diversas características los productos necesitados por este flujo de trabajo puede variar en gran medida. Aunque el artefacto clave es una distribución (release) del producto, que en general puede consistir de:

- Software ejecutable (en todos los casos).
- Productos de instalación: scripts, herramientas, archivos, guias, información sobre licencia, ...
- Notas de la distribución, describiendola al usuario final.
- Material de apoyo, como pueden ser los manuales de usuario, de operaciones y mantenimiento.
- Materiales formativos.

A diferencia de otros flujos de trabajo RUP da un detalle menor al despliegue, debido a la ya citada diversidady especificidad de cada proyecto.

5. Conclusiones

En este documento se ha dado una visión general de lo que es el RUP, así como de la estructura bidimensional que sigue, dividiendo el proceso en fases, y estas en flujos de trabajo. Se han dado apuntes de lo que se espera de cada fase así como la forma de manejar los flujos de trabajo.

Para aplicar el RUP en pequeños equipos y proyectos se deberá mapear los diferentes roles (a los que no hemos dado especial relevancia en este documento) entre los distintos miembros del equipo, pero la diferencia clave con un proyecto de mayor envergadura, según [9], es el grado de formalidad a la hora de usar los distintos artefactos, planes del proyecto, requisitos, clases, ... En [10] se pueden encontrar consejos sobre que artefactos utilizar y cómo hacerlo.

El RUP es una metodología completa y extensa que intenta abarcar todo el mundo del desarrollo software, tanto para pequeños proyectos, como proyectos más ambiciosos de varios años de duración. Por lo que existe una gran cantidad de documentación sobre el mismo, tanto en libros como en la red, eso sí en ingles. Es sin embargo difícil empezar a aplicar esta metodología en una organización. Por eso esperamos que este documento sirva tanto para familiarizar con el Proceso Unificado a aquellos que no lo conocían, así como de servir de guía durante la ejecución del mismo.

5. Referencias

[1]Philippe Kruchten, *The Rational Unified Process An Introduction*, Addison Wesley, 2001.

[2]http://www.rational.com/media/whitepapers/apprmuc.pdf Roger O., Leslee P., Maria E., *Applying Requirements Management with Use Case.*

[3]http://www.therationaledge.com/content/oct_02/f_iterativePlanning_pk.jsp Philippe Kruchten, *Planning an Iterative Project.*

[4]http://www.asia.nust.ac.th/~mukdaprp/fall2002/is008/lectures/inception.pdf Pat Mukdaprakorn, *Inception Phase*, Asian University of Science and Technology.

[5]http://atenea.ucauca.edu.co/~gramirez/archivos/AnotacionesRUP.pdf Ramírez González, Gustavo A., *Laboratorio III de Electrónica, Anotaciones RUP*, 2001.

[6]http://davidfrico.com/rup-slc.pdf Rational Unified Process Software Life Cycle Una tabla resumen de los flujos, trabajadores y productos.

[7]http://www.public.iastate.edu/~shaf2/cs362docs/Templates/rup_wd_tmpl.zip Plantillas de productos de RUP

[8]Rational White Paper, *Best Practices for Software Development Teams*, 1998

[9] http://www.rational.com/products/rup/faq.jsp FAQ sobre RUP.
[10] http://www.yoopeedoo.com/upedu/, Pagina web de UPEDU (Unified Process for EDUcation).
[11] http://www.yoopeedoo.com/upedu/process/artifact/tmpl_cs/ovu_tmplcs.htm Ejemplo 1.
[12] http://www.public.iastate.edu/~shaf2/cs362_main.htm Ejemplo 2.

Anexo I

El objetivo de este anexo es servir de guía para la instalación y uso de las plantillas de los productos de RUP.

Paso 1. Obtener las plantillas.
Pueden conseguirse en esta dirección:
http://www.public.iastate.edu/~shaf2/cs362docs/Templates/rup_wd_tmpl.zip

En ésta otra dirección se puede conseguir otras plantillas un poco más simples y que ya han sido parcialmente completadas. Desafortunadamente no son plantillas de RUP, sino de una variante llamada UPEDU. En esencia es lo mismo, pero más simplificado y orientado al mundo de la educación. Para las prácticas servirán igual o mejor, de hecho con ellas se hizo el ejemplo 1.
http://www.yoopeedoo.com/upedu/process/artifact/tmpl_cs/wrdtmpl/upedu_wd_tmpl.zip

Paso 2. Instalación.
Descomprime el archivo ZIP. Arranca el Microsoft Word, ve a Herramientas, Opciones, Ubicación de Archivos. Anota la dirección de Plantillas Personales o modifícala según te convenga. En esa ubicación copia la carpeta con las plantillas y llámala "Plantillas RUP".

Paso 3. Uso.
Arranca el Microsoft Word, ve a Archivo, Nuevo… Elige Nuevo a partir de una plantilla. En el cuadro que aparezca habrá una pestaña llamada Plantillas RUP. Púlsala y elige una plantilla.
Cuando tengas la plantilla abierta tendrás que cambiar las variables como el nombre de proyecto. Para ello pulsa Archivo, Propiedades. En la pestaña Resumen modifica los campos según te interese. Pulsa Aceptar. Para actualizar los campos selecciona todo (Edición, Seleccionar Todo) y pulsa F9 (equivale a Actualizar Campos).
Ahora sólo queda rellenar la plantilla. Cada una contiene instrucciones (en inglés) sobre su uso. También es conveniente leerse este documento entero.

Rellenando las plantillas

Todas las plantillas tienen la sección 1 en común, que pasaremos a estudiar con detenimiento. Es conveniente que aunque las plantillas estén en inglés, los documentos los generemos en castellano.

Revision History (Historia de Revisiones)
Se trata de una tabla que contiene los distintos cambios que se han realizado sobre el documento. En concreto hay que poner la fecha, a que versión del sistema corresponde el cambio, una muy breve descripción del cambio y el o los autores. Está en todos los ejemplos, así que míralos para captar el espíritu de la tabla.

Table of Contents (Tabla de Contenidos)
Índice del documento. Se rellena automáticamente, no tocar. Para actualizarla, haz clic con el botón derecho y pulsa Actualizar Campos.

Introduction (Introducción)
Todos los documentos tienen una introducción para indicar para qué sirven. Cada plantilla la trae casi completamente hecha, en algunos casos sólo hay que sustituir el nombre del proyecto.

Purpose (Propósito)
El propósito del documento. Si no sabes para que sirve, más vale que no lo hagas.

Scope (Ámbito)

El área que cubre el documento. Por ejemplo un glosario cubre las palabras extrañas, la lista de riesgos se extiende por toda la vida del proyecto y el modelo de diseño abarca todo el sistema. Puede ser un lío diferenciar que poner en la introducción, el propósito o el ámbito. Si crees que todo queda dicho en un punto o dos, puedes saltarte tranquilamente el resto.

Definitions, Acronyms, and Abbreviations (Definiciones, Acrónimos y Abreviaturas)
Se trata de un pequeño glosario específico de este documento. Si vas a poner muchas definiciones, pon simplemente una referencia al glosario general en lugar de repetirlo casi entero aquí.

References (Referencias)
Lista de documentos que son referenciados en éste.

Overview (Visión de conjunto)
Resumen del resto del documento. Pon aquí cualquier cosa que quede por decir sobre el documento y no esté en los puntos anteriores. Si crees que todo está claro, sáltatelo.