RedAbogacía
ABOGACÍA ESPAÑOLA

Metodología de desarrollo de IT CGAE

MEDRA

ITCGAE_MetodologíaDesarrollo.docx *Uso Interno*

RedAbogacía
ABOGACÍA ESPAÑOLA

CONTROL DE VERSIONES

Versión	Fecha	Autor	Descripción / Cambios Relevantes
1.00	13/11/2014	Gregorio Ferrero	Versión inicial
1.01	28/09/2015	Gregorio Ferrero	Se completan algunos aspectos de MEDRA^{C} Se incluye capítulo de gestión de pequeños mantenimientos Se incluye capítulo de organización y estructura de proyectos Se incluye anexo de herramientas de apoyo Se incluyen cuadros-resumen y guía rápida
1.02	29/09/2015	Gregorio Ferrero	Se incluyen los diagramas a alto nivel de MEDRA y MEDRA^{C} Se elimina el apartado *Comunicación del despliegue de nuevas versiones* que era erróneo
1.03	28/10/2015	Gregorio Ferrero	Se incluyen cambios tras revisión conjunta por parte del área de Desarrollo. Se cambia RedMine por JIRA como herramienta propuesta para gestión de tareas.
1.04	18/08/2016	Gregorio Ferrero	Se añade detalle en la especificación de cambios menores Recomendación de pruebas con distintos perfiles Añadida buena práctica relativa a espacios de nombres
1.05	22/11/2016	Gregorio Ferrero	Se cambia la estructura de carpetas para aglutinar en una misma carpeta por cada S.I. los proyectos y su mantenimiento

RedAbogacía
ABOGACÍA ESPAÑOLA

ÍNDICE

1. Introducción

El Consejo General de la Abogacía Española (en adelante CGAE) es el órgano representativo, coordinador y ejecutivo superior de los colegios de abogados de España y tiene, a todos los efectos, la condición de corporación de derecho público, con personalidad jurídica propia y plena capacidad para el cumplimiento de sus fines.

Infraestructura Tecnológica del CGAE (en adelante IT CGAE) es una sociedad limitada unipersonal (SLU) cuyo accionista único es el CGAE, creada para la prestación de servicios tecnológicos al CGAE y a los colegios de abogados, y que gestiona a través de encomiendas de ejecución los proyectos y activos tecnológicos del CGAE. IT CGAE opera bajo la marca RedAbogacía, por lo que en el presente documento puede hacerse referencia a IT CGAE y RedAbogacía de manera indistinta.

Muchos de esos proyectos consisten en el desarrollo de productos *software* que presten servicio al CGAE, a los Ilustres Colegios de Abogados (ICAs) y a los propios letrados, por lo que se estima conveniente y necesario disponer de una metodología de trabajo que permita llevar a término dichos proyectos de desarrollo con unos niveles de calidad y control adecuados.

El presente documento contiene la descripción de dicha metodología, denominada **ME**todología de **D**esarrollo de **R**ed**A**bogacía (MEDRA).

1.1. Alcance

El alcance del presente documento cubre la metodología que se aplicará en RedAbogacía para los proyectos de desarrollo de sistemas o aplicaciones informáticas (*software*), tanto si estos son ejecutados por personal interno de IT CGAE como si son subcontratados a proveedores ajenos a la empresa.

Dicha metodología cubre todo el ciclo de desarrollo, desde el análisis inicial del proyecto hasta la entrega del producto *software* terminado.

1.2. Objetivos

Los objetivos principales de la metodología MEDRA son:

1. Homogeneizar la forma de trabajar del personal involucrado en el desarrollo de *software* para RedAbogacía, tanto si se trata de personal interno como subcontratado
2. Normalizar la documentación producida en el proceso de desarrollo de *software*
3. Alinear la funcionalidad del *software* desarrollado con las necesidades del cliente o usuario final
4. Minimizar el número de errores en el *software* desarrollado
5. Asegurar adecuados niveles de usabilidad, rendimiento, mantenibilidad y escalabilidad del *software* desarrollado

Todos estos objetivos se pueden resumir en uno: Garantizar la **calidad** del proceso de desarrollo de software y de sus productos.

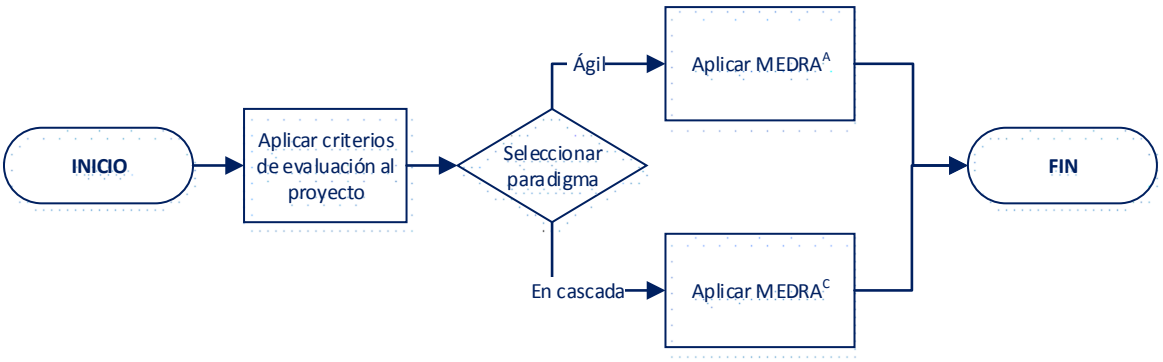
2. Descripción general

El presente apartado ofrece una descripción general de la metodología MEDRA, que será desarrollada en detalle en los apartados posteriores de este documento.

MEDRA pretende cubrir bajo un mismo marco metodológico dos tipos de proyectos: aquellos en los que la aproximación más adecuada sea un desarrollo "ágil" (más flexible y adaptado al cambio) y aquellos en los que sea más adecuado un enfoque más tradicional "en cascada" (más formalista y estricto).

Para los proyectos "ágiles" se aplicará un enfoque metodológico basado en SCRUM, mientras que para el resto de proyectos se aplicará un enfoque basado en Métrica v3. En ninguno de los casos MEDRA pretender ser una trasposición literal de estas metodologías, sino que adopta las líneas maestras de ambas y las adapta a la idiosincrasia de IT CGAE y sus clientes (CGAE, ICAs y abogados).

El siguiente diagrama ilustra el flujo de MEDRA a muy alto nivel:



En líneas generales, la metodología MEDRA contempla las siguientes etapas:

1. **Evaluación del paradigma metodológico**
   En esta etapa se analiza el proyecto y, en base a ciertos parámetros, se determina el paradigma metodológico más adecuado para el proyecto: ágil o en cascada. En base a este análisis se continúa por la etapa 2 ó 3.

2. **Paradigma ágil (MEDRA^{A}):**
   a. Composición del equipo de proyecto
   b. Definición del sistema
   c. Establecimiento de requisitos
   d. Diseño de alto nivel
   e. Planificación de *sprints*
   f. Iteración de *sprints* (x N)
      i. Planificación del *sprint*
      ii. Reunión diaria
      iii. Diseño técnico
      iv. Implementación de código
      v. Pruebas unitarias y de integración
      vi. Despliegue en Preproducción
      vii. Revisión del *sprint*
   g. Pruebas finales y de aceptación del sistema
   h. Elaboración de manuales y formación a usuarios
   i. Preparación de la entrega final

3. **Paradigma en cascada (MEDRA^{C}):**
    a. Análisis del Sistema de Información (ASI)
        i. Definición del sistema
        ii. Establecimiento de requisitos
        iii. Identificación de subsistemas de análisis
        iv. Análisis de los casos de uso
        v. Análisis de clases
        vi. Definición de interfaces de usuario
        vii. Especificación del plan de pruebas
    b. Diseño del Sistema de Información (DSI)
        i. Definición de la arquitectura del sistema
        ii. Diseño de casos de uso
        iii. Diseño de clases
        iv. Diseño físico de datos
        v. Generación de especificaciones de construcción
        vi. Diseño de la migración y carga inicial de datos
        vii. Especificación técnica del plan de pruebas
        viii. Establecimiento de requisitos de implantación
    c. Construcción del Sistema de Información (CSI)
        i. Preparación del entorno de generación y construcción
        ii. Implementación del código
        iii. Ejecución de las pruebas unitarias y de integración
        iv. Elaboración de los manuales de usuario
        v. Construcción de los componentes y procedimientos de migración y carga inicial de datos
    d. Implantación y Aceptación del Sistema (IAS)
        i. Implantación en entorno de preproducción
        ii. Pruebas de aceptación del sistema
        iii. Formación a usuarios
        iv. Preparación de la entrega final

Como ya se ha mencionado, los siguientes capítulos de este documento describen en detalle cada una de las etapas aquí mostradas.

Mención aparte merecen las tareas de mantenimiento de las aplicaciones y servicios que se encuentran en producción, cuando éstas se reducen a pequeños cambios que no pueden ser gestionados como auténticos "proyectos". Este tipo de cambios suelen corresponder, en su gran mayoría, a cambios correctivos que hay que aplicar al *software* en producción a la mayor brevedad posible. Por su pequeña dimensión (en esfuerzo de desarrollo) y su urgencia, no tiene sentido en estos casos aplicar todos los pasos que implica la metodología completa. En cualquier caso, para dar cobertura metodológica a este tipo de tareas, MEDRA incluye el capítulo 7: *Gestión de pequeños mantenimientos*.

2.1. Relación entre MEDRA y el Sistema Integrado de Gestión de IT CGAE

MEDRA es una metodología específica para el trabajo de desarrollo de *software*. Siempre que se realiza un trabajo de desarrollo su objetivo será uno de los siguientes:

a) Construir una nueva aplicación o servicio
b) Evolucionar una aplicación o servicio existente a una nueva versión
c) Aplicar un cambio correctivo o una pequeña mejora funcional a una aplicación o servicio existente

En cualquiera de estos casos, el objetivo es un **cambio** en una aplicación o servicio, o en el catálogo de aplicaciones y servicios de IT CGAE. Así, MEDRA encaja en el procedimiento de gestión de cambios estipulado para IT CGAE dentro de su Sistema Integrado de Gestión, que se alinea con la norma ISO 20.000 – Gestión del servicio de TI.

El manual del procedimiento de gestión de cambios para IT CGAE se encuentra recogido en el documento **PGS.11 Gestión de Cambios-VX.docx** (donde **X** representa la última versión liberada del documento).

Por tanto, cualquier cambio en un sistema *software* desarrollado por IT CGAE implicará la gestión de ese cambio según el procedimiento citado. Este procedimiento dicta la gestión del cambio a una escala corporativa, mientras que MEDRA baja al detalle de cómo debe trabajar el área de Desarrollo de IT CGAE a bajo nivel para implementar ese cambio. Así pues, deberá aplicarse tanto lo dictado por el procedimiento de gestión de cambios como lo dictado por MEDRA, con las siguientes consideraciones:

a) Si se va a construir una nueva aplicación o servicio:
   - Se creará una única petición de cambio en la herramienta de gestión (EasyVista) de tipo "Nuevo servicio"
   - En el campo "Nota" de la ficha del cambio en EasyVista se pondrá un enlace (ruta UNC) al documento de plan de proyecto, en el cual deberá estar especificado el alcance del nuevo sistema
   - Se construirá el sistema siguiendo la metodología MEDRA (capítulos 3 a 5 del presente documento)
   - La entrega del nuevo sistema se asociará a la petición de cambio creada

b) Si se va a evolucionar una aplicación o servicio existente a una nueva versión:
   - Se creará una única petición de cambio en la herramienta de gestión (EasyVista) de tipo "Modificación relevante de un servicio"
   - En el campo "Nota" de la ficha del cambio en EasyVista se pondrá un enlace (ruta UNC) al documento de plan de proyecto, en el cual deberá estar especificado el alcance de los desarrollos evolutivos a realizar
   - Se construirá la nueva versión del sistema siguiendo la metodología MEDRA (capítulos 3 a 5 del presente documento)
   - La entrega de la nueva versión se asociará a la petición de cambio creada

c) Si se va a aplicar un cambio correctivo (o varios) o una pequeña mejora funcional (o varias) a una aplicación o servicio existente:
   - Se creará una petición de cambio en la herramienta de gestión (EasyVista) por cada cambio funcional o correctivo que se vaya a realizar. El tipo de petición de cambio a crear será:
     - Si se trata de un cambio correctivo:
       - *Preautorizado / Cambio Correctivo de Software (Asociado a entrega de Versión)*
     - Si se trata de un cambio funcional:
       - *Cambio ordinario software* si la implementación del cambio no se considera urgente
       - *Cambio urgente software* si se considera urgente. Obsérvese que el flujo de gestión de este tipo de cambio es el mismo que el del cambio ordinario, sólo que se permite que los pasos que no sean imprescindibles para la implementación se hagan a posteriori
   - En el campo "Justificación" de la ficha del cambio en EasyVista se explicará el alcance del cambio a realizar. El campo "Nota" puede utilizarse para describir detalles técnicos del cambio, si estos no son muy extensos; si son extensos se deberán describir en un documento de especificación del cambio (DEC) y se pondrá un enlace (ruta UNC) al mismo en el campo "Nota" de la ficha del cambio en EasyVista, tal y como se describe en el apartado 7.2.
   - Se implementará el (los) cambio(s) siguiendo los especificado por MEDRA en el capítulo 7 del presente documento (*Gestión de pequeños mantenimientos*)
   - La entrega de la nueva versión se asociará a la(s) petición(es) de cambio creada(s)

Obsérvese que algunos sistemas de información pueden encontrarse en situación de "mantenimiento continuo"; es decir, que continuamente se están produciendo pequeños cambios correctivos o funcionales sobre el sistema, de tal modo que no cabe plantearse la realización de proyectos evolutivos de gran entidad. En estos casos, para optimizar la gestión de las entregas y reducir los riesgos asociados a las mismas, se agruparán estos pequeños cambios en conjuntos y se realizarán entregas periódicas con estos conjuntos de cambios. Este tipo de mantenimientos se gestionarán según lo estipulado en el apartado c) del párrafo anterior.

2.2. MEDRA y la gestión de proyectos y servicios

MEDRA es una metodología de trabajo específicamente dirigida a la actividad del desarrollo de *software*. Sin embargo, no se puede olvidar que el desarrollo de un sistema es tan sólo un proceso dentro de un escenario más global que debe contemplar asimismo, y entre otros aspectos:

- La implantación y puesta en marcha del sistema
- El soporte y la formación a los usuarios
- El mantenimiento de los sistemas en los que se apoya el *software* desarrollado
- Las relaciones con los clientes / usuarios, antes, durante y después del desarrollo, la gestión de sus expectativas y percepción del sistema, la buena imagen de la empresa
- La gestión de riesgos (no sólo asociados al desarrollo del sistema, sino también a su explotación) y contingencias (recuperación de fallos o desastres, continuidad del servicio)

Todas estas cuestiones son objeto de las metodologías de gestión de proyectos y de gestión de servicios de TI, por lo que quedan fuera del alcance de la metodología de desarrollo MEDRA y, por tanto, del presente documento.

No obstante, dada su relevancia de cara a asegurar el éxito de un proyecto de desarrollo, se incluirán en el presente apartado algunas prácticas que, aunque serían más propias de un manual de gestión de proyectos, se especifican aquí y deben considerarse de obligado cumplimiento.

2.2.1. Reunión de lanzamiento de proyecto

Cuando se vaya a iniciar un proyecto de desarrollo, el responsable del mismo en ITCGAE (jefe de proyecto de desarrollo) deberá convocar una reunión de lanzamiento cuyos objetivos principales son:

- Dar a conocer a la organización (ITCGAE y, en su caso, CGAE) el nuevo proyecto: sus objetivos, implicaciones, participantes y planificación inicial estimada.
- Coordinar con otras áreas participantes (Gestión del servicio, Explotación, Soporte e Implantación) su intervención en el mismo (con fechas estimadas).
- Presentar al equipo de trabajo encargado del desarrollo. En caso de que el desarrollo lo vaya a realizar un proveedor externo, se presentará como mínimo el jefe de proyecto de la empresa adjudicataria.
- Decidir los primeros pasos a dar en el proyecto y establecer la periodicidad de las reuniones de seguimiento del mismo.

A esta reunión deberán estar convocados, al menos:

- Un representante de cada una de las áreas de Gestión del servicio, Explotación y Soporte e Implantación. Para proyectos estratégicos o de gran envergadura, deberán ser los correspondientes jefes de área.
- El equipo interno de desarrollo de ITCGAE que participe en el proyecto, así como el jefe de área de Desarrollo
- Para proyectos desarrollados por un proveedor externo, el jefe de proyecto de la empresa adjudicataria, y el equipo que él considere.
- Para proyectos estratégicos o de gran envergadura, el director de ITCGAE

- Para proyectos cuyo cliente sea el Consejo General de la Abogacía, el representante del CGAE que sea apropiado en función del área funcional de que se trate y de la criticidad y envergadura del proyecto

Se deberá levantar acta de dicha reunión, la cual quedará archivada en el lugar correspondiente de la carpeta de proyecto, según se especifica en el capítulo 0.

2.2.2. Coordinación con otras áreas de ITCGAE

Es necesario que las áreas de Gestión del servicio, Explotación y Soporte e Implantación conozcan la existencia, alcance y envergadura de un nuevo proyecto de desarrollo desde el principio, para que puedan planificar los recursos necesarios para el futuro despliegue, explotación, soporte y gestión del nuevo servicio.

Como se ha indicado en el punto anterior, estas áreas estarán informadas del proyecto desde la reunión de lanzamiento. En dicha reunión se deberán coordinar con dichas áreas:

- Con Explotación:
  - Los recursos materiales extraordinarios que puedan ser necesarios para el desarrollo del sistema
  - Si se prevé que vaya a ser necesario algún entorno adicional a los habituales (Desarrollo/Integración, Preproducción, Producción), y si se prevé que los entornos tengan necesidad de algún recurso fuera de lo ordinario
- Con Gestión del servicio y Soporte e Implantación:
  - El impacto previsto del nuevo sistema sobre los usuarios / clientes, las necesidades de publicitar, informar, formar y, en general, gestionar la relación con dichos usuarios / clientes
- Con todas:
  - Las fechas previstas inicialmente (de forma estimativa) para los pasos a preproducción y producción
  - La posible documentación específica que se pueda requerir para el sistema en cuestión, más allá de la documentación estándar de todo proyecto de desarrollo

A partir de ese momento, Desarrollo deberá mantener cierta coordinación con las demás áreas; en particular, se les deberá notificar:

- Cuando haya cambios de calado en la planificación inicial del proyecto de desarrollo
- Cuando haya cambios de calado en el alcance del proyecto
- Cuando esté próximo un despliegue relevante en preproducción, y el despliegue en producción

Se deberá proporcionar acceso de lectura a las demás áreas a la documentación de desarrollo del proyecto para que puedan conocer el alcance funcional, la arquitectura del sistema, planes de pruebas, etc. En la reunión de lanzamiento se les informará de la ubicación de la carpeta de proyecto.

En el caso de pequeños mantenimientos, cuando se vaya a proceder a la subida a preproducción de una entrega, se deberá proporcionar a las áreas un listado de las peticiones de cambio y, en su caso, incidencias, que se resuelven en dicha entrega. Dicho listado puede obtenerse de la herramienta EasyVista a través de la opción:

Transition -> Informes -> RedAbogacía -> Relaciones Entregas - RFC - Incidencias

2.2.3. Prevención de problemas de despliegue. Entrega temprana

Uno de los problemas más reiterativos que se han detectado en los proyectos de ITCGAE cuyo desarrollo se encarga a proveedores externos es que la empresa encargada realiza la primera entrega cuando el desarrollo ya está muy avanzado, en muchas ocasiones cuando el hito de puesta en producción comprometido con el usuario / cliente está ya cercano; y es frecuente que esta primera entrega presente problemas de despliegue debido a las características propias de la arquitectura corporativa de ITCGAE. Esto provoca retrasos en la planificación debido al tiempo extra que el proveedor debe invertir en conseguir ajustar el entregable de forma que despliegue adecuadamente en dicha arquitectura.

Este problema se da especialmente con proveedores que no tienen mucha (o ninguna) experiencia en la arquitectura propia de ITCGAE.

Para paliar este problema se plantea la necesidad de que los proveedores realicen en todos los proyectos que deban desplegarse sobre la plataforma corporativa de ITCGAE una "entrega temprana". Esta entrega deberá realizarse lo antes que sea posible dentro de la planificación del proyecto, y deberá contar con una funcionalidad mínima que permita verificar el correcto despliegue sobre la arquitectura.

En particular, esta mínima funcionalidad deberá incluir al menos:

- La integración con el portal de servicios de RedAbogacía, cuando se haya definido que la aplicación deba integrarse en dicho portal. Es decir, verificar que se accede desde un elemento de menú de RedAbogacía a la aplicación, con el modelo de integración que se haya definido.
- Autenticación del usuario según el modelo que se haya definido, si se trata de una aplicación web no integrada en el portal de servicios de RedAbogacía
- Obtención de datos del usuario a partir del certificado, cuando se utilice un certificado para la autenticación (sea a través del portal RedAbogacía o de forma independiente)
- Establecer conexión con la base de datos (si la aplicación o servicio lo van a necesitar para su funcionalidad)
- Para aplicaciones web, mostrar una página inicial de la interfaz de usuario. Esta página deberá contener:
    - Contenido estático: al menos algún recurso gráfico (texto e imagen), una hoja de estilos en cascada (css) básica y un script básico en lenguaje *javascript*. Este contenido se desplegará en un servidor de *front-end*
    - Contenido dinámico: la página deberá generar algún texto de forma dinámica y mostrarlo. Este contenido se desplegará en un servidor de *back-end*
- Para servicios web, implementar un método básico de prueba que retorne un valor constante
- Tanto para aplicaciones como para servicios web, registrar una línea de texto informativo en un fichero de *log*

Se coordinará con el área de Explotación el despliegue de esta entrega temprana y su prueba. Los problemas de despliegue que surjan se podrán corregir desde ese momento y en paralelo al avance del desarrollo del proyecto, de modo que el impacto sobre las fechas planificadas de entrega final sea mínimo.

Este procedimiento podrá obviarse si el proveedor tiene suficiente experiencia en el desarrollo de aplicaciones para la arquitectura específica de RedAbogacía. En caso contrario, este hito deberá exigirse al proveedor (deberá estar incluido como requisito en la RFQ) y deberá ser incluido en la planificación del proyecto que se establezca a su inicio.

2.2.4. Gestión de los proveedores

Aquellos proyectos en los que se encarga a proveedores externos de ITCGAE el trabajo de desarrollo deben contemplar la problemática asociada a la relación con el proveedor y la gestión de la calidad de su trabajo.

No entra en el alcance de MEDRA la gestión de la relación con los proveedores, pero sí se citarán al menos algunas recomendaciones que se deberían seguir en este tipo de proyectos:

- Intentar especificar el alcance de la forma más detallada posible en la RFQ para evitar posteriores discrepancias en el alcance del proyecto una vez comenzado éste.
- Solicitar en la RFQ o en posterior negociación con el proveedor que las facturaciones parciales vayan ligadas a hitos de entregas validadas por ITCGAE. En caso de incumplimiento de estas entregas, retener las correspondientes facturas hasta que se subsane la situación.
- Hacer revisiones tempranas de la calidad del código entregado. Esto implica que el proveedor empiece a entregar código fuente lo más pronto que sea posible en el proyecto, no esperar al final del desarrollo para revisar el código. El código escrito por los proveedores debería seguir los mismos cánones de calidad que deseamos para el código propio.
- El proveedor debe aportar una persona con perfil claro de jefe de proyecto que tenga los conocimientos y experiencia adecuados a ese puesto. Si se detecta una carencia en este sentido, levantar la alarma con el proveedor lo antes posible y exigir que se incorpore dicho perfil.
- Tener reuniones de seguimiento periódicas en las que el jefe de proyecto del proveedor informe sobre el avance del proyecto conforme a lo planificado. Ser exigentes en el cumplimiento de la planificación; cuando aparezcan retrasos significativos, pedir explicaciones y plan de acción para recuperar el retraso.
- El proveedor debe comprometerse a seguir la metodología MEDRA.
- En caso de incidencias graves (incumplimientos, retrasos, baja calidad del trabajo, etc.) informar lo antes posible al jefe de área de Desarrollo e incluso a la Dirección de ITCGAE.

3. Evaluación del paradigma metodológico

Como ya se ha avanzado en el apartado anterior, MEDRA pretende cubrir bajo un mismo marco metodológico dos tipos de proyectos: aquellos en los que la aproximación más adecuada sea un desarrollo "ágil" (más flexible y adaptado al cambio) y aquellos en los que sea más adecuado un enfoque más tradicional "en cascada" (más formalista y estricto).

Para ello, en la fase inicial del proyecto se procederá a realizar una evaluación del mismo, analizándolo a la luz de determinados parámetros que determinarán cuál de los dos enfoques es más apropiado. En función del enfoque elegido, se procederá a seguir un enfoque metodológico basado en SCRUM (capítulo **4** de este documento) o en Métrica v3 (capítulo **5**). En proyectos subcontratados a proveedores, lo idóneo es que esta evaluación se realice en el momento de generar la RFQ, de modo que se dé ya a los ofertantes una indicación del paradigma metodológico escogido para el proyecto. Para ello, podría ser necesario o conveniente celebrar reuniones previas con los proveedores candidatos para contar con su opinión.

La siguiente tabla recoge los criterios que deberán evaluarse al inicio de cada proyecto para escoger el paradigma metodológico más adecuado. Para cada criterio se evaluará el valor que le aplica, y en función de este, se escogerá un carácter "A" (de "ágil"), "C" (de "en cascada") o "I" (indistinto).
Al final deberán sumarse los caracteres "A" y "C" y el que obtenga un número mayor determinará el paradigma metodológico a aplicar. En caso de obtenerse igual número de "A" y "C", el responsable del proyecto, basándose en su experiencia y con el apoyo de su equipo, deberá decantarse por una u otra opción.
La tabla de evaluación y el paradigma seleccionado deberán quedar reflejados en el documento de plan de proyecto.

Orden	Criterio	Valor		
1.	Duración prevista del proyecto	<=3 m	>3 m <1 año	>= 1año
		A	C	A
2.	Grado de definición previsto para los requisitos (es decir, si se prevé que los requisitos van a estar claramente establecidos)	Bajo	Moderado	Alto
		A	A	C
3.	Probabilidad de cambio de los requisitos durante la ejecución del proyecto	Baja	Moderada	Alta
		C	C	A
4.	Expectativa o necesidad del cliente de tener entregas a corto plazo, aunque sean versiones incompletas	Baja	Moderada	Alta
		C	A	A
5.	Disponibilidad de la(s) persona(s) que debe(n) suministrar los requisitos del sistema y validar el *software* entregado	Baja	Moderada	Alta
		C	I	A
6.	Capacidad de autoorganización del equipo de trabajo	Baja	Moderada	Alta
		C	C	A
7.	Conocimiento del entorno funcional por parte del equipo de trabajo	Bajo	Moderado	Alto
		C	C	A
8.	Necesidad de justificar documentalmente los trabajos realizados	Baja	Moderada	Alta
		A	C	C

4. Metodología para el paradigma ágil - MEDRA^{A}

Para aquellos proyectos en los que se determine que el enfoque metodológico más adecuado es el paradigma "ágil", se aplicará una metodología de trabajo basada en SCRUM, según se describe en este apartado.

SCRUM es un modelo de desarrollo ágil caracterizado por:

- Adoptar una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto
- Basar la calidad del resultado más en el conocimiento tácito de las personas en equipos autoorganizados, que en la calidad de los procesos empleados
- Solapamiento de las diferentes fases del desarrollo, en lugar de realizar una tras otra en un ciclo secuencial o de cascada.

La metodología descrita en el presente apartado no pretende ser una trasposición literal de SCRUM, sino que adopta las líneas maestras de esta metodología y las adapta a la idiosincrasia de IT CGAE y sus clientes (CGAE, ICAs y abogados).

Por brevedad, denominaremos MEDRA^{A} a la metodología ágil contemplada dentro de MEDRA.

4.1. Concepto de *sprint*

MEDRA^{A} se basa en el concepto de *sprint*, que podemos definir brevemente de la siguiente forma: *Un ciclo de trabajo a lo largo del cual se implementa un incremento de funcionalidad*.

Típicamente, la duración de un *sprint* debe ser aproximadamente de un mes, aunque esta duración puede ajustarse ligeramente según las necesidades del proyecto.

El *sprint* arranca con una reunión de planificación en la que se establece el alcance y la duración del mismo. A partir de esa reunión el equipo empieza a trabajar en el diseño e implementación del *software* que cubrirá la funcionalidad prevista en el alcance del *sprint*. Al finalizar el *sprint* el equipo entrega el *software* para su validación, y se realiza una reunión de revisión del *sprint*. Inmediatamente después comenzará el siguiente *sprint*.

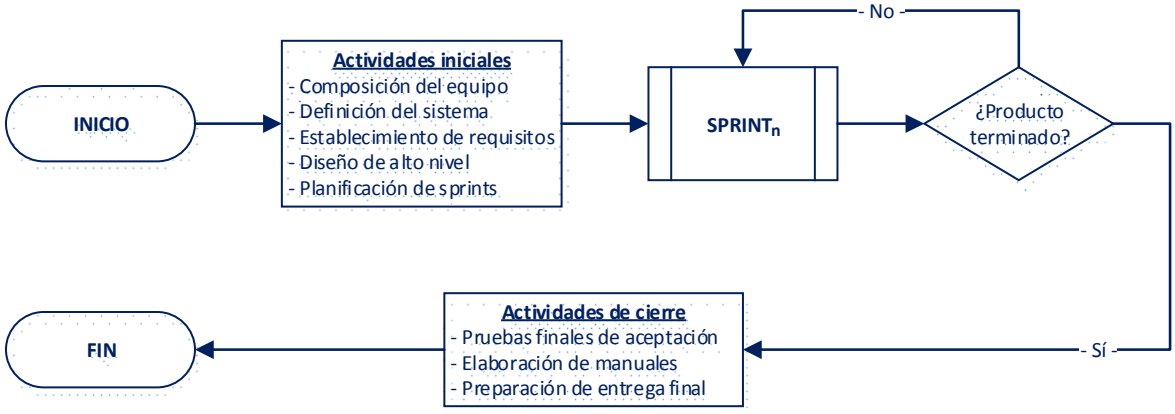
El avance del proyecto se produce, por tanto, por una sucesión de *sprints* que van añadiendo funcionalidad al producto hasta que este está terminado. En proyectos muy sencillos, puede plantearse incluso un único *sprint*.

Es importante resaltar que, durante un *sprint*:

- No se pueden realizar cambios que afecten al objetivo de ese *sprint*
- No se pueden disminuir los objetivos de calidad (no debe disminuir la calidad del producto porque se esté agotando el tiempo del *sprint*)
- El alcance del *sprint* puede ser renegociado si fuera necesario, siempre sin contravenir los dos puntos anteriores

4.2. Visión general del flujo de trabajo

El flujo de trabajo especificado por MEDRA[A] puede representarse por el siguiente diagrama:



Como se puede apreciar, existe una serie de actividades que se realizan una única vez al inicio del proyecto:

1. Composición del equipo de proyecto
2. Definición del sistema
3. Establecimiento de requisitos
4. Diseño de alto nivel
5. Planificación de *sprints*

A continuación se inicia la ejecución de los *sprints.* Cada *sprint* añade funcionalidad al sistema, hasta que éste esté terminado. Dentro de cada *sprint* podemos distinguir las siguientes actividades:

1. Planificación del *sprint*
2. Reunión diaria
3. Diseño técnico
4. Implementación de código
5. Pruebas unitarias y de integración
6. Despliegue en Preproducción
7. Revisión del *sprint*

Por último, se lleva a cabo una serie de actividades para la finalización y cierre del proyecto:

1. Pruebas finales y de aceptación del sistema
2. Elaboración de manuales
3. Preparación de la entrega final

Los apartados siguientes detallan cada una de estas etapas y actividades.

4.3. Actividades de inicio del proyecto

Las actividades descritas en este apartado se realizarán una única vez, al inicio del proyecto.

4.3.1. Composición del equipo de proyecto

En las metodologías "ágiles" uno de los factores fundamentales es la identificación de roles y la composición del equipo de proyecto.

En MEDRA[A] se establecen los siguientes roles:

- **Definidor-validador (DV)**. Es la persona o conjunto de personas que define(n) los requisitos del sistema y que posteriormente validará(n) si el producto entregado se ajusta a sus necesidades
- **Gestor del proyecto (GP)**. Es la persona encargada de velar por la ejecución del proyecto. Tiene un papel de "facilitador" más que de "jefe". Sus principales cometidos son:
    - Acordar con el definidor-validador el alcance general del sistema y la planificación general de *sprints*
    - Facilitar la comunicación entre el definidor-validador y el equipo de trabajo
    - Asegurarse de que cada sprint se dedica a cubrir el alcance acordado (ni más ni menos)
    - Velar por el cumplimiento de las etapas y actividades definidas en la metodología
    - Asegurar la provisión de las herramientas que el equipo necesite para el desarrollo
    - Mantener información sobre el estado y grado de avance del proyecto en general y de cada *sprint* en particular.
- **Miembro del equipo (ME)**. Cada una de las personas que componen el equipo de trabajo del proyecto y que se encargarán de:
    - Tomar y analizar requisitos
    - Diseñar el sistema (a alto y bajo nivel)
    - Implementar código
    - Realizar pruebas (unitarias, de integración, de rendimiento, etc.)
    - Preparar entregas de *software*
    - Escribir documentación del proyecto

  Obsérvese que dentro del equipo no se distinguen roles por criterios tradicionales como analista funcional, analista-programador, etc. La idea es que todos los miembros del equipo trabajen en un plano de igualdad y colaboración, asumiendo en cada caso la tarea más adecuada a sus conocimientos y aptitudes, y con la posibilidad de ser polivalentes; es decir, una misma persona puede en distintos momentos hacer análisis, diseñar, codificar, probar, etc. El equipo debe ser auto-organizado: es decir, sus propios miembros deciden entre ellos, por acuerdo, qué tareas realiza cada uno en cada momento del proyecto. Por tanto, siempre que en este documento se haga referencia a que una tarea la desempeña el "equipo de proyecto" o "los miembros del equipo" deberá entenderse que la realizan el / los miembro/s que se consideren más adecuados para esa tarea.

En esta primera actividad se deberá identificar por tanto:

1. La persona o personas que desempeñarán el rol de **Definidor-Validador** (en general se tratará de clientes o usuarios finales, o alguien que los represente)
2. La persona que desempeñará el rol de **Gestor del proyecto**. Idealmente deberá ser una persona con conocimiento y experiencia en metodologías de trabajo ágiles.
3. Las personas que compondrán el **equipo de proyecto** (**miembros del equipo**). Idealmente se tratará de un elenco de personas que, entre todos, reúnan las aptitudes y conocimientos necesarios para abordar las tareas descritas anteriormente para este rol. El número de miembros del equipo se definirá también en este momento, y deberá ser el adecuado para abordar la ejecución del proyecto en el plazo objetivo. El dimensionamiento deberá realizarse en base a técnicas de estimación de esfuerzos, que quedan fuera del alcance del presente documento.

La asignación de roles descrita en los puntos precedentes deberá quedar explícitamente reflejada en el documento de plan de proyecto.

4.3.2. Definición del sistema

En esta actividad se realiza una definición del alto nivel del sistema a desarrollar, que incluirá los siguientes elementos:

1. **Plataforma**. Se definirá la plataforma sobre la que se desarrollará el sistema:
    a. **Arquitectura general:** Número y distribución de capas del sistema, así como sus elementos de arquitectura, y tecnología de cada uno de ellos (tipo de *front-end*, de *back-end*, *middleware*, gestor de base de datos, servidores de ficheros, máquinas cliente, dispositivos móviles, tipos de red, etc.). Se recomienda la inclusión de un diagrama de contexto que los represente gráficamente.
    b. **Tipo de plataforma software:** Java (JSE o JEE), Microsoft .Net, Android, PhoneGap, etc.

2. **Alcance.** Se definirá a alto nivel el alcance del sistema a desarrollar, especificando:
    a. Universo de usuarios
    b. Entradas y salidas del sistema
    c. Principales casos de uso a cubrir
    d. Principales flujos de trabajo, datos y/o documentos
    e. Casos de uso especialmente relevantes que queden fuera de alcance, si procede
    f. Interfaces y comunicaciones con otros sistemas, si procede

La definición del sistema descrita en este apartado deberá quedar reflejada en el apartado correspondiente dentro del documento de plan de proyecto.

Se registrará una petición de cambio en EasyVista:

- Si el proyecto va a dar lugar a un nuevo sistema o servicio, será de tipo “Nuevo servicio”
- Si el proyecto va a dar lugar a una nueva versión de un sistema o servicio, incluyendo un número relevante de cambios en el mismo, será de tipo “Modificación relevante de un servicio”

El responsable de la realización de esta actividad es el gestor del proyecto, en colaboración con los miembros del equipo.

4.3.3. Establecimiento de requisitos

En esta actividad se deberá realizar una descripción funcional a alto nivel del sistema, así como recopilar y analizar los requisitos del mismo. Los requisitos deberán ser recopilados por el equipo de proyecto, a partir de la información facilitada por el (los) definidor-validador(es). Para ello se utilizará una o varias de las siguientes técnicas:

- **Entrevistas.** Los miembros del equipo se reúnen con el (los) definidor-validador(es) en varias sesiones de trabajo y realizan baterías de preguntas orientadas a conocer las necesidades del usuarios y las funcionalidades que debe cumplir el sistema.
- **Observación directa.** Los miembros del equipo presencian la forma de trabajar actual de los usuarios para comprender y deducir cómo podría el sistema a desarrollar ayudarles en el desempeño de su actividad.
- ***Brainstorming*.** Los miembros del equipo se reúnen con el (los) definidor-validador(es) en varias sesiones de trabajo y lanzan con rapidez todas las ideas que se les ocurran sobre posibles funcionalidades del sistema. Estas ideas se van anotando y, posteriormente, se van filtrando, desechando las que carezcan de sentido, no sean aplicables o rentables, etc.
- **Prototipado "en papel".** Los miembros del equipo se reúnen con el (los) definidor-validador(es) en varias sesiones de trabajo y dibujan en papel bocetos de interfaces de usuario y diagramas de navegación que representen distintos casos de uso del sistema. Los bocetos se van ajustando y modificando sobre la marcha en el papel hasta llegar a una versión que satisfaga al (los) definidor-validador(es). Lógicamente, en lugar de utilizar lápiz y papel se pueden utilizar herramientas informáticas de prototipado rápido de interfaces de usuario.

Se deberán recopilar requisitos de los siguientes caracteres:

- Funcionales
- No funcionales:
    - De nivel de servicio (tiempos de respuesta, rendimiento, disponibilidad)
    - De usabilidad y accesibilidad
    - De interoperabilidad con otros sistemas
    - De confidencialidad de la información, atendiendo, entre otros, a los criterios de la LOPD
    - De seguridad. Entre estos se deberán considerar específicamente:
        - Mecanismos de identificación, autenticación y autorización
        - Mecanismos de protección de la información tratada. En particular, y en relación con la confidencialidad de los datos, se analizará la necesidad o conveniencia del cifrado de datos
        - Generación y tratamiento de pistas de auditoría

Los requisitos recopilados deberán registrarse en un catálogo en la herramienta corporativa de gestión de requisitos definida en el anexo B.

Debe tenerse en cuenta que en MEDRA[A], por tratarse de una metodología ágil, no es necesario ni conveniente llegar a un alto grado de detalle en la especificación de los requisitos. La filosofía de la metodología consiste en que el producto de cada sprint se irá validando y refinando con el definidor-validador en aproximaciones sucesivas; por tanto no es tan necesario como en las metodologías "tradicionales" partir de unos requisitos especificados con gran detalle. No debe por tanto invertirse un tiempo excesivo en esta actividad, y debe alcanzarse sólo el nivel de especificación mínimo que el equipo de proyecte considere necesario para empezar a diseñar la solución.

En esta actividad se deberá generar también un breve documento de análisis que contenga:

- Una descripción funcional a alto nivel del sistema. Como ya se ha mencionado, no se debe entrar en excesivos detalles ni debe ser muy extensa, ya que la filosofía ágil huye del exceso de documentación y se centra en las validaciones continuas, pero sí es conveniente que al menos se dé una impresión general de qué es lo que debe hacer funcionalmente el sistema que se está construyendo.
- Propuestas de diseño (*mock-ups*) de la interfaz de usuario. Se utilizarán para validar la interfaz de usuario con el definidor-validador antes de construir el software. En esta actividad se realizarán unas propuestas iniciales, que se irán ampliando y refinando a lo largo de los distintos *sprints*.

4.3.4. Diseño de alto nivel

En esta actividad el equipo de proyecto, basándose en los requisitos establecidos en la actividad anterior, realiza un diseño general del sistema a desarrollar. No se trata de un diseño detallado (que se realizará en los distintos *sprints*), sino de un diseño general del sistema, de alto nivel, en el que se definirá:

- Descomposición en subsistemas (si procede)
- Distribución del sistema en capas (*front-end*, *back-end*, *middleware*, servicios…)
- Principales componentes o módulos y sus relaciones
- Interfaces con sistemas externos

Para plasmar este diseño, además de la descripción textual, se utilizarán los diagramas UML que se consideren necesarios:

- Diagramas de despliegue
- Diagramas de clases
- Diagramas de componentes
- Diagramas de secuencia y/o colaboración

También se emplearán, cuando aplique, esquemas de sitios web y navegación.

Los requisitos de seguridad deberán ser contemplados empezando desde este primer nivel de diseño del sistema.

El diseño de alto nivel deberá quedar plasmado en el correspondiente documento de diseño técnico del proyecto.

4.3.5. Planificación de *sprints*

En esta actividad se reúne el gestor del proyecto con el equipo para definir:

- El número de *sprints* que se realizarán en el proyecto. Para determinar este número deberán basarse en los siguientes factores:
  - Plazo total con el que se cuenta para el desarrollo
  - Duración de cada sprint, que debe ser, idealmente, de aproximadamente un mes (aunque puede ajustarse ligeramente al alza o a la baja)
  - Disponibilidad del (los) definidor-validador(es) para realizar las validaciones en cada final de *sprint*
- Reparto a alto nivel de la funcionalidad del sistema en "paquetes" y priorización de los mismos. Para ello deberán considerar:
  - Importancia de la funcionalidad para el usuario (intentar priorizar las funcionalidades más relevantes)
  - Priorizar funcionalidades de interacción con el usuario (para intentar validarlas cuanto antes) frente a funcionalidades "internas" del sistema
  - Posibles dependencias entre paquetes
- Asignación de los paquetes de funcionalidad a los *sprints*, en orden de prioridad
- Estimación de fechas de inicio y fin de cada *sprint*. Esto se reflejará en un diagrama tipo Gantt.

La planificación determinada en esta actividad deberá ser reflejada en el documento de plan de proyecto.

4.4. Actividades de cada *sprint*

Las actividades descritas en este apartado se realizarán dentro de cada *sprint*.

4.4.1. Planificación del *sprint*

En esta actividad se reúne el gestor del proyecto con el equipo para definir:

- **Alcance detallado del *sprint***. Se partirá de la planificación global de *sprints* realizada en la actividad 0. En este paso se detallarán las funcionalidades concretas que se abordarán en el sprint actual y se les asignarán prioridades. Se contempla la posibilidad de que algunas funcionalidades inicialmente previstas para este sprint puedan quedar fuera, o al contrario, que se incluyan funcionalidades inicialmente previstas para el sprint siguiente; esto dependerá del análisis detallado de las funcionalidades a implementar que se realice durante la reunión. El alcance del *sprint* debe estar orientado a alcanzar la consecución de un objetivo concreto (por ejemplo: *implementar el módulo de administración del sistema*), para ayudar al equipo a enfocarse en lograr ese objetivo. También se deberán ampliar en extensión y detalle, si es necesario, los *mock-ups* de interfaz de usuario de las partes del sistema incluidas en el alcance del *sprint*, que deberán quedar reflejados en el documento de análisis.
- **Reparto de tareas** entre los miembros del equipo. Los miembros del equipo deberán comprometerse en esta reunión a tener las tareas asignadas en el plazo previsto para cada una. Como ya se ha dicho, el equipo es auto-organizado, por lo que la asignación de tareas la hace el propio equipo, atendiendo a las aptitudes, conocimientos y área de experiencia de cada miembro.

La duración de la reunión de planificación no deberá superar en ningún caso una jornada completa de trabajo.

La planificación del sprint surgida de esta reunión deberá quedar reflejada en el documento de plan de proyecto, y las tareas asignadas a cada miembro del equipo se registrarán en la herramienta de gestión de tareas del proyecto (ver anexo B) para su seguimiento.

4.4.2. Reunión diaria

Durante la ejecución del sprint, el equipo de proyecto se deberá reunir diariamente, al principio de la jornada laboral, a una hora establecida y en un lugar fijo (siempre los mismos para ese *sprint*) y de forma breve. Esta reunión diaria, cuya duración idealmente no deberá exceder de 15-20 minutos, sirve para que el equipo se mantenga sincronizado y cree un pequeño plan de trabajo para esa jornada.

Básicamente en la reunión cada miembro del equipo debe exponer de forma breve y concisa:

- Qué hizo el día anterior y cómo contribuyó a alcanzar el objetivo del *sprint*
- Qué hará ese día para contribuir a alcanzar el objetivo del *sprint*
- Qué impedimentos o riesgos percibe que puedan dificultar el alcance del objetivo

El resto del equipo, cuando lo considere oportuno, dará su opinión o ayudará al miembro que está exponiendo a resolver sus dificultades.

Esta reunión servirá además para ir sincronizando el trabajo de aquellos miembros del equipo que estén desarrollando piezas de software que deban interoperar.

Se trata de una reunión de todo el equipo, breve y obligatoria. Esto no impide que miembros del equipo, a lo largo de la jornada, puedan o deban reunirse entre ellos para tratar temas concretos con más detalle.

Para dar agilidad al trabajo, no se realizará acta formal de esta reunión ni se documentará en el plan de proyecto. Es conveniente, sin embargo, que los miembros del equipo tomen sus propias notas para consultarlas si lo consideran necesario.

4.4.3. Diseño técnico

En esta actividad, que será desarrollada por el equipo de proyecto, se realiza un diseño técnico detallado del (los) módulo(s) que vaya a ser implementado en el *sprint* actual para cubrir las funcionalidades previstas en el mismo.

Se deberán definir las clases, interfaces, eventos, mensajes, etc. que se utilizarán en la construcción del *software*. Para ello se utilizarán las siguientes técnicas (las que sean necesarias en cada caso) y las herramientas indicadas en el anexo B:

- Descripciones textuales
- Diagramas UML:
  - Diagramas de clases y objetos
  - Diagramas de componentes
  - Diagramas de actividad
  - Diagramas de secuencia y/o colaboración
  - Diagramas de estado
- Esquemas de sitios web y navegación (cuando aplique)
- Prototipos de interfaz de usuario
- Casos de pruebas unitarias

Los requisitos de seguridad y privacidad deberán ser tenidos en cuenta en el diseño técnico de todos los módulos del sistema. En particular, deberán tenerse en consideración desde el diseño del sistema:

- Los mecanismos de identificación, autenticación y autorización
- Los mecanismos de protección de la información tratada.
- La generación y tratamiento de pistas de auditoría

El diseño detallado deberá quedar plasmado en el correspondiente documento de diseño técnico del proyecto.

4.4.4. Implementación de código

En esta actividad, que será desarrollada por el equipo de proyecto, se implementa el código según lo diseñado en el paso 4.4.3 para construir el *software* que proporcione las funcionalidades previstas en el *sprint*.

Para ello se utilizarán las siguientes herramientas:

- Entorno integrado de desarrollo (IDE). Para desarrollo Java se propone el uso de *Eclipse*.
  - Para la escritura del código la metodología establece plantillas de formateo de código (*formatters*) para cada lenguaje (Java, Javascript, etc.). Los miembros del equipo deberán utilizar estas plantillas para conseguir que el formato del código sea homogéneo.
- Sistema de control de versiones. Se utilizará el producto *Subversion*.
  - Los desarrolladores sólo deberán hacer *check-in* del código fuente que esté adecuadamente probado, y no de módulos fuente que estén en proceso de ser modificados. En particular, el código debe pasar positivamente las pruebas unitarias automatizadas definidas para ese código (ver apartado 4.4.5.1)
  - Cada vez que un desarrollador termine de modificar un código fuente, y ese código haya sido debidamente probado y verificado su correcto funcionamiento, deberá hacer *check-in* de ese módulo a la mayor brevedad posible para evitar dejar código "bloqueado" en su PC

La implementación de código deberá realizarse siguiendo la guía de buenas prácticas incluida en el anexo C del presente documento.

Entre ellas cabe destacar las relativas a la implementación de la seguridad en el código.

4.4.5. Pruebas unitarias y de integración

4.4.5.1. Pruebas unitarias automatizadas

Durante la etapa de diseño, y también en paralelo a la implementación de código, se deberán definir juegos de pruebas unitarias para su ejecución automática, siguiendo la arquitectura *xUnit*.

Específicamente, para desarrollo Java se utilizará el *framework* *JUnit* para la definición y ejecución de pruebas unitarias automatizadas.

El objetivo del uso de esta práctica es doble:

- Por un lado, aprovechar los beneficios del proceso de desarrollo TDD (*Test-Driven Development, Desarrollo orientado a la prueba*):
  - Conduce al desarrollador a entender mejor para qué sirve el código que va a escribir, lo que redunda en una mejora de la calidad del código
  - Reduce el número de errores, al hacer al desarrollador más consciente de que debe hacer el código más robusto para pasar las pruebas
  - Provoca una mayor modularidad del código, ya que fuerza a hacer unidades de código más pequeñas que puedan ser probadas de forma unitaria
- Por otro, las pruebas automatizadas facilitan la realización de pruebas de regresión en las que se pueda detectar que una modificación en el código haya "roto" alguna funcionalidad anterior (efectos colaterales)

Los trabajos de compilación definidos en sistemas de construcción de código como *Ant* o *Maven* deberán incluir como paso necesario la ejecución de las pruebas unitarias tras la construcción de los ejecutables. Un fallo en las pruebas unitarias implicará un fallo en la construcción (*build*) y será condición suficiente para que el código no pueda subirse con *check-in* al repositorio.

4.4.5.2. Pruebas unitarias manuales

Adicionalmente a la ejecución de las pruebas unitarias automáticas, cada desarrollador será responsable de probar el módulo de *software* que esté creando o modificando de forma unitaria (es decir, centrándose en la funcionalidad de ese módulo, y no en la interacción con otros módulos).

Se deberá hacer especial hincapié en las pruebas de la interfaz de usuario, ya que los errores de visualización no pueden ser detectados por herramientas automáticas.

Es muy importante que se realicen pruebas no sólo de los casos "típicos" sino también de los casos "límite", es decir, aquellos en los que se reciben valores o comportamientos no esperados. Por ejemplo:

- En un método que recibe un argumento numérico que se espera sea positivo, pasar valores negativos o cero
- En un método que recibe un argumento numérico, intentar pasar valores muy grandes para ver si se controla el *overflow.*
- En un método que recibe un argumento de cualquier tipo de objeto (por ejemplo un *string*), probar a pasar el valor a *null*. En caso de *strings*, probar también el caso de "cadena vacía".
- En un método que recibe un *string* que representa una fecha, probar a pasar valores que no sean fechas correctas.
- En un método que recibe un argumento que actúa como índice de una colección o *array*, probar a pasar valores de índice fuera de rango

- Probar acciones no esperadas en la interfaz de usuario, por ejemplo: hacer clic repetidamente en un control mientras ya se está ejecutando la acción asociada (control de reentrancias no deseadas), introducir textos extremadamente largos en campos de texto, etc.

Todos estos comportamientos no esperados deben estar adecuadamente controlados en el código para evitar excepciones no controladas, errores funcionales, corrupción de datos, etc.

Otro aspecto importante a observar es la realización de pruebas con usuarios de distintos perfiles, roles, grupos, etc. Por ejemplo, realizar pruebas con un usuario que pertenezca al CGAE y con otros que pertenezcan a distintos Colegios de Abogados; con usuarios "normales" y usuarios con mayores privilegios (administradores, etc.), etc.

Siempre que sea posible, la metodología recomienda la realización de pruebas "cruzadas" (es decir, cada desarrollador del equipo prueba módulos de sus compañeros, y los compañeros prueban los suyos).

Como en el caso de las pruebas automáticas, el código no podrá considerarse terminado hasta que no se hayan realizado todas las pruebas unitarias que se entiendan necesarias.

4.4.5.3. Pruebas unitarias y dependencias

Si para las pruebas existe dependencia de otro módulo y no se dispone de él, se implementará un módulo de *mock* que "imite" al módulo real del cual se tiene dependencia, presentando una interfaz equivalente (esto se aplica tanto a las pruebas unitarias automatizadas como manuales).

También pueden darse casos en los que sí disponemos del módulo del cual tenemos dependencia, pero no nos interesa utilizarlo (porque su implementación es incompleta o poco fiable, porque consume muchos recursos, o porque implica acciones "en real" que no deben ejecutarse en las pruebas, como por ejemplo un cargo en una tarjeta de crédito); en este caso también puede interesar crear un *mock*.

4.4.5.4. Pruebas de integración

Una vez verificado el correcto funcionamiento de manera individual (unitaria) del módulo que se está desarrollando, se procederá a su prueba dentro del entorno de ejecución en el que vaya a participar, interactuando con el resto de módulos *software* que componen la solución.

Las pruebas de integración deberán ser coordinadas por el equipo de proyecto, y supervisadas por aquél o aquellos miembros del equipo que tengan una visión más global del sistema. Se deben enfocar las pruebas de integración en:

- Funcionalidad. La ejecución de las pruebas de los módulos interactuando entre ellos debe producir los resultados funcionales previstos.
- Integridad de datos. Se debe verificar que los datos no se corrompen o se malinterpretan como consecuencia de su paso entre módulos
- Rendimiento. Se debe comprobar que el funcionamiento del sistema tenga un rendimiento adecuado
- Consumo de recursos. Se debe comprobar que el consumo de recursos (CPU, memoria, operaciones E/S, *handlers*, *sockets*, etc.) sea adecuado para el código que se ejecuta, y en particular, que no haya "filtraciones" (*leaks*) de memoria u otros recursos.

Los fallos detectados en estas pruebas implicarán que el (los) módulo(s) afectado(s) deba(n) ser corregido(s), provocando por tanto nuevas etapas de pruebas unitarias y de integración.

Los datos utilizados para las pruebas deberán ser lo más similares posible a los datos reales que se utilizarán en el entorno de producción. Cuando sea conveniente, en el caso de proyectos evolutivos sobre sistemas ya en producción, se solicitará al área de Explotación la realización de volcados de datos desde el entorno de producción al de pruebas para contar con datos más realistas. Al realizar estos volcados deberán tenerse en cuenta las consideraciones de privacidad derivadas de la Ley Orgánica de Protección de Datos de carácter personal (LOPD); en particular, si en los datos de producción a volcar existen datos personales, se deberán aplicar modificaciones sobre los datos en el entorno de prueba que eviten la identificación de sujetos reales (alteración de nombres y apellidos y/o documentos de identidad).

4.4.6. Despliegue en Preproducción

Al final de cada *sprint* el equipo de proyecto deberá preparar una entrega que permita desplegar el software producido durante el sprint en un entorno de preproducción, de cara a su revisión y validación por parte del definidor-validador.

La confección y gestión de la entrega deberá realizarse conforme a los procedimientos establecidos en el Plan de Gestión de Servicios de IT CGAE.

Véanse los siguientes documentos (**x** representa el último número de versión liberada para cada documento):

- PGS.12 Gestión de Entrega y Despliegue-Vx.docx
- RPGS.12-01 Plan de Entregas.docx
- RPGS.12-02 Formato de informe de entrega.docx
- Modelo estándar de documento de entrega de IT CGAE
- ITPGS.12-01-Proceso Gestión de Entregas de Software-Vx_xx.docx

4.4.7. Revisión del *sprint*

Una vez desplegado el producto del *sprint* en el entorno de preproducción el equipo de proyecto, junto con el gestor del mismo y el definidor-validador, llevará a cabo la reunión de revisión del *sprint*. Esta reunión no deberá exceder en ningún caso una jornada completa de trabajo y durante la misma se llevarán a cabo las siguientes acciones:

- Se dejará constancia de en qué medida se cubrió el alcance del sprint (es decir, si hubo funcionalidades previstas que no se hayan implementado)
- Se realizará una demostración del producto del sprint al (los) definidor-validador(es) sobre el entorno de preproducción y se recogerán los comentarios de este. Estos comentarios se aportarán a la reunión de planificación del siguiente *sprint*.
- El equipo comentará los problemas que se encontró en el desarrollo del *sprint*, y entre todos se tratará de buscar soluciones para evitar esos problemas en futuros *sprints*.
- En base a la revisión realizada sobre el producto, se analizarán posibles cambios en alcance o en prioridad con respecto a la planificación general de los *sprints* realizada en el punto 0
- Si hubiera cambios en el alcance general del proyecto, o en sus condicionantes de plazo, coste, recursos, etc., estos deberán ser comunicados por el definidor-validador. En función de la envergadura de los cambios, podría ser necesario un replanteo total del proyecto, volviendo a ejecutarse los pasos descritos a partir del punto 4.3.2.

Una vez alcanzado este punto, el (los) definidor-validador(es) abandonarán la reunión, llevándose la información necesaria para poder acceder al entorno de preproducción y seguir evaluando el producto.

El gestor y el equipo de proyecto continúan la reunión abordando los siguientes puntos:

- Se revisa cómo fue el *sprint* en cuanto a relaciones interpersonales, colaboración de los miembros del equipo, procesos y herramientas utilizadas
- Se identifican los elementos principales que fueron bien (puntos fuertes) al objeto de promoverlos, y los que fueron mal (puntos débiles), y para estos últimos se proponen soluciones
- Se planteará un plan para implementar las mejoras que se estimen oportunas en el desarrollo del trabajo del equipo

Las principales conclusiones de esta reunión deberán reflejarse en el documento de plan de proyecto.

4.5. Actividades de cierre del proyecto

Las actividades descritas en este apartado se realizarán una única vez, al finalizar el proyecto.

4.5.1. Pruebas finales y de aceptación del sistema

4.5.1.1. Pruebas funcionales finales

Una vez finalizado el último *sprint*, se deberá realizar un ciclo completo de pruebas funcionales sobre el producto terminado. La responsabilidad de realizar estas pruebas es del equipo de proyecto; todo el equipo debe dedicarse a probar el sistema en esta etapa. Las pruebas se realizarán en el entorno de preproducción.

El objetivo de estas pruebas es intentar asegurar al máximo que el producto final está libre de errores funcionales. Si se detectan errores en esta etapa, se deberá corregir inmediatamente el *software* y se volverá a desplegar en preproducción.

El tiempo que se dedique a esta etapa deberá ser proporcional al "tamaño" y complejidad del *software* que se haya construido en el proyecto, y deberá ser acordado por el equipo de proyecto junto con el gestor.

4.5.1.2. Pruebas de seguridad

Finalizadas las pruebas funcionales, se deberá realizar un ciclo de pruebas específico orientado a evaluar el cumplimiento de los requisitos de seguridad establecidos para el sistema.

Estas pruebas serán llevadas a cabo por algún desarrollador, designado por el Responsable de Seguridad, que no forme parte del equipo de proyecto y el resultado de las mismas deberá quedar documentado. En caso de detectarse alguna incidencia, se comunicará al equipo de proyecto para que corrija el problema.

Se deberá comprobar, al menos, que:

- Se cumplen los criterios de aceptación en materia de seguridad.
- No se deteriora la seguridad de otros componentes del servicio.

Las pruebas se realizarán en el entorno de preproducción. En este entorno no se realizarán en ningún caso pruebas con datos reales.

Una vez superadas las pruebas, se recomienda llevar a cabo sobre el sistema, cuando sea posible, un análisis de vulnerabilidades, proceso en el cual, además, se intentarán explotar las deficiencias de seguridad identificadas (prueba de intrusión), para lo cual se utilizará alguna herramienta especializada.

4.5.1.3. Pruebas de carga y rendimiento

Una vez finalizadas las pruebas funcionales y de seguridad, se procederá a realizar un ciclo de pruebas de carga y rendimiento del sistema. Estas pruebas las coordinará el gestor de proyecto, que deberá contar con el soporte técnico del área de Explotación.

El objetivo de estas pruebas es analizar la respuesta del sistema bajo situaciones de carga iguales o superiores a las que se esperan en el entorno de producción, observando que:

- El rendimiento del sistema (en cuanto a tiempo de respuesta al usuario) es adecuado, y no se degrada excesivamente en situaciones de alta carga
- No se produzca agotamiento de recursos en las máquinas
- Al finalizar la situación de carga, se liberen los recursos y no haya, por tanto, filtraciones (*leaks*)

Si se producen alguna de estas situaciones indeseadas, se deberá analizar el sistema para intentar solucionar estos problemas antes de liberar el producto.

Para la realización de estas pruebas y el análisis de resultados se recomienda utilizar herramientas especializadas (véase anexo B).

4.5.1.4. Pruebas de aceptación

Una vez realizadas las pruebas reseñadas en los dos puntos anteriores y subsanados los posibles problemas encontrados, se realizará una sesión de validación y aceptación del producto final. En esta sesión participarán:

- El (los) definidor-validador(es)
- El gestor del proyecto
- El equipo de proyecto

Esta sesión, que no deberá exceder en duración de una jornada laboral en ningún caso, tiene como objetivo la revisión del sistema completo en cuanto a su funcionalidad, para que el (los) definidor-validador(es) puedan dar el visto bueno definitivo al producto.

Dado que MEDRA^{A} es una metodología ágil y, por tanto, adaptativa, el producto desarrollado ha debido ir adaptándose a lo largo del ciclo de vida del proyecto a los posibles cambios en las necesidades y requisitos del usuario; por tanto, no cabe esperar que haya desviaciones significativas de alcance.

Si en esta sesión surgen solicitudes de cambio por parte del (los) definidor-validador(es), el gestor de proyecto deberá negociar con él (ellos) la planificación de estos cambios como cambios *a posteriori* que serán gestionados según el procedimiento establecido de "gestión del cambio" conforme al Plan de Gestión de Servicios de IT CGAE (*véase el documento PGS.11 Gestión de Cambios-Vx.docx, donde* ***x*** *representa la última versión liberada del mismo*).

Finalmente, el (los) definidor-validador(es) deberá(n) dar explícitamente (mediante un acta) su aprobación al producto final entregado.

4.5.2. Elaboración de manuales y formación a usuarios

Como parte de la entrega final el equipo de proyecto deberá aportar los manuales del sistema necesarios, entre los que se encontrarán:

- Manual de usuario
- Manual de explotación y seguridad

Los documentos se realizarán conforme a las plantillas de documentación estándar definidas en el Plan de Gestión de Servicios de IT CGAE.

Debe tenerse en cuenta que aunque esta actividad se describa en este punto, la elaboración de los manuales puede y debe llevarse a cabo en paralelo a las actividades 4.5.1 (*Pruebas finales y de aceptación del sistema*) y 4.5.3 (*Preparación de la entrega final*).

Adicionalmente, y en los casos en que así se requiera, el equipo de proyecto deberá encargarse de impartir formación en el uso del sistema, bien directamente a los usuarios finales del mismo, bien a un equipo de formadores que a su vez impartan la formación a los usuarios.

La formación puede obviarse si la sencillez e *intuitividad* del sistema la hacen innecesaria, o si se han preparado mecanismos de autoformación, por ejemplo, a través de la plataforma de formación de IT CGAE.

4.5.3. Preparación de la entrega final

Una vez finalizado y aprobado el sistema se preparará el entregable para su despliegue en el entorno de producción.

La confección y gestión de la entrega deberá realizarse conforme a los procedimientos establecidos en el Plan de Gestión de Servicios de IT CGAE.

Véanse los siguientes documentos (**x** representa el último número de versión liberada para cada documento):

- PGS.12 Gestión de Entrega y Despliegue-Vx.docx
- RPGS.12-01 Plan de Entregas.docx
- RPGS.12-02 Formato de informe de entrega.docx
- Modelo estándar de documento de entrega de IT CGAE
- ITPGS.12-01-Proceso Gestión de Entregas de Software-Vx_xx.docx

Una vez el sistema está desplegado en producción, se considera finalizado el proyecto de desarrollo, y el software producido pasa a ser un elemento de configuración (*CI, Configuration Item*) controlado dentro de la CMDB (*Configuration Management Data Base*) y sujeto al procedimiento de gestión del cambio, tal y como se contempla en el Plan de Gestión de Servicios de IT CGAE (*véase el documento PGS.11 Gestión de Cambios-Vx.docx, donde **x** representa la última versión liberada del mismo*).

5. Metodología para el paradigma en cascada - MEDRA^C

Para aquellos proyectos en los que se determine que el enfoque metodológico más adecuado es el paradigma "en cascada", se aplicará una metodología de trabajo basada en Métrica v3, según se describe en este apartado.

Métrica v3 es una metodología definida por el (en su día) Ministerio de Administraciones Públicas, diseñada para cubrir todo el ciclo de vida del *software* y cuyos objetivos son:

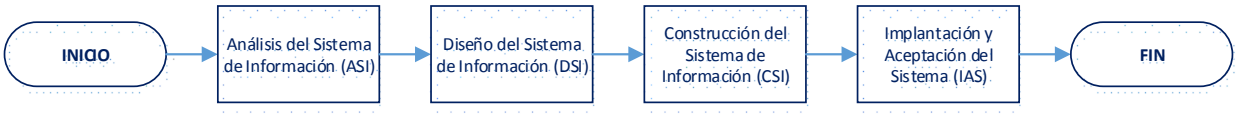
- Proporcionar o definir Sistemas de Información que ayuden a conseguir los fines de la organización mediante la definición de un marco estratégico para el desarrollo de los mismos
- Dotar a la organización de productos software que satisfagan las necesidades de los usuarios dando una mayor importancia al análisis de requisitos
- Mejorar la productividad de los departamentos de Sistemas y Tecnologías de la Información y las Comunicaciones, permitiendo una mayor capacidad de adaptación a los cambios y teniendo en cuenta la reutilización en la medida de lo posible
- Facilitar la comunicación y entendimiento entre los distintos participantes en la producción de software a lo largo del ciclo de vida del proyecto, teniendo en cuenta su papel y responsabilidad, así como las necesidades de todos y cada uno de ellos
- Facilitar la operación, mantenimiento y uso de los productos software obtenidos

Si bien Métrica v3 cubre todo el ciclo de vida del software, la metodología de desarrollo MEDRA se apoya únicamente en el *Proceso de Desarrollo de Sistemas de Información*, dejando por tanto al margen los procesos de *Planificación de Sistemas de Información (PSI)* y *Mantenimiento de Sistemas de Información (MSI)*.

La metodología descrita en el presente apartado no pretende ser una trasposición literal de dicho proceso, sino que adopta las líneas maestras del mismo y las adapta a la idiosincrasia de IT CGAE y sus clientes (CGAE, ICAs y abogados).

Concretamente, se omiten las partes de Métrica v3 que hacen referencia a los desarrollos de tipo "estructurado", ya que todos los desarrollos realizados en IT CGAE se acogen al paradigma de "orientación a objetos".

Por brevedad, denominaremos MEDRA^C a la metodología en cascada contemplada dentro de MEDRA. El siguiente diagrama ilustra su flujo a alto nivel:



Cabe destacar que en esta metodología los participantes trabajarán en un equipo organizado siguiendo los roles "tradicionales" en los proyectos de TI: jefe de proyecto, analista funcional, analista-programador, programador, probador, etc.

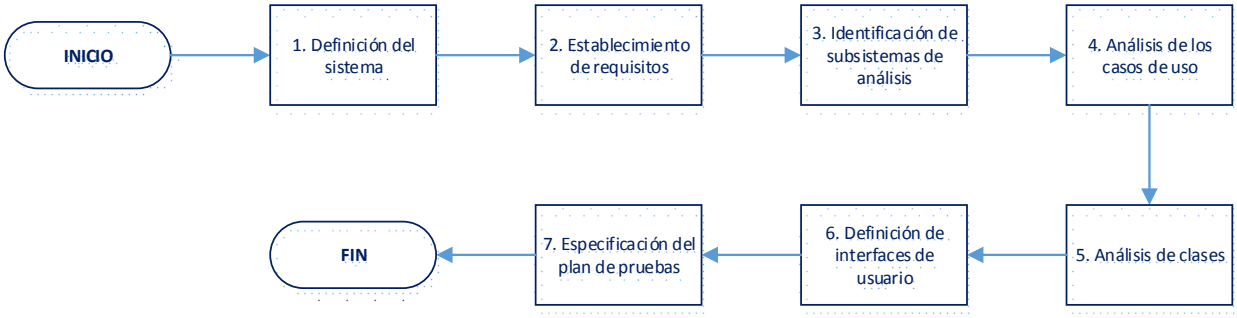
Las tareas asignadas a cada participante a lo largo de las distintas fases del desarrollo se registrarán en la herramienta de gestión de tareas correspondiente para permitir su seguimiento; en caso de proyectos desarrollados internamente en ITCGAE, se utilizará la herramienta designada para el área de Desarrollo; en caso de proyectos subcontratados, el proveedor podrá utilizar su propia herramienta de gestión de tareas.

En cada una de las tareas descritas en la metodología se proponen las técnicas a utilizar para su realización. Las herramientas concretas a utilizar para cada tarea o actividad se recogen en el anexo B.

5.1. Análisis del Sistema de Información (ASI)

El objetivo de esta fase es la obtención de una especificación detallada del sistema de información que satisfaga las necesidades de información de los usuarios y sirva de base para el posterior diseño del sistema. Dicha especificación deberá ser validada con la(s) persona(s) que representen a los clientes / usuarios y que tengan la competencia para aprobar dicha especificación antes de pasar a la fase de diseño.

El flujo de esta fase queda representado por el siguiente diagrama:



5.1.1. Definición del sistema

Esta actividad tiene como objetivo efectuar una descripción del sistema, delimitando su alcance, estableciendo las interfaces con otros sistemas e identificando a los usuarios representativos.

Esta actividad se descompone a su vez en las siguientes tareas:

5.1.1.1. Determinación del alcance del sistema

En esta tarea se delimita el sistema de información, indicando qué procesos pertenecen al ámbito del Sistema de Información y se identifican las entidades externas al sistema que aportan o reciben información. Asimismo, se obtiene un modelo conceptual de datos identificando las principales entidades y relaciones que forman parte del sistema de información objeto de este análisis.

Entradas:

- Reuniones de definición
- Descripción general de la necesidad
- RFQ (*Request For Quotation*)

Salidas:

- Apartado de “Alcance del sistema” (dentro del documento de Plan de proyecto). Incluirá:
  - Descripción textual del alcance
  - Diagramas UML que lo clarifiquen
- Se registrará una petición de cambio en EasyVista:
  - Si el proyecto va a dar lugar a un nuevo sistema o servicio, será de tipo “Nuevo servicio”
  - Si el proyecto va a dar lugar a una nueva versión de un sistema o servicio, incluyendo un número relevante de cambios en el mismo, será de tipo “Modificación relevante de un servicio”

Técnicas:

- Reuniones de trabajo
- Análisis de legislación / normativa aplicable
- Diagramación UML

Participantes:

- Jefe de proyecto
- Analista(s) funcional(es)
- Representantes (a nivel directivo) de los usuarios / clientes

5.1.1.2. Identificación del Entorno Tecnológico

El objetivo de esta tarea es definir, a alto nivel, el entorno tecnológico que se requiere para dar respuesta a las necesidades de información, especificando sus posibles condicionantes y restricciones.

Entradas:

- Apartado de "Alcance del sistema" (dentro del documento de Plan de proyecto)
- Documentos de arquitectura general de IT CGAE

Salidas:

- Apartado de "Arquitectura general del sistema" (dentro del documento de Plan de proyecto). Incluirá:
  - Descripción textual del entorno tecnológico
  - Diagramas que lo clarifiquen

Técnicas:

- Reuniones de trabajo
- Diagramación

Participantes:

- Jefe de proyecto
- Analista(s) funcional(es)
- Equipo de arquitectura de IT CGAE (área de Explotación)

5.1.1.3. Especificación de estándares y normas

La realización de esta tarea permite considerar las referencias para el sistema de información en estudio, desde el punto de vista de estándares, normativas, leyes o recomendaciones, que deben tenerse en cuenta a lo largo de todo el proceso de desarrollo.

Entradas:

- Apartado de "Alcance del sistema" (dentro del documento de Plan de proyecto)
- Catálogo de normas de IT CGAE
- Legislación aplicable

Salidas:

- Apartado de "Normativa y legislación" (dentro del documento de Plan de proyecto)

Técnicas:

- Análisis de legislación / normativa aplicable

Participantes:

- Jefe de proyecto
- Analista(s) funcional(es)
- Representantes (a nivel directivo) de los usuarios / clientes

5.1.1.4. Identificación de los participantes

En esta tarea se identifican los usuarios participantes y finales, interlocutores tanto en la obtención de requisitos como en la validación de los distintos productos y la aceptación final del sistema, así como el equipo de trabajo del proyecto.

Entradas:

- Apartado de "Alcance del sistema" (dentro del documento de Plan de proyecto)
- Apartado de "Normativa y legislación" (dentro del documento de Plan de proyecto)

Salidas:

- Apartado de "Equipo de proyecto" (dentro del documento de Plan de proyecto)
- Apartado de "Participantes del proyecto" (dentro del documento de Plan de proyecto)

Técnicas:

- Reuniones

Participantes:

- Jefe de proyecto
- Representantes (a nivel ejecutivo) de los usuarios / clientes

5.1.1.5. Planificación de tareas de desarrollo

En esta tarea se establece una planificación inicial de las tareas a realizar en el desarrollo del sistema software, desde la actividad de establecimiento de requisitos hasta la entrega en producción.

Esta planificación se realizará en base a la información recabada en las tareas anteriores a la presente, y apoyándose en la experiencia del jefe de proyecto y del equipo. Dado que nos encontramos en una fase muy inicial del proceso, esta planificación será aproximada, y deberá irse corrigiendo a medida que avance el proceso de análisis. Una vez terminadas las fases de análisis (ASI) y diseño (DSI), la planificación debería ser mucho más precisa y realista y el equipo de proyecto debería comprometerse a su cumplimiento.

Entradas:

- Apartado de "Alcance del sistema" (dentro del documento de Plan de proyecto)
- Apartado de "Arquitectura general del sistema" (dentro del documento de Plan de proyecto)
- Apartado de "Equipo de proyecto" (dentro del documento de Plan de proyecto)

Salidas:

- Apartado de "Planificación y cronograma" (dentro del documento de Plan de proyecto)

Técnicas:

- Reuniones
- Estimación de tareas
- Planificación de proyectos (Gantt, PERT, WBS, etc.)

Participantes:

- Jefe de proyecto
- Equipo de proyecto

5.1.2. Establecimiento de requisitos

En esta actividad se lleva a cabo la definición, análisis y validación de los requisitos a partir de la información facilitada por el usuario. El objetivo de esta actividad es obtener un catálogo detallado de los requisitos, a partir del cual se pueda comprobar que los productos generados en las actividades de modelización se ajustan a los requisitos de usuario.

Esta actividad se descompone en las siguientes tareas:

5.1.2.1. Obtención de requisitos

En esta tarea comienza la obtención detallada de información mediante sesiones de trabajo con los usuarios, previamente identificados en la actividad Definición del sistema.

Se recoge información de los requisitos que debe cumplir el software. En la definición de los requisitos, que sirven de base para establecer los niveles de servicios del sistema, hay que tener en cuenta, si existen, las posibles restricciones del entorno, tanto hardware como software, que puedan afectar al sistema de información.

También se definen las prioridades que hay que asignar a los requisitos, considerando los criterios de los usuarios acerca de las funcionalidades a cubrir.

Los principales tipos de requisitos que se deben especificar son:

- Funcionales
- No funcionales:
    - De nivel de servicio (tiempos de respuesta, rendimiento, disponibilidad)
    - De usabilidad y accesibilidad
    - De interoperabilidad con otros sistemas
    - De confidencialidad de la información, atendiendo, entre otros, a los criterios de la LOPD
    - De seguridad. Entre estos se deberán considerar específicamente:
        - Mecanismos de identificación, autenticación y autorización
        - Mecanismos de protección de la información tratada. En particular, y en relación con la confidencialidad de los datos, se analizará la necesidad o conveniencia del cifrado de datos
        - Generación y tratamiento de pistas de auditoría

Los requisitos recopilados deberán registrarse en la herramienta corporativa de gestión de requisitos (véase anexo B)

Se especificarán, además, los casos de uso asociados a los requisitos funcionales, identificando:

- Actores.
- Casos de uso.
- Breve descripción de cada caso de uso.

Entradas:

- Apartado de "Alcance del sistema" (dentro del documento de Plan de proyecto)
- Apartado de "Descripción general del entorno tecnológico del sistema"
- Apartado de "Normativa y legislación" (dentro del documento de Plan de proyecto)
- Legislación / normativa aplicable

Salidas:

- Catálogo de requisitos
- Modelo de casos de uso

Técnicas:

- Entrevistas con usuarios / clientes
- Observación directa del trabajo
- *Brainstorming*
- Prototipado en papel
- Gestión de requisitos con herramienta corporativa

Participantes:

- Analista(s) funcional(es)
- Usuarios / clientes

5.1.2.2. Especificación de casos de uso

El objetivo de esta tarea es especificar cada caso de uso identificado en la tarea anterior, desarrollando el escenario.

Para completar los casos de uso, es preciso especificar información relativa a:

- Descripción del escenario, es decir, cómo un actor interactúa con el sistema, y cuál es la respuesta obtenida
- Precondiciones y postcondiciones
- Identificación de interfaces de usuario.
- Condiciones de fallo que afectan al escenario, así como la respuesta del sistema (escenarios secundarios)

Entradas:

- Apartado de "Alcance del sistema" (dentro del documento de Plan de proyecto)
- Catálogo de requisitos
- Modelo de casos de uso

Salidas:

- Especificación de casos de uso (dentro del documento de análisis)

Técnicas:

- Reuniones
- Casos de uso
- Diagramación UML

Participantes:

- Analista(s) funcional(es)
- Usuarios / clientes

5.1.2.3. Análisis de Requisitos

En esta tarea se estudia la información capturada previamente en esta actividad, para detectar inconsistencias, ambigüedades, duplicidad o escasez de información, etc.

También se analizan las prioridades establecidas por el usuario y se asocian los requisitos relacionados entre sí.

El análisis de los requisitos y de los casos de uso asociados permite identificar funcionalidades o comportamientos comunes, reestructurando la información de los casos de uso a través de las generalizaciones y relaciones entre ellos.

Mediante sesiones de trabajo con los usuarios, se contrastan las conclusiones del análisis de la información recogida.

Entradas:

- Catálogo de requisitos
- Modelo de casos de uso
- Especificación de casos de uso (dentro del documento de análisis)

Salidas:

- Catálogo de requisitos
- Modelo de casos de uso
- Especificación de casos de uso (dentro del documento de análisis)

Técnicas:

- Reuniones
- Análisis de requisitos
- Análisis de casos de uso
- Diagramación UML
- Gestión de requisitos con herramienta corporativa

Participantes:

- Analista(s) funcional(es)
- Usuarios / clientes

5.1.2.4. Validación de requisitos

Mediante esta tarea, los usuarios confirman que los requisitos especificados en el catálogo de requisitos, así como los casos de uso, son válidos, consistentes y completos.

Entradas:

- Catálogo de requisitos
- Modelo de casos de uso
- Especificación de casos de uso (dentro del documento de análisis)

Salidas:

- Catálogo de requisitos (validado)
- Modelo de casos de uso (validado)
- Especificación de casos de uso (dentro del documento de análisis) (validado)

Técnicas:

- Reuniones
- Diagramación UML
- Gestión de requisitos con herramienta corporativa

Participantes:

- Analista(s) funcional(es)
- Usuarios / clientes

5.1.3. Identificación de subsistemas de análisis

El objetivo de esta actividad es facilitar el análisis del sistema de información llevando a cabo la descomposición del sistema en subsistemas. Se realiza en paralelo con el resto de las actividades de generación de modelos del análisis. Por tanto, se asume la necesidad de una realimentación y ajuste continuo con respecto a la definición de los subsistemas, sus dependencias y sus interfaces.

Esta actividad se descompone en las siguientes tareas:

5.1.3.1. Determinación de subsistemas de análisis

La descomposición del sistema en subsistemas debe estar, principalmente, orientada a los procesos de negocio, aunque también es posible adoptar otros criterios lógicos. Entre los criterios que pueden ayudar a su identificación, se encuentran los siguientes:

- Homogeneidad de procesos
- Servicios comunes
- Prioridad
- Afinidad de requisitos
- Localización geográfica

En análisis orientado a objetos, se identifican y definen las dependencias entre subsistemas analizando los elementos compartidos entre ellos o las interfaces entre subsistemas. En el caso de que se decida abstraer un subsistema para su análisis como una unidad con una funcionalidad concreta, se puede, opcionalmente, definir la interfaz de dicho subsistema para poder delimitar su comportamiento y utilización en el modelo general del sistema. Por tanto, se establece como obligatoria la asociación entre subsistemas indicando sólo la dependencia. Además, opcionalmente, se propone la especificación de la interfaz de subsistemas de análisis, y la definición del comportamiento del sistema.
Se asignarán los requisitos y casos de uso a cada uno de los subsistemas identificados, actualizando el catálogo de requisitos.

Entradas:

- Apartado de “Alcance del sistema” (dentro del documento de Plan de proyecto)
- Modelo de casos de uso
- Especificación de casos de uso (dentro del documento de análisis)

Salidas:

- Descripción de subsistemas de análisis
- Descripción de interfaces entre subsistemas

Técnicas:

- Diagrama de Flujo de Datos
- Diagrama de Paquetes (Subsistemas)
- Gestión de requisitos con herramienta corporativa

Participantes:

- Jefe de proyecto
- Analista(s) / Diseñador(es)

5.1.3.2. Integración de subsistemas de análisis

El objetivo de esta tarea es la coordinación en la elaboración de los distintos modelos de análisis de cada subsistema, asegurando la ausencia de duplicidad de elementos y la precisión en la utilización de los términos del glosario. Esta tarea se realiza en paralelo con el resto de las actividades de elaboración de modelos del análisis, y permite tener una visión global y unificada de los distintos modelos.

Como consecuencia de la coordinación de modelos, se pueden identificar elementos comunes con posible implicación en la propia definición de subsistemas y en sus dependencias o interfaces.

Entradas:

- Descripción de subsistemas de análisis
- Descripción de interfaces entre subsistemas

Salidas:

- Descripción de subsistemas de análisis
- Descripción de interfaces entre subsistemas

Técnicas:

- Análisis de subsistemas
- Diagrama de Flujo de Datos
- Diagrama de Paquetes (Subsistemas)

Participantes:

- Jefe de proyecto
- Analista(s) / Diseñador(es)

5.1.4. Análisis de los casos de uso

El objetivo de esta actividad es identificar las clases cuyos objetos son necesarios para realizar un caso de uso y describir su comportamiento mediante la interacción dichos objetos. Esta actividad se lleva a cabo para cada uno de los casos de uso contenidos en un subsistema de los definidos en la actividad *Identificación de subsistemas de análisis*.

Las tareas de esta actividad no se realizan de forma secuencial sino en paralelo, con continuas realimentaciones entre ellas y con las realizadas en las actividades *Establecimiento de requisitos*, *Identificación de subsistemas de análisis*, *Análisis de clases* y *Definición de interfaces de usuario*.

Esta actividad se descompone en las siguientes tareas:

5.1.4.1. Identificación de clases asociadas a un caso de uso

En esta tarea se comienzan a identificar los objetos necesarios para realizar el caso de uso, basándose en la especificación que tenemos del mismo. A partir del estudio del caso de uso, se extrae una lista de objetos candidatos a ser clases. Es posible que, inicialmente, no se disponga de la información necesaria para identificar todas, por lo que se hace una primera aproximación que se va refinando posteriormente, durante esta actividad y en el proceso de diseño. Además, algunos de los objetos representan mejor la información del sistema si se les identifica como atributos en vez de como clases. Para poder diferenciarlas, es necesario estudiar el comportamiento de esos objetos en el diagrama de interacción y además se debe tener en cuenta una serie de reglas, como puede ser el suprimir palabras no pertinentes, con significados vagos o sinónimos. Una vez definidas cada una de las clases, se incorporan al modelo de clases de la actividad *Análisis de clases*, donde se identifican sus atributos, responsabilidades y relaciones.

Las clases que se identifican en esta tarea pueden ser:

- Clases de Entidad (representan la información manipulada en el caso de uso)
- Clases de Interfaz de Usuario (se utilizan para describir la interacción entre el sistema y sus actores. Suelen representar abstracciones de ventanas, interfaces de comunicación, formularios, etc.)
- Clases de Control (son responsables de la coordinación, secuencia de transacciones y control de los objetos relacionados con un caso de uso)

Entradas:

- Modelo de casos de uso
- Especificación de casos de uso

Salidas:

- Modelo de clases de análisis

Técnicas:

- Análisis de clases
- Diagrama de clases UML

Participantes:

- Analista(s) / Diseñador(es)

5.1.4.2. Descripción de la interacción de objetos

El objetivo de esta tarea es describir la cooperación entre los objetos utilizados para la realización de un caso de uso, que ya fueron identificados en la tarea anterior. Para representar esta información, se usan diagramas de interacción que contienen instancias de los actores participantes, objetos, y la secuencia de mensajes intercambiados entre ellos. Se pueden establecer criterios para determinar qué tipo de objetos y mensajes se va a incluir en este diagrama, como por ejemplo: si se incluyen objetos y llamadas a bases de datos, objetos de interfaz de usuario, de control, etc. Estos diagramas pueden ser tanto de secuencia como de colaboración, y su uso depende de si se quieren centrar en la secuencia cronológica o en cómo es la comunicación entre los objetos.

En aquellos casos en los que se especifique más de un escenario para un caso de uso, puede ser conveniente representar cada uno de ellos en un diagrama de interacción. También es recomendable, sobre todo en el caso anterior, completar los diagramas con una descripción textual.

Entradas:

- Modelo de casos de uso
- Especificación de casos de uso

Salidas:

- Análisis de la realización de los casos de uso

Técnicas:

- Análisis de casos de uso
- Diagrama de Interacción de objetos (de secuencia o de colaboración)

Participantes:

- Analista(s) / Diseñador(es)

5.1.5. Análisis de clases

El objetivo de esta actividad es describir cada una de las clases que ha surgido, identificando las responsabilidades que tienen asociadas, sus atributos, y las relaciones entre ellas. Para esto, se debe tener en cuenta la normativa establecida en la tarea *Especificación de estándares y normas*, de forma que el modelo de clases cumpla estos criterios, con el fin de evitar posibles inconsistencias en el diseño.

Teniendo en cuenta las clases identificadas en la actividad *Análisis de los casos de uso*, se elabora el modelo de clases para cada subsistema. A medida que avanza el análisis, dicho modelo se va completando con las clases que vayan apareciendo, tanto del estudio de los casos de uso, como de la interfaz de usuario necesaria para el sistema de información.

Esta actividad se descompone en las siguientes tareas:

5.1.5.1. Identificación de responsabilidades y atributos

El objetivo de esta tarea es identificar las responsabilidades y atributos relevantes de una clase. Las responsabilidades de una clase definen la funcionalidad de esa clase, y están basadas en el estudio de los papeles que desempeñan sus objetos dentro de los distintos casos de uso. A partir de estas responsabilidades, se puede comenzar a encontrar las operaciones que van a pertenecer a la clase. Estas deben ser relevantes, simples, y participar en la descripción de la responsabilidad.

Los atributos de una clase especifican propiedades de la clase, y se identifican por estar implicados en sus responsabilidades. Los tipos de estos atributos deberían ser conceptuales y conocidos en el dominio.

De manera opcional, se elabora una especificación para cada clase, que incluye: la lista de sus operaciones y las clases que colaboran para cubrir esas operaciones y una descripción de las responsabilidades, atributos y operaciones de esa clase.

Para aquellas clases cuyo comportamiento dependa del estado en el que se encuentren se realiza, también de manera opcional, un diagrama de transición de estados.

Entradas:

- Modelo de casos de uso
- Especificación de casos de uso
- Modelo de clases de análisis

Salidas:

- Modelo de clases de análisis
- Comportamiento de clases de análisis

Técnicas:

- Diagrama de clases
- Diagrama de transición de estados

Participantes:

- Analista(s) / Diseñador(es)

5.1.5.2. Identificación de asociaciones y agregaciones

En esta tarea se estudian los mensajes establecidos entre los objetos del diagrama de interacción para determinar qué asociaciones existen entre las clases correspondientes. Estas asociaciones suelen corresponderse con expresiones verbales incluidas en las especificaciones.

Las relaciones surgen como respuesta a las demandas en los distintos casos de uso, y para ello puede existir la necesidad de definir agregaciones y herencia entre objetos. Una asociación está caracterizada por:

- Los papeles que desempeña
- Su direccionalidad, que representa el sentido en el que se debe interpretar
- Su cardinalidad, que representa el número de instancias implicadas en la asociación

Dichas características pueden obtenerse a partir de la especificación de los casos de uso.

A medida que se establecen las relaciones entre las clases, se revisa la especificación de subsistemas de análisis en la actividad *Identificación de subsistemas de análisis*, para conseguir optimizar los subsistemas.

Entradas:

- Modelo de casos de uso
- Especificación de casos de uso
- Análisis de la realización de los casos de uso
- Modelo de clases de análisis

Salidas:

- Modelo de clases de análisis

Técnicas:

- Diagrama de clases

Participantes:

- Analista(s) / Diseñador(es)

5.1.5.3. Identificación de generalizaciones

El objetivo de esta tarea es representar una organización de las clases que permita una implementación sencilla de la herencia y una agrupación semántica de las diferentes clases, basándose siempre en las normas y estándares definidos en la actividad *Definición del sistema*.

Entradas:

- Modelo de clases de análisis

Salidas:

- Modelo de clases de análisis

Técnicas:

- Diagrama de clases

Participantes:

- Analista(s) / Diseñador(es)

5.1.6. Definición de interfaces de usuario

En esta actividad se especifican las interfaces entre el sistema y el usuario: formatos de pantallas, diálogos, e informes, principalmente. El objetivo es realizar un análisis de los procesos del sistema de información en los que se requiere una interacción del usuario, con el fin de crear una interfaz que satisfaga todos los requisitos establecidos, teniendo en cuenta los diferentes perfiles a quiénes va dirigido.

Al comienzo de este análisis es necesario seleccionar el entorno en el que es operativa la interfaz, considerando estándares internacionales y de la instalación, y establecer las directrices aplicables en los procesos de diseño y construcción. El propósito es construir una interfaz de usuario acorde a sus necesidades, flexible, coherente, eficiente y sencilla de utilizar, teniendo en cuenta la facilidad de cambio a otras plataformas, si fuera necesario.

Se identifican los distintos grupos de usuarios de acuerdo con las funciones que realizan, conocimientos y habilidades que poseen, y características del entorno en el que trabajan. La identificación de los diferentes perfiles permite conocer mejor las necesidades y particularidades de cada uno de ellos.

Asimismo, se determina la naturaleza de los procesos que se llevan a cabo (en lotes o en línea). Para cada proceso en línea se especifica qué tipo de información requiere el usuario para completar su ejecución realizando, para ello, una descomposición en diálogos que refleje la secuencia de la interfaz de pantalla tipo carácter o pantalla gráfica.

Finalmente, se define el formato y contenido de cada una de las interfaces de pantalla especificando su comportamiento dinámico.

Como resultado de esta actividad se genera la especificación de interfaz de usuario, como producto que engloba los siguientes elementos:

- Principios generales de la interfaz
- Catálogo de perfiles de usuario
- Descomposición funcional en diálogos
- Catálogo de controles y elementos de diseño de interfaz de pantalla
- Formatos individuales de interfaz de pantalla
- Modelo de navegación de interfaz de pantalla

- Formatos de impresión
- Prototipo de interfaz interactiva
- Prototipo de interfaz de impresión

Esta actividad se descompone en las siguientes tareas:

5.1.6.1. Especificación de principios generales de la interfaz

El objetivo de esta tarea es especificar los estándares, directrices y elementos generales a tener en cuenta en la definición de la interfaz de usuario, tanto para la interfaz interactiva (gráfica o carácter), como para los informes y formularios impresos.

En primer lugar, se selecciona el entorno de la interfaz interactiva (gráfico, carácter, etc.), siguiendo estándares internacionales y de la instalación, y se determinan los principios de diseño de la interfaz de usuario, contemplando:

- Directrices generales en cuanto a la interfaz y aspectos generales de interacción
- Principios de composición de pantallas y criterios de ubicación de los distintos elementos dentro de cada formato
- Normas para los mensajes de error y aviso, codificación, presentación y comportamientos
- Normas para la presentación de ayudas

Hay que establecer criterios similares para la interfaz impresa:

- Directrices generales
- Principios de composición de informes y formularios
- Normas de elaboración, distribución y salvaguarda de la información

Entradas:

- Apartado de “Descripción general del entorno tecnológico del sistema” (dentro del documento de Plan de proyecto)
- Apartado de “Normativa y legislación” (dentro del documento de Plan de proyecto)

Salidas:

- Especificación de Interfaz de Usuario: Principios Generales de la Interfaz

Técnicas:

- Reuniones
- Bosquejos en papel

Participantes:

- Analista(s) / Diseñador(es)
- Usuarios / clientes

5.1.6.2. Especificación de formatos individuales de la interfaz de pantalla

El objetivo de esta tarea es especificar cada formato individual de la interfaz de pantalla, desde el punto de vista estático. Para cada proceso en línea identificado en la tarea anterior o en la especificación de los casos de uso, y teniendo en cuenta los formatos estándar definidos en la tarea *Especificación de principios generales de la interfaz*, se definen los formatos individuales de la interfaz de pantalla requerida para completar la especificación de cada diálogo. Estos formatos individuales van completando las especificaciones de los casos de uso.

También se considera el catálogo de requisitos, para especificar las interfaces relacionadas con las consultas.

En la definición de cada interfaz de pantalla se deben definir aquellos aspectos considerados de interés para su posterior diseño y construcción:

- Posibilidad de cambio de tamaño, ubicación, modalidad (modal del sistema, modal de aplicación), etc.
- Dispositivos de entrada necesarios para su ejecución.
- Conjunto y formato de datos asociados, identificando qué datos se usan y cuáles se generan como consecuencia de su ejecución.
- Controles y elementos de diseño asociados, indicando cuáles aparecen inicialmente activos e inactivos al visualizar la interfaz de pantalla

Entradas:

- Especificación de Interfaz de Usuario: Principios Generales de la Interfaz
- Modelo de casos de uso
- Especificación de casos de uso

Salidas:

- Especificación de Interfaz de Usuario:
  - Formatos individuales de interfaz de pantalla
  - Catálogo de controles y elementos de diseño de interfaz de pantalla

Técnicas:

- Reuniones
- Bosquejos en papel
- Prototipado

Participantes:

- Analista(s) / Diseñador(es)
- Usuarios / clientes

5.1.6.3. Especificación del comportamiento dinámico de la interfaz

El objetivo de esta tarea es definir los flujos entre los distintos formatos de interfaz de pantalla, y también dentro del propio formato. Este comportamiento se describe mediante un modelo de navegación de interfaz de pantalla.

Para cada formato individual de pantalla o ventana, definido en la tarea *Especificación de formatos individuales de la interfaz de pantalla*, se establece la entrada lógica de los datos y las reglas de validación, incluyendo dependencia de valores (reflejo de los requisitos de validación de sistema)

Se analiza y determina la secuencia de acciones específicas para completar cada diálogo, tal y como se ejecuta en el ámbito de la interfaz, así como las condiciones que se deben cumplir para su inicio, y las posibles restricciones durante su ejecución. El comportamiento está dirigido y representado por los controles y los eventos que provocan su activación.

Se identifican aquellos diálogos o formatos considerados críticos para el correcto funcionamiento del sistema, basándose en el número de usuarios, frecuencia de uso, datos implicados, alcance de las funciones asociadas al diálogo, diálogos comunes a diferentes funciones, marco de seguridad establecido en los requisitos del sistema, etc.

Para los diálogos o comportamientos complejos de interfaz se propone la técnica de diagrama de transición de estados, siendo suficiente en la mayoría de los casos una especificación del comportamiento con matrices control / evento / acción, detallándose la acción con una descripción textual.

Se propone, opcionalmente, la realización de prototipos como técnica de ayuda a la especificación y validación de la interfaz de usuario.

Entradas:

- Especificación de Interfaz de Usuario: Principios Generales de la Interfaz
- Modelo de casos de uso
- Especificación de casos de uso

Salidas:

- Especificación de Interfaz de Usuario:
  - Modelo de navegación de interfaz de pantalla
  - Prototipo de interfaz interactiva

Técnicas:

- Reuniones
- Bosquejos en papel
- Prototipado
- Diagrama de transición de estados
- Diagrama de interacción de objetos

Participantes:

- Analista(s) / Diseñador(es)
- Usuarios / clientes

5.1.6.4. Especificación de formatos de impresión

El objetivo de esta tarea es especificar los formatos y características de las salidas o entradas impresas del sistema.

De acuerdo a los estándares establecidos en la tarea *Especificación de principios generales de la interfaz*, se definen los formatos individuales de informes y formularios, estos últimos si son necesarios, así como sus características principales, entre las que se especifican la periodicidad, confidencialidad, procedimientos de entrega o difusión, y salvaguarda de copia.

Opcionalmente, se recomienda la utilización de prototipos.

Entradas:

- Especificación de Interfaz de Usuario: Principios Generales de la Interfaz
- Modelo de casos de uso
- Especificación de casos de uso

Salidas:

- Especificación de Interfaz de Usuario:
  - Formatos de impresión
  - Prototipo de interfaz de impresión

Técnicas:

- Reuniones
- Bosquejos en papel
- Prototipado

Participantes:

- Analista(s) / Diseñador(es)
- Usuarios / clientes

5.1.7. Especificación del plan de pruebas

En esta actividad se inicia la definición del plan de pruebas, el cual sirve como guía para la realización de las pruebas, y permite verificar que el sistema de información cumple las necesidades establecidas por el usuario, con las debidas garantías de calidad.

El plan de pruebas es un producto formal que define los objetivos de la prueba de un sistema, establece y coordina una estrategia de trabajo, y provee del marco adecuado para elaborar una planificación paso a paso de las actividades de prueba. El plan se inicia en el proceso Análisis del Sistema de Información (ASI), definiendo el marco general, y estableciendo los requisitos de prueba de aceptación, relacionados directamente con la especificación de requisitos.

Dicho plan se va completando y detallando a medida que se avanza en los restantes procesos del ciclo de vida del software, Diseño del Sistema de Información (DSI), Construcción del Sistema de Información (CSI) e Implantación y Aceptación del Sistema (IAS).

Se plantean los siguientes niveles de prueba:

- Pruebas unitarias
- Pruebas de integración
- Pruebas del sistema
- Pruebas de implantación
- Pruebas de aceptación

En esta actividad también se avanza en la definición de las pruebas de aceptación del sistema. Con la información disponible, es posible establecer los criterios de aceptación de las pruebas incluidas en dicho nivel, al poseer la información sobre los requisitos que debe cumplir el sistema, recogidos en el catálogo de requisitos.

Esta actividad se descompone en las siguientes tareas:

5.1.7.1. Definición del alcance de las pruebas

En función de la solución adoptada en el desarrollo de un sistema de información, es posible que determinados niveles de pruebas sean especialmente críticos y otros no sean necesarios. Por ejemplo, puede haber grandes diferencias en función de una solución de desarrollo completo o un producto de mercado cerrado integrado con otros sistemas.

En esta tarea se especifican y justifican de los niveles de pruebas a realizar, así como el marco general de planificación de cada nivel de prueba, según el siguiente esquema:

- Definición de los perfiles implicados en los distintos niveles de prueba
- Planificación temporal
- Criterios de verificación y aceptación de cada nivel de prueba
- Definición, generación y mantenimiento de verificaciones y casos de prueba
- Análisis y evaluación de los resultados de cada nivel de prueba
- Productos a entregar como resultado de la ejecución de las pruebas

Entradas:

- Catálogo de requisitos
- Apartado de “Normativa y legislación” (dentro del documento de Plan de proyecto)
- Apartado de “Descripción general del entorno tecnológico del sistema” (dentro del documento de Plan de proyecto)
- Especificación de Interfaz de Usuario
- Modelo de casos de uso
- Especificación de casos de uso
- Descripción de subsistemas de análisis
- Descripción de interfaces entre subsistemas
- Modelo de clases
- Comportamiento de clases
- Análisis de la realización de los casos de uso

Salidas:

- Plan de pruebas:
    - Especificación de los niveles de pruebas

Técnicas:

- Reuniones

Participantes:

- Jefe de proyecto
- Analista(s) funcional(es)
- Equipo de soporte técnico de IT CGAE
- Usuarios / clientes

5.1.7.2. Definición de requisitos del entorno de pruebas

El objetivo de esta tarea es la definición o recopilación de los requisitos relativos al entorno de pruebas, completando el plan de pruebas.

La realización de las pruebas aconseja disponer de un entorno de pruebas separado del entorno de desarrollo y del entorno de operación, garantizando cierta independencia y estabilidad en los datos y elementos a probar, de modo que los resultados obtenidos sean objetivamente representativos, punto especialmente crítico en pruebas de rendimiento.

No es objeto de MEDRA^{C} en general, ni de esta tarea en particular, la especificación formal de entornos y procedimientos de pruebas en el ámbito de una instalación. Independientemente de la existencia o no de dichos entornos, en esta tarea se inicia la definición de las especificaciones necesarias para la correcta ejecución de las distintas pruebas del sistema de información. Entre ellas podemos citar las siguientes:

- Requisitos básicos de hardware y software base: sistemas operativos, gestores de bases de datos, monitores de teleproceso, etc.
- Requisitos de configuración de entorno: librerías, bases de datos, ficheros, procesos, comunicaciones, necesidades de almacenamiento, configuración de accesos, etc.
- Herramientas auxiliares. Por ejemplo, de extracción de juegos de ensayo, análisis de rendimiento y calidad, etc.
- Procedimientos para la realización de pruebas y migración de elementos entre entornos

Entradas:

- Catálogo de requisitos
- Apartado de “Descripción general del entorno tecnológico del sistema” (dentro del documento de Plan de proyecto)
- Plan de pruebas

Salidas:

- Plan de pruebas:
    - Definición de Requisitos del entorno de pruebas

Técnicas:

- Reuniones

Participantes:

- Jefe de proyecto
- Analista(s) funcional(es)
- Equipo de soporte técnico de IT CGAE
- Equipo de arquitectura de IT CGAE (área de Explotación)

5.1.7.3. Definición de las pruebas de aceptación del sistema

En esta tarea se realiza la especificación de las pruebas de aceptación del sistema, labor fundamental para que el usuario valide el sistema, como último paso, previo a la puesta en explotación.

Se debe insistir, principalmente, en los criterios de aceptación del sistema que sirven de base para asegurar que satisface los requisitos exigidos.

Los criterios de aceptación deben ser definidos de forma clara, prestando especial atención a aspectos como:

- Procesos críticos del sistema
- Rendimiento del sistema
- Seguridad
- Disponibilidad

Entradas:

- Catálogo de requisitos
- Especificación de interfaz de usuario
- Plan de pruebas
- Modelo de casos de uso
- Especificación de casos de uso
- Descripción de subsistemas de análisis
- Descripción de interfaces entre subsistemas
- Modelo de clases
- Comportamiento de clases
- Análisis de la realización de los casos de uso

Salidas:

- Plan de pruebas completo

Técnicas:

- Reuniones

Participantes:

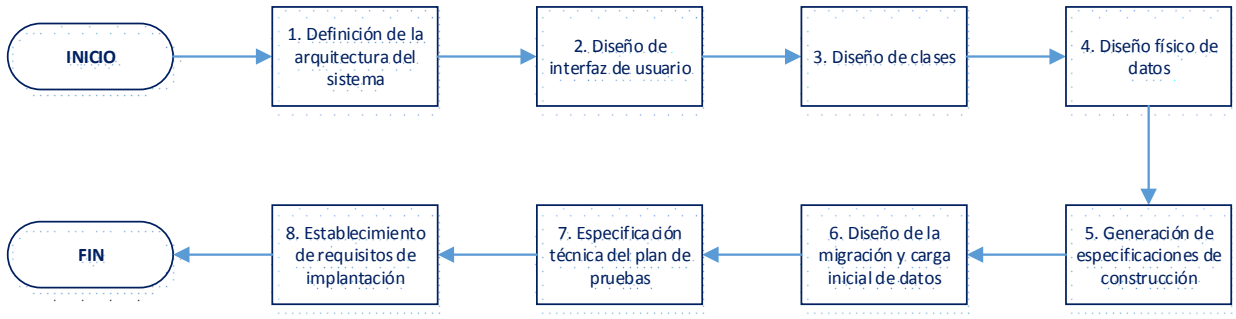
- Jefe de proyecto
- Analista(s) funcional(es)
- Equipo de soporte técnico de IT CGAE
- Usuarios / clientes

5.2. Diseño del Sistema de Información (DSI)

El objetivo del proceso de Diseño del Sistema de Información (DSI) es la definición de la arquitectura del sistema y del entorno tecnológico que le va a dar soporte, junto con la especificación detallada de los componentes del sistema de información.

A partir de dicha información, se generan todas las especificaciones de construcción relativas al propio sistema, así como la descripción técnica del plan de pruebas, la definición de los requisitos de implantación y el diseño de los procedimientos de migración y carga inicial, éstos últimos cuando proceda.

El siguiente diagrama ilustra el flujo de esta fase:



5.2.1. Definición de la arquitectura del sistema

En esta actividad se define la arquitectura general del sistema de información, especificando las distintas particiones físicas del mismo, la descomposición lógica en subsistemas de diseño y la ubicación de cada subsistema en cada partición, así como la especificación detallada de la infraestructura tecnológica necesaria para dar soporte al sistema de información.

El particionamiento físico del sistema de información se especifica identificando los nodos y las comunicaciones entre los mismos, con cierta independencia de la infraestructura tecnológica que da soporte a cada nodo.

Con el fin de organizar y facilitar el diseño, se realiza una división del sistema de información en subsistemas de diseño, como partes lógicas coherentes y con interfaces claramente definidas.

Se establece una distinción entre subsistemas específicos del sistema de información (en adelante, subsistemas específicos) y subsistemas de soporte, con la finalidad de independizar, en la medida de lo posible, las funcionalidades a cubrir por el sistema de información de la infraestructura que le da soporte. En la mayoría de los casos, los subsistemas específicos provienen directamente de las especificaciones de análisis y de los subsistemas de análisis, mientras que los subsistemas de soporte provienen de la necesidad de interacción del sistema de información con la infraestructura y con el resto de los sistemas, así como de la reutilización de módulos o subsistemas ya existentes en la instalación.

Debido a que la definición de los subsistemas de soporte puede exigir la participación de distintos perfiles técnicos, se propone el diseño de ambos tipos de subsistemas en actividades distintas, aunque en paralelo.

Una vez identificados y definidos los distintos subsistemas de diseño, se determina su ubicación óptima de acuerdo a la arquitectura propuesta. La asignación de dichos subsistemas a cada nodo permite disponer, en función de la carga de proceso y comunicación existente entre los nodos, de la información necesaria para realizar una estimación de las necesidades de infraestructura tecnológica que da soporte al sistema de información. Este factor es especialmente crítico en arquitecturas multinivel o cliente/servidor, donde las comunicaciones son determinantes en el rendimiento final del sistema.

Se propone crear un catálogo de excepciones en el que se especifiquen las situaciones anómalas o secundarias en el funcionamiento y ejecución del sistema de información, y que se irá completando a medida que se avance en el diseño detallado de los subsistemas.

En esta actividad también se establecen los requisitos, normas y estándares originados como consecuencia de la adopción de una determinada solución de arquitectura o infraestructura, que serán aplicables tanto en este proceso como en la Construcción del Sistema de Información (CSI).

Se detallan a su vez, de acuerdo a las particularidades de la arquitectura del sistema propuesta, los requisitos de operación, seguridad y control, especificando los procedimientos necesarios para su cumplimiento.

Esta actividad se descompone en las siguientes tareas:

5.2.1.1. Definición de niveles de arquitectura

En esta tarea se describen los niveles de la arquitectura software, mediante la definición de las principales particiones físicas del sistema de información, representadas como nodos y comunicaciones entre nodos.

Se entiende por nodo cada partición física o parte significativa del sistema de información, con características propias de ejecución o función, e incluso de diseño y construcción.

Para facilitar la comprensión del sistema, se recomienda identificar como nodos los elementos de infraestructura más significativos de la arquitectura en la que se va a implementar el sistema de información. Los elementos que se aconseja especificar son los siguientes:

- Gestores de datos
- Tipos de puesto cliente
- Tipos de dispositivos de impresión
- Monitores de teleproceso
- Servidores
- Comunicaciones

La comunicación se expresa por una conexión entre nodos, indicando su carácter bidireccional o unidireccional, con las principales características de los protocolos o tipo de mensajes utilizados.

La especificación de los niveles de la arquitectura se realiza con el detalle suficiente como para permitir un diseño dirigido hacia una solución concreta. En general, no es preciso indicar en cada nodo detalles relativos al hardware, capacidad, rendimiento o configuraciones de tolerancia a fallos, entre otros. Esta información se concreta en la tarea *Especificación del entorno tecnológico*.

Los criterios para diseñar la arquitectura se obtienen a partir de directrices tecnológicas o de integración, propias de la instalación, y del catálogo de requisitos del sistema de información.

Es necesario tener en cuenta, especialmente, aspectos relacionados con:

- Usuarios: ubicación, movilidad, concurrencia, número, etc.
- Datos: variabilidad, volúmenes, necesidades de consolidación, seguridad, etc.
- Procesos: distribución, reutilización, concurrencia, carácter crítico, etc.

Entradas:

- Apartado de "Descripción general del entorno tecnológico del sistema" (dentro del documento de Plan de proyecto)
- Catálogo de requisitos
- Especificación de interfaz de usuario
- Modelo de casos de uso
- Especificación de casos de uso
- Descripción de subsistemas de análisis
- Descripción de interfaces entre subsistemas
- Modelo de clases de análisis
- Análisis de la realización de los casos de uso

Salidas:

- Diseño de la arquitectura del sistema
    - Particionamiento físico del sistema de información

Técnicas:

- Diagrama de despliegue

Participantes:

- Analista(s) - diseñador(es)
- Equipo de arquitectura de IT CGAE (área de Explotación)
- Equipo de Seguridad de IT CGAE

5.2.1.2. Identificación de requisitos de diseño y construcción

En esta tarea se realiza la especificación de los requisitos que están directamente relacionados con la adopción o diseño de una arquitectura o infraestructura concreta, y que pueden condicionar el diseño o la construcción del sistema de información.

Entre estos requisitos pueden estar los relacionados con lenguajes, rendimiento de los distintos elementos de la arquitectura, así como criterios de ubicación de módulos y datos en los distintos nodos.

Por tanto, como resultado de esta tarea se actualiza el catálogo de requisitos elaborado en el proceso Análisis de Sistemas de Información.

Entradas:

- Catálogo de requisitos
- Diseño de la arquitectura del sistema

Salidas:

- Catálogo de requisitos

Técnicas:

- Reuniones
- Gestión de requisitos con herramienta corporativa

Participantes:

- Analista(s) - diseñador(es)
- Equipo de arquitectura de IT CGAE (área de Explotación)
- Equipo de Seguridad de IT CGAE
- Experto en usabilidad

5.2.1.3. Especificación de excepciones

El objetivo de esta tarea es la definición de los comportamientos no habituales en el sistema, que reflejan situaciones anómalas o secundarias en el funcionamiento y ejecución del sistema de información. Para ello, se establece previamente el nivel de especificación de las mismas, así como los criterios de catalogación y clasificación.

Se propone su catalogación como ayuda para el diseño del sistema de información y como guía en la especificación técnica de las pruebas, al permitir la generación de algunos casos de prueba de forma inmediata. Dicho catálogo se va completando a partir de las actividades correspondientes al diseño detallado de los subsistemas.

Las excepciones se describen incluyendo, al menos, los siguientes conceptos:

- Tipo y descripción de la excepción.
- Condiciones previas del sistema de información.
- Elemento afectado (nodo, módulo, caso de uso).
- Respuesta del sistema de información.
- Elemento asociado a la respuesta esperada del sistema (módulo, clase, procedimiento, etc.).

Las excepciones que se proponen como obligatorias son las relacionadas con el funcionamiento general del sistema de información, habitualmente asociadas a:

- Nodos y comunicaciones del particionamiento físico del sistema de información. Este tipo de excepciones tiene lugar cuando no están disponibles los gestores de bases de datos o los recursos compartidos del sistema (representados como nodos), cuando se producen fallos en las comunicaciones entre nodos, etc.
- Rangos o valores no válidos en la entrada de datos, como pueden ser atributos obligatorios, con formatos específicos, etc.

Se recomienda, según el nivel de especificación que se establezca en cada caso, catalogar también las excepciones particulares que se identifiquen en las actividades del diseño de detalle.

Entradas:

- Catálogo de requisitos
- Diseño de la arquitectura del sistema
- Modelo de casos de uso
- Especificación de casos de uso

Salidas:

- Catálogo de excepciones

Técnicas:

- Reuniones

Participantes:

- Analista(s) - diseñador(es)
- Equipo de arquitectura de IT CGAE (área de Explotación)

5.2.1.4. Especificación de estándares y normas de diseño y construcción

En esta tarea se definen los estándares técnicos y de nomenclatura, normas y recomendaciones, que generalmente están relacionados con la adopción o diseño de una arquitectura o infraestructura tecnológica concreta, y que pueden condicionar el diseño o la construcción del sistema de información.

Como resultado de esta tarea, se actualiza el catálogo de normas obtenido en el proceso Análisis del Sistema de Información.

La información recogida en el catálogo se debe tener en cuenta en la elaboración de los productos resultantes del diseño y construcción del sistema de información. El catálogo de normas es, por tanto, producto de entrada en todas las tareas, aunque por sencillez se omite la referencia al mismo.

Entradas:

- Catálogo de normas de IT CGAE
- Apartado de "Normativa y legislación" (dentro del documento de Plan de proyecto)
- Diseño de la arquitectura del sistema

Salidas:

- Catálogo de normas de IT CGAE

Técnicas:

- Reuniones

Participantes:

- Analista(s) - diseñador(es)
- Equipo de arquitectura de IT CGAE (área de Explotación)

5.2.1.5. Identificación de subsistemas de diseño

En esta tarea se divide de forma lógica el sistema de información en subsistemas de diseño, con el fin de reducir la complejidad y facilitar el mantenimiento. Hay que tomar como referencia inicial los subsistemas de análisis especificados en el proceso de Análisis del Sistema de Información (ASI).

La división en subsistemas de diseño se puede realizar con una continuidad directa de los modelos del análisis, o aplicando nuevos criterios de diseño, entre los que es posible citar los siguientes:

- Facilidad de mantenimiento
- Reutilización de elementos del propio sistema o de la instalación
- Optimización de recursos (por ejemplo, líneas de comunicaciones)
- Características de ejecución (en línea o por lotes)
- Funcionalidad común
- Aplicación de patrones genéricos de diseño al nivel de arquitectura

Los subsistemas resultantes se califican como específicos o de soporte, asignando cada subsistema al nodo correspondiente.

Los subsistemas específicos contemplan las funcionalidades propias del sistema de información, mientras que los de soporte cubren servicios comunes, proporcionando un acceso transparente a los distintos recursos. Estos últimos están relacionados con:

- Comunicaciones entre subsistemas
- Gestión de datos (acceso a bases de datos, ficheros, áreas temporales, importación y exportación de datos, sincronización de bases de datos, etc.)
- Gestión de transacciones
- Control y gestión de errores
- Seguridad y control de acceso
- Gestión de interfaz
- Interacción con los recursos propios del sistema

La interacción del sistema de información con la infraestructura que le da soporte, así como con el resto de los sistemas y servicios de la instalación, puede originar la necesidad de nuevos subsistemas, módulos, clases o servicios no especificados en el análisis.

La definición del comportamiento externo de cada subsistema se completa durante el diseño de detalle con la especificación de su interfaz, así como con la dependencia entre subsistemas.

El diseño de detalle de los subsistemas identificados por criterios de optimización y reutilización, puede aconsejar la reorganización y reubicación de los elementos que forman parte de cada subsistema y, a su vez, puede dar lugar a la identificación de nuevos subsistemas de soporte.

La ubicación de subsistemas en nodos y la dependencia entre subsistemas se especifica por medio de técnicas matriciales, o bien en lenguaje natural o pseudocódigo.

Entradas:

- Apartado de “Descripción general del entorno tecnológico del sistema” (dentro del documento de Plan de proyecto)
- Catálogo de requisitos
- Diseño de la arquitectura del sistema
- Descripción de subsistemas de análisis
- Descripción interfaces entre subsistemas

Salidas:

- Diseño de la arquitectura del sistema
    - Descripción de subsistemas de diseño

Técnicas:

- Diagrama de estructura
- Diagrama de interacción de objetos
- Diagrama de paquetes
- Diagrama de despliegue

Participantes:

- Analista(s) - diseñador(es)
- Equipo de arquitectura de IT CGAE (área de Explotación)
- Equipo de seguridad de IT CGAE

5.2.1.6. Especificación del entorno tecnológico

En esta tarea se definen en detalle los distintos elementos de la infraestructura técnica que dan soporte al sistema de información, determinando la implementación concreta de los nodos y comunicaciones especificados en la tarea *Definición de niveles de arquitectura*.

Se propone agrupar los elementos de la infraestructura en los siguientes conceptos:

- Hardware: procesadores, unidades de almacenamiento, estaciones de trabajo, etc.
- Software: sistemas operativos, subsistemas, middleware, gestores de bases de datos, sistemas de ficheros, software de base, herramientas y utilidades de gestión propias del sistema, etc.
- Comunicaciones: diseño de la topología de la red, protocolos, nodos de red, etc.

La definición de los distintos elementos puede generar restricciones técnicas que afecten al diseño o construcción del sistema de información.

Asimismo, se realiza una estimación de la planificación de capacidades (*capacity planning*) o se especifican los parámetros que Explotación y Sistemas precisen para realizar dicha planificación. Se indican, al menos, las necesidades previstas de:

- Almacenamiento: espacio en disco, espacio en memoria, pautas de crecimiento y evolución estimada del sistema de información, etc.
- Procesamiento: número y tipo de procesadores, memoria, etc.
- Comunicaciones: líneas, caudal, capacidades de elementos de red, etc.

Para poder determinar la planificación de capacidades, es necesario conocer los diseños detallados de los módulos / clases y escenarios, incluida la información de control en las comunicaciones, así como el diseño físico de datos optimizado, productos que se están generando en paralelo a esta actividad. También se tienen en cuenta, cuando proceda, las estimaciones de volúmenes de datos propios de la migración y carga inicial de datos.

Entradas:

- Apartado de “Descripción general del entorno tecnológico del sistema” (dentro del documento de Plan de proyecto)
- Catálogo de requisitos
- Diseño de la arquitectura del sistema
- Plan de migración

Salidas:

- Entorno Tecnológico del Sistema:
  - Especificación del entorno tecnológico
  - Restricciones técnicas
  - Estimación de planificación de capacidades

Técnicas:

- Reuniones
- Diagrama de representación

Participantes:

- Analista(s) - diseñador(es)
- Equipo de arquitectura de IT CGAE (área de Explotación)

5.2.1.7. Especificación de requisitos de operación y seguridad

El objetivo de esta tarea es definir los procedimientos de seguridad y operación necesarios para no comprometer el correcto funcionamiento del sistema y garantizar el cumplimiento de los niveles de servicios que exigirá el sistema en cuanto a la gestión de operaciones (procesos por lotes, seguridad, comunicaciones, etc.). Los niveles de servicio se especifican formalmente en el proceso Implantación y Aceptación del Sistema (IAS).

Tomando como referencia los requisitos establecidos para el sistema, y teniendo en cuenta la arquitectura propuesta y las características del entorno tecnológico definido en esta actividad, se lleva a cabo la definición de los requisitos de seguridad y control de acceso necesarios para garantizar la protección del sistema y minimizar el riesgo de pérdida, alteración o consulta indebida de la información. Para ello, se diseñan los procedimientos relacionados con:

- Identificación, autenticación y autorización
- Acceso al sistema y a sus recursos (datos, transacciones, librerías, etc.)
- Mantenimiento de la integridad y confidencialidad de los datos
- Control y registro de accesos al sistema (logs, certificación, pistas de auditoría, etc.)
- Copias de seguridad y recuperación de datos y su periodicidad
- Recuperación ante catástrofes

Asimismo, se definen los requisitos de operación para los distintos elementos del sistema (módulos, clases, estructuras físicas de datos, sistemas de ficheros), que se están elaborando en paralelo a esta actividad, y se diseñan los procedimientos asociados relacionados con:

- Tratamiento en línea (franja horaria/periodos críticos, número máximo de usuarios, etc.)
- Tratamiento por lotes (periodicidad y secuencia de ejecución, interdependencias, petición de ejecución, etc.)
- Control y planificación de trabajos
- Recuperación y reanudación de trabajos
- Distribución de información generada por el sistema, tanto trabajos planificados o bajo petición
- Control y seguimiento del correcto funcionamiento de los procedimientos de *backup* y recuperación utilizados habitualmente

Entradas:

- Catálogo de requisitos
- Diseño de la arquitectura del sistema
- Entorno Tecnológico del Sistema

Salidas:

- Procedimientos de seguridad y control de acceso
- Procedimientos de operación y administración del sistema

Técnicas:

- Reuniones
- Diagrama de representación

Participantes:

- Analista(s) - diseñador(es)
- Equipo de arquitectura de IT CGAE (área de Explotación)
- Equipo de seguridad de IT CGAE

5.2.2. Diseño de casos de uso

Esta actividad tiene como propósito especificar el comportamiento del sistema de información para un caso de uso, mediante objetos o subsistemas de diseño que interactúan, y determinar las operaciones de las clases e interfaces de los distintos subsistemas de diseño.

Para ello, una vez identificadas las clases participantes dentro de un caso de uso, es necesario completar los escenarios que se recogen del análisis, incluyendo las clases de diseño que correspondan y teniendo en cuenta las restricciones del entorno tecnológico, esto es, detalles relacionados con la implementación del sistema. Es necesario analizar los comportamientos de excepción para dichos escenarios. Algunos de ellos pueden haber sido identificados en el proceso de análisis, aunque no se resuelven hasta este momento. Dichas excepciones se añadirán al catálogo de excepciones para facilitar las pruebas.

Algunos de los escenarios detallados requerirán una nueva interfaz de usuario. Por este motivo es necesario diseñar el formato de cada una de las pantallas o impresos identificados.

Es importante validar que los subsistemas definidos en la tarea Identificación de *Subsistemas de diseño* tienen la mínima interfaz con otros subsistemas. Por este motivo, se elaboran los escenarios al nivel de subsistemas y, de esta forma, se delimitan las interfaces necesarias para cada uno de ellos, teniendo en cuenta toda la funcionalidad del sistema que recogen los casos de uso. Además, durante esta actividad pueden surgir requisitos de implementación, que se recogen en el catálogo de requisitos.

Las tareas de esta actividad se realizan en paralelo con las de *Diseño de clases*.

Esta actividad se descompone en las siguientes tareas:

5.2.2.1. Identificación de clases asociadas a un caso de uso

El objetivo de esta tarea es identificar las clases que intervienen en cada caso de uso, a partir del conjunto de clases definidas en la tarea *Identificación de clases adicionales*, ya que, como se ha señalado en la introducción de esta actividad, las actividades 5.2.2 y 0 se realizan en paralelo. Dichas clases se identifican a partir de las clases del modelo del análisis y de aquellas clases adicionales necesarias para el escenario que se está diseñando.

A su vez, a medida que se va estudiando la descripción de los casos de uso, pueden aparecer nuevas clases de diseño que no hayan sido identificadas anteriormente y que se incorporan al modelo de clases en la tarea *Identificación de clases adicionales*.

Entradas:

- Modelo de clases de diseño
- Modelo de casos de uso
- Especificación de casos de uso
- Análisis de la realización de los casos de uso

Salidas:

- Diseño de la realización de los casos de uso
    - Especificación detallada

Técnicas:

- Diagrama de interacción de objetos

Participantes:

- Analista(s) - diseñador(es)

5.2.2.2. Diseño de la realización de los casos de uso

El objetivo de esta tarea es definir cómo interactúan entre sí los objetos identificados en la tarea anterior para realizar, desde un punto de vista técnico, un caso de uso del sistema de información. Para ello, se parte de los escenarios especificados en el análisis, y se detallan teniendo en cuenta que se deben llevar cabo sobre un entorno tecnológico concreto y unos mecanismos genéricos de diseño.

Durante el desarrollo de esta tarea, es posible que surjan excepciones que se incluyen en el catálogo de excepciones, y que ahora quedan resueltas en los escenarios correspondientes.

Algunos de estos escenarios necesitan nueva interfaz de usuario. Por lo tanto, las clases de interfaz que se identifiquen se incorporan al modelo de clases de la tarea *Identificación de clases adicionales*, para realizar su diseño detallado.

También se realiza el estudio de los escenarios de los distintos casos de uso, para identificar comportamientos comunes sobre los que se puedan aplicar mecanismos genéricos de diseño, o se puede decidir diseñar un subsistema de soporte que contenga dicho comportamiento, como un servicio.

El estudio de los comportamientos comunes identificados puede servir de ayuda para detallar o revisar la herencia entre clases en la tarea *Diseño de la jerarquía*.

Entradas:

- Modelo de casos de uso
- Especificación de casos de uso
- Análisis de la realización de los casos de uso
- Especificación de interfaz de usuario
- Diseño de la realización de los casos de uso

Salidas:

- Diseño de la realización de los casos de uso
    - Especificación detallada

Técnicas:

- Diagrama de interacción de objetos

Participantes:

- Analista(s) - diseñador(es)

5.2.2.3. Revisión de la interfaz de usuario

El objetivo de esta tarea es realizar el diseño detallado del comportamiento de la interfaz de usuario a partir de la especificación de la misma, obtenida en el proceso de análisis, y de acuerdo con el entorno tecnológico definido. Si se hubiera realizado un prototipo de la interfaz de usuario, éste se tomaría como punto de partida para el diseño. Además, se incluyen las ventanas alternativas o elementos de diseño surgidos como consecuencia del diseño de los escenarios definidos en la tarea anterior.

Además, se revisa: la interfaz de usuario, la navegación entre ventanas, los elementos que forman cada interfaz, sus características (que deben ser consistentes con los atributos con los que están relacionadas), su disposición, y cómo se gestionan los eventos relacionados con los objetos.

En aquellos casos en los que se realizan modificaciones significativas sobre la interfaz de usuario, es conveniente que éste las valide, siendo opcional la realización de un nuevo prototipo.

Entradas:

- Especificación de interfaz de usuario
- Diseño de la realización de los casos de uso

Salidas:

- Diseño de interfaz de usuario
    - Formatos individuales de interfaz de pantalla gráfica
    - Catálogo de controles y elementos de diseño de interfaz de pantalla gráfica
    - Modelo de navegación de interfaz de pantalla gráfica
    - Formatos de impresión
    - Prototipo de interfaz de pantalla gráfica

Técnicas:

- Diagrama de interacción de objetos
- Diagrama de transición de estados
- Prototipado

Participantes:

- Analista(s) - diseñador(es)
- Usuarios / clientes

5.2.2.4. Revisión de subsistemas de diseño e interfaces

El objetivo de esta tarea es describir cada caso de uso en términos de los subsistemas que participan en el caso de uso y las interfaces que se requieren entre ellos.

Para un caso de uso hay que definir, además de los subsistemas y actores que intervienen en el mismo, los mensajes que intercambian los objetos de un subsistema con otro.

Estos mensajes sirven para verificar y detallar las interfaces de cada subsistema, teniendo en cuenta todos los casos de uso en los que interviene, y completar de esta manera la definición de subsistemas establecida en la tarea *Identificación de subsistemas de diseño*.

Entradas:

- Modelo de casos de uso
- Especificación de casos de uso
- Diseño de la realización de los casos de uso

Salidas:

- Diseño de la realización de los casos de uso
    - Definición a nivel de subsistemas e interfaz

Técnicas:

- Diagrama de interacción de objetos

Participantes:

- Analista(s) - diseñador(es)

5.2.3. Diseño de clases

El propósito de esta actividad es transformar el modelo de clases lógico, que proviene del análisis, en un modelo de clases de diseño. Dicho modelo recoge la especificación detallada de cada una de las clases, es decir, sus atributos, operaciones, métodos, y el diseño preciso de las relaciones establecidas entre ellas, bien sean de agregación, asociación o jerarquía. Para llevar a cabo todos estos puntos, se tienen en cuenta las decisiones tomadas sobre el entorno tecnológico y el entorno de desarrollo elegido para la implementación.

Se identifican las clases de diseño, que denominamos clases adicionales, en función del estudio de los escenarios de los casos de uso, que se está realizando en paralelo en la actividad *Diseño de casos de uso*, y aplicando los mecanismos genéricos de diseño que se consideren convenientes por el tipo de especificaciones tecnológicas y de desarrollo. Entre ellas se encuentran clases abstractas, que integran características comunes con el objetivo de especializarlas en clases derivadas. Se diseñan las clases de interfaz de usuario, que provienen del análisis. Como consecuencia del estudio de los escenarios secundarios que se está realizando, pueden aparecer nuevas clases de interfaz.

También hay que considerar que, por el diseño de las asociaciones y agregaciones, pueden aparecer nuevas clases, o desaparecer incluyendo sus atributos y métodos en otras, si se considera conveniente por temas de optimización.

La jerarquía entre las clases se va estableciendo a lo largo de esta actividad, a medida que se van identificando comportamientos comunes en las clases, aunque haya una tarea propia de diseño de la jerarquía.

Otro de los objetivos del diseño de las clases es identificar para cada clase, los atributos, las operaciones que cubren las responsabilidades que se identificaron en el análisis, y la especificación de los métodos que implementan esas operaciones, analizando los escenarios del *Diseño de casos de uso*. Se determina la visibilidad de los atributos y operaciones de cada clase, con respecto a las otras clases del modelo.

Una vez que se ha elaborado el modelo de clases, se define la estructura física de los datos correspondiente a ese modelo, en la actividad *Diseño físico de datos*.

Además, en los casos en que sea necesaria una migración de datos de otros sistemas o una carga inicial de información, se realizará su especificación a partir del modelo de clases y las estructuras de datos de los sistemas origen.

Como resultado de todo lo anterior se actualiza el modelo de clases del análisis, una vez recogidas las decisiones de diseño.

Esta actividad se descompone en las siguientes tareas:

5.2.3.1. Identificación de clases adicionales

El objetivo de esta tarea es identificar un conjunto de clases que completen el modelo de clases analizado en la tarea *Validación de los modelos* del proceso anterior (clases y/o interfaces) teniendo en cuenta que:

- Cada interfaz identificada en el análisis se corresponde en el diseño con una clase que proporcione esa interfaz.
- El conjunto de clases del análisis puede modificarse en función de las tecnologías de desarrollo utilizadas y de los mecanismos genéricos de diseño especificados.

Las clases de control deben contemplar la coordinación y secuencia entre objetos y, en algunos casos, deben contener lógica de negocio. De cualquier manera, se deben considerar cuestiones de distribución, de rendimiento, de transacción y de serialización.

El diseño de las clases de entidad varía según el sistema de gestión de datos utilizado.

Las clases pueden ser construidas por el propio desarrollador, adquiridas en forma de bibliotecas, facilitadas por el entorno de trabajo o por el entorno tecnológico.

El diseño de las clases de interfaz de usuario depende de la tecnología específica que se esté utilizando. Así, por ejemplo, la interfaz puede crearse a partir de los objetos gráficos disponibles en el entorno de desarrollo, sin necesidad de que estos se contemplen en el modelo de clases correspondiente.

Entre las clases identificadas a lo largo de esta tarea se encuentran clases abstractas, que reúnen características comunes a varias clases. Cada subclase aumenta su estructura y comportamiento con la clase abstracta de la que hereda.

Entradas:

- Modelo de clases de análisis
- Especificación de interfaz de usuario

Salidas:

- Modelo de clases de diseño

Técnicas:

- Diagrama de clases

Participantes:

- Analista(s) - diseñador(es)

5.2.3.2. Diseño de asociaciones y agregaciones

En esta tarea se completan las asociaciones entre las clases del modelo de clases del diseño, estudiando la secuencia de mensajes entre los objetos correspondientes en el diagrama de interacción de los escenarios definidos en la tarea *Descripción de la interacción entre objetos*.

Para definir las asociaciones, partimos de las que fueron identificadas en la tarea *Identificación de asociaciones y agregaciones*, teniendo en cuenta que:

- Las características de la asociación (papeles que desempeña, multiplicidad, etc.) se detallan según el entorno de desarrollo utilizado
- Las relaciones bidireccionales se transforman en unidireccionales, para simplificar la implementación del sistema
- Se realiza la modelización de las rutas de acceso óptimas entre las asociaciones para evitar problemas de rendimiento
- Se analiza la posibilidad de diseñar como clases algunas de las asociaciones

Opcionalmente, se especifica la forma en la que se va a implementar cada asociación (punteros, colecciones, etc.).

Entradas:

- Modelo de clases de análisis
- Modelo de clases de diseño

Salidas:

- Modelo de clases de diseño

Técnicas:

- Diagrama de clases

Participantes:

- Analista(s) - diseñador(es)

5.2.3.3. Identificación de atributos de las clases

El objetivo de esta tarea es identificar y describir, una vez que se ha especificado el entorno de desarrollo, los atributos de las clases.

Para identificar los atributos se revisa el modelo de clases obtenido en el proceso de *Análisis del sistema de información*, considerando que, a partir de uno de ellos, puede ser necesario definir atributos adicionales. Para cada atributo identificado se define su tipo, con formatos específicos, y si existieran, las restricciones asociadas a ese atributo.

Asimismo, se analiza la posibilidad de convertir un atributo en clase en aquellos casos en los que:

- El atributo se defina en varias clases de diseño
- La complejidad del atributo aumente la dificultad para comprender la clase a la que pertenece

Entradas:

- Modelo de clases de análisis
- Modelo de clases de diseño

Salidas:

- Modelo de clases de diseño

Técnicas:

- Diagrama de clases

Participantes:

- Analista(s) - diseñador(es)

5.2.3.4. Identificación de operaciones de las clases

El objetivo de esta tarea es definir, de forma detallada, las operaciones de cada clase de diseño. Para ello, se toma como punto de partida el modelo de clases generado en el análisis, así como el diseño de los casos de uso reales y los requisitos de diseño que pueden aparecer al definir el entorno de desarrollo.

Las operaciones de las clases de diseño surgen para dar respuesta a las responsabilidades de las clases de análisis y, además, para definir las interfaces que ofrece esa clase.

Según el entorno de desarrollo utilizado, se describe cada operación especificando: su nombre, parámetros y visibilidad (pública, privada, protegida). Si el entorno de desarrollo lo permite, se tiene en cuenta la posibilidad de simplificar el modelo de clases haciendo uso del polimorfismo y la sobrecarga de operaciones.

Para identificar las operaciones de aquellos objetos que presenten distintos estados, por lo que su comportamiento depende del estado en el que se encuentren, es recomendable realizar un diagrama de transición de estados, y traducir cada acción o actividad del mismo en una de estas operaciones.

Entradas:

- Modelo de clases de análisis
- Comportamiento de clases de análisis
- Modelo de clases de diseño

Salidas:

- Modelo de clases de diseño

Técnicas:

- Diagrama de clases
- Diagrama de transición de estados

Participantes:

- Analista(s) - diseñador(es)

5.2.3.5. Diseño de la jerarquía

El objetivo de esta tarea es revisar la jerarquía de clases que ha surgido en el modelo de clases a lo largo de las tareas anteriores y comprobar que esa jerarquía es viable según los mecanismos disponibles en el entorno de desarrollo utilizado.

Entre las modificaciones realizadas sobre la jerarquía se identifican clases abstractas, que son superclases en las que se agrupan atributos y operaciones que heredan sus subclases.

Entradas:

- Modelo de clases de diseño

Salidas:

- Modelo de clases de diseño

Técnicas:

- Diagrama de clases

Participantes:

- Analista(s) - diseñador(es)

5.2.3.6. Descripción de métodos de las operaciones

En esta tarea se describen los métodos que se usan para detallar como se realiza cada una de las operaciones de una clase. Los métodos pueden especificarse mediante un algoritmo, usando pseudocódigo o lenguaje natural. Su implementación se basa en la secuencia de interacciones del diagrama de interacción en los que la clase aparezca o en la secuencia de transiciones del diagrama de transición de estados.

En la mayoría de los casos, esta tarea no se realiza hasta el proceso de construcción, en el que los métodos se describen directamente en el lenguaje de programación que se va a utilizar.

Entradas:

- Modelo de clases de diseño
- Comportamiento de clases de diseño

Salidas:

- Modelo de clases de diseño

Técnicas:

- Diagrama de clases

Participantes:

- Analista(s) - diseñador(es)

5.2.3.7. Especificación de necesidades de migración y carga inicial de datos

En esta tarea se realiza, en los casos que sea necesario, una primera especificación de las necesidades de migración o carga inicial de los datos requeridos por el sistema, que se completa en la actividad *Diseño de la migración y carga inicial de datos*.

Entradas:

- Estructura de datos del sistema origen
- Modelo de clases de diseño

Salidas:

- Plan de migración y carga inicial de datos

Técnicas:

- Reuniones
- Análisis de estructuras de datos

Participantes:

- Analista(s) - diseñador(es)
- Usuarios / clientes

5.2.4. Diseño físico de datos

En esta actividad se define la estructura física de datos que utilizará el sistema, a partir del modelo lógico de datos normalizado o modelo de clases, de manera que teniendo presentes las características específicas del sistema de gestión de datos concreto a utilizar, los requisitos establecidos para el sistema de información, y las particularidades del entorno tecnológico, se consiga una mayor eficiencia en el tratamiento de los datos.

También se analizan los caminos de acceso a los datos utilizados por cada módulo/clase del sistema en consultas y actualizaciones, con el fin de mejorar los tiempos de respuesta y optimizar los recursos de máquina.

Las tareas de esta actividad se realizan de forma iterativa y en paralelo con las realizadas en las actividades *Definición de la arquitectura del sistema*, dónde se especifican los detalles de arquitectura e infraestructura y la planificación de capacidades, y *Diseño de casos de uso* y *de clases*, dónde se especifica la lógica de tratamiento y las interfaces utilizadas.

La obtención del modelo físico de datos se realiza aplicando una serie de reglas de transformación a cada elemento del modelo de clases que se está generando en la actividad *Diseño de clases*.

Esta actividad se descompone en las siguientes tareas:

5.2.4.1. Diseño del modelo físico de datos

El objetivo de esta tarea es realizar el diseño del modelo físico de datos a partir del modelo de clases.

Como paso previo al diseño de la estructura física de datos, se analizan las peculiaridades técnicas del gestor de bases de datos o sistema de ficheros a utilizar, y las estimaciones sobre la utilización y volumen de las ocurrencias de cada clase del modelo de clases. Además, si se ha establecido la necesidad de llevar a cabo una migración de datos, se deben tener en cuenta también los volúmenes de las estructuras de datos implicadas en la conversión. Esta información sirve para decidir la mejor implementación del modelo lógico de datos/modelo de clases, así como para hacer una estimación del espacio de almacenamiento.

De acuerdo al análisis anterior, se determina cómo se van a convertir las clases en tablas, considerando las relaciones existentes entre ellas y los identificadores, definiendo sus claves primarias, ajenas, alternativas u otros medios de acceso en general.

También se definen aquellos elementos que, en función del gestor o sistemas de ficheros a utilizar, se considere necesario implementar. Entre estos elementos podemos citar los siguientes:

- Bloqueo y compresión de datos
- Agrupamientos (*clustering*)
- Punteros

Entradas:

- Plan de migración y carga inicial de datos
- Modelo de clases de diseño

Salidas:

- Modelo físico de datos

Técnicas:

- Reglas de Obtención del Modelo Físico a partir del Lógico
- Reglas de Transformación
- Modelado de datos
- Mapeo objeto – relacional (ORM)

Participantes:

- Analista(s) - diseñador(es)
- Experto en base de datos

5.2.4.2. Especificación de los caminos de acceso a los datos

El objetivo de esta tarea es determinar los caminos de acceso a los datos persistentes del sistema, utilizados por los principales módulos/clases de acuerdo al modelo físico de datos, con el fin de optimizar el rendimiento de los gestores de datos o sistemas de ficheros y el consumo de recursos, así como disminuir los tiempos de respuesta.

Se recomienda realizar esta tarea para aquellas clases que reúnan, entre otras, alguna de las siguientes características:

- Tratamiento crítico
- Concurrencia
- Accesos complejos a datos

Para el inicio de esta tarea, se toma como referencia el *Diseño de clases* de los subsistemas específicos, producto que se está generando en paralelo a esta actividad.

Para cada clase se identifican las tablas o ficheros y el tipo de acceso realizado, así como el orden que debe seguirse para la obtención de los datos. Asimismo, se efectúa una estimación del número de accesos que deben realizarse teniendo en cuenta, a su vez, la frecuencia y la prioridad del acceso.

La información obtenida sirve para identificar accesos excesivamente costosos o redundantes que pueden comprometer el rendimiento final del sistema y que, por lo tanto, exigen la optimización del modelo físico de datos, mediante la creación de nuevos accesos, posibles desnormalizaciones o particiones del modelo físico de datos.

Entradas:

- Modelo físico de datos
- Modelo de clases de diseño

Salidas:

- Especificación de los caminos de acceso a los datos

Técnicas:

- Cálculo de Accesos Físicos
- Caminos de Acceso
- Modelado de datos

Participantes:

- Analista(s) - diseñador(es)
- Experto en base de datos

5.2.4.3. Optimización del modelo físico de datos

En esta tarea se optimiza el diseño físico de datos, con el objetivo de mejorar el tiempo de respuesta en el acceso a datos persistentes, hacer una adecuada utilización de los recursos del sistema y, en consecuencia, garantizar que el diseño satisface las necesidades de tratamiento establecidas para el sistema de información en cuanto a que se ajusta a los requisitos de rendimiento exigidos.

A partir de la especificación de la secuencia de accesos de aquellas clases identificadas como críticas, obtenida en la tarea anterior, se detectan las posibles mejoras con el fin de conseguir los niveles de rendimiento establecidos y, por lo tanto, una mayor eficiencia del sistema. Como resultado, puede ser necesaria una desnormalización controlada que se aplica para reducir o simplificar el número de accesos a los sistemas de almacenamiento de datos.

La desnormalización puede obligar a:

- Introducir elementos redundantes (campos, campos derivados, etc.)
- Definir nuevos caminos de acceso
- Redefinir relaciones
- Dividir o unir tablas

En la revisión de la estructura física de datos se deben tener en cuenta criterios relacionados con:

- Clases identificadas como críticas
- Estimación de volúmenes
- Frecuencia y tipo de acceso
- Estimaciones de crecimiento por periodo

- Requisitos relativos al rendimiento, seguridad, confidencialidad y disponibilidad, entre otros, considerados relevantes.

Es importante que la desnormalización se lleve a cabo de una forma controlada, para evitar anomalías en el tratamiento de los datos.

Entradas:

- Catálogo de requisitos
- Modelo físico de datos
- Especificación de los caminos de acceso a los datos

Salidas:

- Modelo físico de datos optimizado

Técnicas:

- Optimización
- Modelado de datos

Participantes:

- Analista(s) - diseñador(es)
- Experto en base de datos
- Equipo de seguridad de IT CGAE

5.2.4.4. Especificación de la distribución de datos

En esta tarea se determina el modelo de distribución de datos, teniendo en cuenta los requisitos de diseño establecidos. Se establece la ubicación de los gestores de bases de datos o sistemas de ficheros, así como de los distintos elementos de la estructura física de datos, en los nodos correspondientes, de acuerdo al particionamiento físico del sistema de información especificado en la actividad Diseño de la *Arquitectura del sistema*.

El resultado de esta actividad es la especificación de los modelos físicos particulares de cada nodo, esquemas físicos de datos, así como su asignación a los nodos.

Entradas:

- Catálogo de requisitos
- Modelo físico de datos optimizado
- Diseño de la Arquitectura del Sistema

Salidas:

- Esquemas físicos de datos
- Asignación esquemas físicos de datos a nodos

Técnicas:

- Matrices de asignación

Participantes:

- Analista(s) - diseñador(es)
- Experto en base de datos
- Equipo de arquitectura de IT CGAE (área de Explotación)

5.2.5. Generación de especificaciones de construcción

En esta actividad se generan las especificaciones para la construcción del sistema de información, a partir del diseño detallado.

Estas especificaciones definen la construcción del sistema de información a partir de las unidades básicas de construcción (en adelante, componentes), entendiendo como tales unidades independientes y coherentes de construcción y ejecución, que se corresponden con un empaquetamiento físico de los elementos del diseño de detalle, como pueden ser módulos, clases o especificaciones de interfaz.

La división del sistema de información en subsistemas de diseño proporciona, por continuidad, una primera división en subsistemas de construcción, definiendo para cada uno de ellos los componentes que lo integran. Si se considera necesario, un subsistema de diseño se podrá dividir a su vez en sucesivos niveles para mayor claridad de las especificaciones de construcción.

Las dependencias entre subsistemas de diseño proporcionan información para establecer las dependencias entre los subsistemas de construcción y, por lo tanto, definir el orden o secuencia que se debe seguir en la construcción y en la realización de las pruebas.

También se generan las especificaciones necesarias para la creación de las estructuras de datos en los gestores de bases de datos o sistemas de ficheros.

El producto resultante de esta actividad es el conjunto de las especificaciones de construcción del sistema de información, que comprende:

- Especificación del entorno de construcción
- Descripción de subsistemas de construcción y dependencias
- Descripción de componentes
- Plan de integración del sistema de información
- Especificación detallada de componentes
- Especificación de la estructura física de datos

Esta actividad se descompone en las siguientes tareas:

5.2.5.1. Especificación del entorno de construcción

El objetivo de esta tarea es la definición detallada y completa del entorno necesario para la construcción de los componentes del sistema de información.

Se propone que la especificación del entorno se realice según los siguientes conceptos:

- Entorno tecnológico: hardware, software y comunicaciones
- Herramientas de construcción, generadores de código, compiladores, etc.
- Restricciones técnicas del entorno
- Planificación de capacidades previstas, o la información que estime oportuno el departamento de sistemas para efectuar dicha planificación
- Requisitos de operación y seguridad del entorno de construcción

Entradas:

- Catálogo de requisitos
- Diseño de la arquitectura del sistema
- Entorno tecnológico del sistema

Salidas:

- Especificaciones de construcción del sistema de información
    - Especificación del entorno de construcción

Técnicas:

- Descripción textual

Participantes:

- Analista(s) - diseñador(es)
- Equipo de arquitectura de IT CGAE (área de Explotación)
- Equipo de Seguridad de IT CGAE

5.2.5.2. Definición de componentes y subsistemas de construcción

La especificación de los subsistemas de construcción se realiza a partir de los subsistemas de diseño, con una continuidad directa, permitiéndose a su vez un mayor nivel de detalle agrupando componentes en subsistemas dentro de un subsistema de construcción.

Los componentes se definen mediante la agrupación de elementos del diseño de detalle de cada subsistema de diseño. En principio, cada módulo o clase y cada formato individual de interfaz se corresponden con un componente, aunque se pueden agrupar o redistribuir módulos o clases en componentes, siguiendo otros criterios más oportunos, como pueden ser:

- Optimización de recursos
- Características comunes de funcionalidad o de acceso a datos
- Necesidades especiales de ejecución: elementos críticos, accesos costosos a datos, etc.

Los subsistemas de construcción y las dependencias entre subsistemas y entre componentes de un subsistema recogen aspectos prácticos relativos a la plataforma concreta de construcción y ejecución. Entre estos aspectos se pueden citar, por ejemplo:

- Secuencia de compilación entre componentes
- Agrupación de elementos en librerías o *packages* (por ejemplo, DLL en el entorno Windows, *packages* en Java)

La asignación de subsistemas de construcción a nodos, por continuidad con el diseño, determina la distribución de los componentes que lo integran.

Opcionalmente, se propone la realización de un plan de integración del sistema de información, especificando la secuencia y organización de la construcción y prueba de los subsistemas de construcción y de los componentes, desde un punto de vista técnico.

Entradas:

- Especificaciones de construcción del sistema de información
- Catálogo de requisitos
- Catálogo de normas de IT CGAE
- Diseño de la arquitectura del sistema
- Asignación de esquemas físicos de datos a nodos
- Diseño de interfaz de usuario
- Diseño de la realización de los casos de uso
- Modelo de clases de diseño
- Comportamiento de clases de diseño

Salidas:

- Especificaciones de construcción del sistema de información
  - Descripción de subsistemas de construcción y dependencias
  - Descripción de componentes
  - Plan de integración del sistema de información

Técnicas:

- Diagrama de estructura
- Diagrama de componentes
- Diagrama de despliegue

Participantes:

- Analista(s) - diseñador(es)
- Equipo de arquitectura de IT CGAE (área de Explotación)

5.2.5.3. Elaboración de especificaciones de construcción

Se realiza una especificación detallada de cada componente, en pseudocódigo o lenguaje natural, completando la información que se considere necesaria según el entorno tecnológico.

Asimismo, se determinan y especifican todos los elementos o parámetros complementarios a la propia definición de componentes que, en función del entorno tecnológico, completan las especificaciones de construcción.

Entradas:

- Especificaciones de construcción del sistema de información
- Catálogo de requisitos
- Catálogo de excepciones
- Catálogo de normas de IT CGAE
- Diseño de la arquitectura del sistema
- Entorno tecnológico del sistema
- Modelo físico de datos optimizado
- Esquemas físicos de datos
- Asignación de esquemas físicos de datos a nodos
- Diseño de interfaz de usuario
- Diseño de la realización de los casos de uso
- Modelo de clases de diseño
- Comportamiento de clases de diseño

Salidas:

- Especificaciones de construcción del sistema de información
  - Especificación detallada de componentes

Técnicas:

- Diagrama de componentes

Participantes:

- Analista(s) - diseñador(es)

5.2.5.4. Elaboración de especificaciones del modelo físico de datos

En esta tarea se generan las especificaciones necesarias para la definición y creación de los elementos del modelo físico de datos, mediante el lenguaje de definición de datos del correspondiente gestor de base de datos o sistema de ficheros, teniendo en cuenta el entorno tecnológico, las normas y estándares de la organización y características intrínsecas del gestor o sistema de ficheros a utilizar.

Entradas:

- Especificaciones de construcción del sistema de información
- Catálogo de requisitos
- Catálogo de normas de IT CGAE
- Entorno tecnológico del sistema
- Modelo físico de datos optimizado
- Esquemas físicos de datos
- Asignación de esquemas físicos de datos a nodos

Salidas:

- Especificaciones de construcción del sistema de información
    - Especificación de la estructura física de datos

Técnicas:

- Lenguaje SQL

Participantes:

- Analista(s) - diseñador(es)
- Experto en base de datos

5.2.6. Diseño de la migración y carga inicial de datos

Esta actividad sólo se lleva a cabo cuando es necesaria una carga inicial de información, o una migración de datos de otros sistemas, cuyo alcance y estrategia a seguir se habrá establecido previamente.

Para ello, se toma como referencia el plan de migración y carga inicial de datos, que recoge las estructuras físicas de datos del sistema o sistemas origen implicadas en la conversión, la prioridad en las cargas y secuencia a seguir, las necesidades previas de depuración de la información, así como los requisitos necesarios para garantizar la correcta implementación de los procedimientos de migración sin comprometer el funcionamiento de los sistemas actuales.

A partir de dicho plan, y de acuerdo a la estructura física de los datos del nuevo sistema, obtenida en la actividad *Diseño físico de datos*, y a las características de la arquitectura y del entorno tecnológico propuesto en la actividad *Definición de la arquitectura del sistema*, se procede a definir y diseñar en detalle los procedimientos y procesos necesarios para realizar la migración.

Se completa el plan de pruebas específico establecido en el plan de migración y carga inicial, detallando las pruebas a realizar, los criterios de aceptación o rechazo de la prueba y los responsables de la organización, realización y evaluación de resultados.

Asimismo, se determinan las necesidades adicionales de infraestructura, tanto para la implementación de los procesos como para la realización de las pruebas.

Esta actividad se descompone en las siguientes tareas:

5.2.6.1. Especificación del entorno de migración

El objetivo de esta tarea es definir el entorno tecnológico propio de los procesos de migración y carga inicial, adecuando al mismo las necesidades y requisitos reflejados en el plan de migración y carga inicial de datos. En la descripción del entorno tecnológico, hay que tener en cuenta las herramientas o utilidades software específicas de estos procesos.

Se realiza una estimación de capacidades (*capacity planning*) para este entorno que permita evaluar las necesidades de infraestructura, principalmente relacionadas con el espacio de almacenamiento y las comunicaciones.

Entradas:

- Plan de migración y carga inicial de datos
- Diseño de la arquitectura del sistema
- Entorno tecnológico del sistema
- Modelo físico de datos optimizado
- Esquemas físicos de datos
- Asignación de esquemas físicos de datos a nodos

Salidas:

- Plan de migración y carga inicial de datos
    - Especificación del entorno de migración y carga inicial

Técnicas:

- Descripción textual

Participantes:

- Analista(s) - diseñador(es)
- Equipo de arquitectura de IT CGAE (área de Explotación)
- Experto en base de datos

5.2.6.2. Diseño de procedimientos de migración y carga inicial

El objetivo de esta tarea es la definición de los procedimientos necesarios para llevar a cabo la migración y carga inicial de datos del sistema.

Como punto de partida se tiene en cuenta, junto con los requisitos y especificaciones de migración y carga inicial, el modelo físico de datos optimizado y su localización en los nodos, así como la definición del entorno tecnológico del sistema de información.

Los procedimientos asociados a la migración y carga inicial de datos son, principalmente, los relacionados con la preparación, la realización y la posterior verificación del proceso. Entre ellos se encuentran los siguientes:

- Procedimientos de seguridad, relativos a:
    - Control de acceso a la información
    - Copias de seguridad de los procesos
    - Recuperación de la información
    - Tratamiento de las posibles contingencias durante la conversión
- Procedimientos de carga de datos, relativos a:
    - Depuraciones previas de información
    - Procesos de validación

  - Procesos de importación
  - Procesos de carga y prioridades
- Procedimientos de verificación de los procesos y comprobación de la integridad de la información resultante al finalizar la conversión, conforme a la estructura física de los datos destino

Entradas:

- Plan de migración y carga inicial de datos
- Diseño de la arquitectura del sistema
- Entorno tecnológico del sistema
- Modelo físico de datos optimizado
- Esquemas físicos de datos
- Asignación de esquemas físicos de datos a nodos

Salidas:

- Plan de migración y carga inicial de datos
  - Definición de procedimientos de migración y carga inicial

Técnicas:

- Descripción textual

Participantes:

- Analista(s) - diseñador(es)
- Equipo de arquitectura de IT CGAE (área de Explotación)
- Equipo de Seguridad de IT CGAE
- Experto en base de datos

5.2.6.3. Diseño detallado de componentes de migración y carga inicial

El objetivo de esta tarea es el diseño detallado, en sucesivos niveles de detalle, de los módulos de migración y carga inicial, indicando la jerarquía y orden de ejecución.

El diseño de los módulos necesarios para la migración y carga inicial no es conceptualmente distinto del diseño de cualquier otro módulo del sistema de información, por lo que se recomienda utilizar pautas similares. Se debe tener en cuenta el modelo físico de datos del sistema de información, así como las estructuras de datos del sistema o sistemas origen recogidas en el plan de migración y carga inicial de datos.

Finalmente, se complementa el plan de migración y carga inicial con la definición de los distintos tipos de prueba a realizar.

Entradas:

- Plan de migración y carga inicial de datos
- Diseño de la arquitectura del sistema
- Entorno tecnológico del sistema
- Modelo físico de datos optimizado
- Esquemas físicos de datos
- Asignación de esquemas físicos de datos a nodos

Salidas:

- Plan de migración y carga inicial de datos
  - Diseño detallado de módulos de migración y carga inicial
  - Especificación técnica de las pruebas de migración y carga inicial

Técnicas:

- Descripción textual

Participantes:

- Analista(s) - diseñador(es)

5.2.6.4. Revisión de la planificación de la migración

El objetivo de esta tarea es completar la especificación del plan de migración y carga inicial, concretando el plan de trabajo de acuerdo a los procedimientos y procesos de migración y carga inicial definidos.

Entradas:

- Plan de migración y carga inicial de datos

Salidas:

- Plan de migración y carga inicial de datos
  - Planificación de la migración y carga inicial

Técnicas:

- Planificación de tareas
- Diagrama de Gantt

Participantes:

- Jefe de proyecto
- Usuarios / clientes

5.2.7. Especificación técnica del plan de pruebas

En esta actividad se realiza la especificación de detalle del plan de pruebas del sistema de información para cada uno de los niveles de prueba establecidos en el proceso *Análisis del Sistema de Información*:

- Pruebas unitarias
- Pruebas de integración
- Pruebas del sistema
- Pruebas de implantación
- Pruebas de aceptación

Para ello se toma como referencia el plan de pruebas, que recoge los objetivos de la prueba de un sistema, establece y coordina una estrategia de trabajo, y provee del marco adecuado para planificar paso a paso las actividades de prueba. También puede ser una referencia el plan de integración del sistema de información propuesto, opcionalmente, en la tarea *Definición de componentes y subsistemas de construcción*.

El catálogo de requisitos, el catálogo de excepciones y el diseño detallado del sistema de información, permiten la definición de las verificaciones que deben realizarse en cada nivel de prueba para comprobar que el sistema responde a los requisitos planteados. La asociación de las distintas verificaciones a componentes, grupos de componentes y subsistemas, o al sistema de información completo, determina las distintas verificaciones de cada nivel de prueba establecido.

Las pruebas unitarias comprenden las verificaciones asociadas a cada componente del sistema de información. Su realización tiene como objetivo verificar la funcionalidad y estructura de cada componente individual.

Las pruebas de integración comprenden verificaciones asociadas a grupos de componentes, generalmente reflejados en la definición de subsistemas de construcción o en el plan de integración del sistema de información. Tienen por objetivo verificar el correcto ensamblaje entre los distintos componentes.

Las pruebas del sistema, de implantación y de aceptación corresponden a verificaciones asociadas al sistema de información, y reflejan distintos propósitos en cada tipo de prueba:

- Las pruebas del sistema son pruebas de integración del sistema de información completo. Permiten probar el sistema en su conjunto y con otros sistemas con los que se relaciona para verificar que las especificaciones funcionales y técnicas se cumplen
- Las pruebas de implantación incluyen las verificaciones necesarias para asegurar que el sistema funcionará correctamente en el entorno de operación al responder satisfactoriamente a los requisitos de rendimiento, seguridad y operación, y coexistencia con el resto de los sistemas de la instalación, y conseguir la aceptación del sistema por parte del usuario de operación
- Las pruebas de aceptación van dirigidas a validar que el sistema cumple los requisitos de funcionamiento esperado, recogidos en el catálogo de requisitos y en los criterios de aceptación del sistema de información, y conseguir la aceptación final del sistema por parte del usuario

Las pruebas unitarias, de integración y del sistema se llevan a cabo en el proceso Construcción del Sistema de Información (CSI), mientras que las pruebas de implantación y aceptación se realizan en el proceso Implantación y Aceptación del Sistema (IAS).

Esta actividad se descompone en las siguientes tareas:

5.2.7.1. Especificación del entorno de pruebas

El objetivo de esta tarea es la definición detallada y completa del entorno necesario para la realización de las pruebas del sistema: unitarias, de integración, de implantación y de aceptación.

Se propone considerar los siguientes conceptos en la especificación del entorno:

- Entorno tecnológico: hardware, software y comunicaciones
- Restricciones técnicas del entorno
- Requisitos de operación y seguridad del entorno de pruebas
- Herramientas de prueba relacionadas con la extracción de juegos de ensayo, análisis de resultados, utilidades de gestión del entorno, etc.
- Planificación de capacidades previstas, o la información que estime oportuno el departamento técnico para efectuar dicha planificación
- Procedimientos de promoción de elementos entre entornos (desarrollo, pruebas, explotación, etc.)
- Procedimientos de emergencia y de recuperación, así como de vuelta atrás

Entradas:

- Plan de pruebas
- Catálogo de requisitos
- Catálogo de normas de IT CGAE
- Catálogo de excepciones

- Diseño de la arquitectura del sistema
- Entorno tecnológico del sistema
- Modelo físico de datos optimizado
- Esquemas físicos de datos
- Asignación de esquemas físicos de datos a nodos
- Especificaciones de construcción del sistema de información

Salidas:

- Plan de pruebas
    - Especificación del entorno de pruebas

Técnicas:

- Descripción textual

Participantes:

- Jefe de proyecto
- Analista(s) - diseñador(es)
- Equipo de Soporte de IT CGAE
- Equipo de Seguridad de IT CGAE

5.2.7.2. Especificación técnica de niveles de prueba

El objetivo de esta tarea es el diseño detallado de los distintos niveles de prueba, especificados en el plan de pruebas elaborado en el proceso Análisis del Sistema de Información.

El plan de integración del sistema de información, si se ha definido en la actividad *Definición de componentes y subsistemas de construcción*, va a servir de referencia para la elaboración detallada del plan de pruebas, principalmente las pruebas de integración y del sistema. En cualquier caso se hay que especificar la estrategia de integración de dichas pruebas.

De acuerdo a la arquitectura del sistema propuesta y a las características intrínsecas del diseño del sistema de información, se definen en detalle las distintas verificaciones a realizar sobre el sistema, conforme a los niveles de prueba establecidos, teniendo en cuenta que una verificación puede ser aplicable a varios componentes o grupos de componentes.

Estas verificaciones deben cubrir aspectos funcionales y no funcionales, considerando las excepciones que puedan producirse, así como las soluciones de diseño adoptadas, tanto del propio diseño de detalle del sistema de información, como de la utilización de subsistemas de soporte propios de la instalación.

Las verificaciones a realizar se especifican detallando:

- Ámbito de aplicación (prueba unitaria, de integración, del sistema, de implantación o aceptación) y objetivo
- Casos de prueba asociados: se definen en detalle los casos de prueba y se detalla cómo proceder en la ejecución de dichos casos, describiendo todas las entradas necesarias para ejecutar la prueba, y las relaciones de secuencialidad existentes entre las entradas, así como todas aquellas salidas que se espera obtener una vez ejecutado el caso de prueba, y las características especiales requeridas, como por ejemplo, tiempo de respuesta
- Procedimientos de prueba: se determina el conjunto de pasos a seguir para asegurar que los casos de prueba se ejecutan adecuadamente, especificando:
    - Casos de prueba a los que se aplica el procedimiento
    - Recursos hardware y software necesarios para ejecutar el procedimiento

  - Requisitos especiales o acciones necesarias para iniciar la ejecución
  - Requisitos especiales o acciones necesarias a realizar durante la ejecución del procedimiento
- Entorno de prueba: herramientas adicionales, condicionantes especiales de ejecución, etc.
- Criterios de aceptación de la prueba
- Análisis y evaluación de resultados

Como resultado final, se obtiene la relación de verificaciones que permiten comprobar:

- El correcto funcionamiento de cada componente (pruebas unitarias), cada subsistema de construcción o conjunto de componentes (pruebas de integración)
- La integración del sistema de información en su totalidad (pruebas del sistema)
- El ajuste del sistema a las necesidades para las que fue creado, de acuerdo a las características del entorno en el que se va a implantar (pruebas de implantación)
- La respuesta satisfactoria del sistema a los requisitos especificados por el usuario (pruebas de aceptación)

Entradas:

- Plan de pruebas
- Catálogo de requisitos
- Catálogo de normas de IT CGAE
- Catálogo de excepciones
- Diseño de la arquitectura del sistema
- Entorno tecnológico del sistema
- Modelo físico de datos optimizado
- Esquemas físicos de datos
- Asignación de esquemas físicos de datos a nodos
- Diseño de interfaz de usuario
- Especificaciones de construcción del sistema de información
- Diseño de la realización de los casos de uso
- Modelo de clases de diseño
- Comportamiento de clases de diseño

Salidas:

- Plan de pruebas
  - Especificación técnica de niveles de prueba

Técnicas:

- Descripción textual

Participantes:

- Jefe de proyecto
- Analista(s) - diseñador(es)
- Equipo de Soporte de IT CGAE
- Usuarios / clientes

5.2.7.3. Revisión de la planificación de pruebas

En esta tarea se completa y especifica la planificación de las pruebas, determinando los distintos perfiles implicados en la preparación y ejecución de las pruebas y en la evaluación de los resultados, así como el tiempo estimado para la realización de cada uno de los niveles de prueba, de acuerdo a la estrategia de integración establecida.

Entradas:

- Plan de pruebas

Salidas:

- Plan de pruebas
  - Planificación de las pruebas

Técnicas:

- Planificación de tareas
- Diagrama de Gantt

Participantes:

- Jefe de proyecto
- Usuarios / clientes
- Equipo de Soporte de IT CGAE

5.2.8. Establecimiento de requisitos de implantación

En esta actividad se completa el catálogo de requisitos con aquéllos relacionados con la documentación que el usuario requiere para operar con el nuevo sistema, y los relativos a la propia implantación del sistema en el entorno de operación.

La incorporación de estos requisitos permite ir preparando, en los procesos de Construcción del Sistema de Información (CSI) e Implantación y Aceptación del Sistema (IAS), los medios y recursos necesarios para que los usuarios, tanto finales como de operación, sean capaces de utilizar el nueva sistema de forma satisfactoria.

Esta actividad se descompone en las siguientes tareas:

5.2.8.1. Especificación de requisitos de documentación de usuario

En esta tarea se recoge toda la información necesaria para la especificación de la documentación a entregar al usuario, que incluirá los manuales de usuario y, cuando proceda, los manuales de explotación.

Para ello, es necesario definir, entre otros, los siguientes aspectos:

- Tipo de documentos y estándares a seguir en la elaboración de los mismos
- Formato en el que se desarrollarán
- Estructura
- Soporte en el que se van a generar
- Distribución y mantenimiento de la documentación y copias a editar
- Control de versiones

Entradas:

- Catálogo de requisitos
- Diseño de la arquitectura del sistema
- Entorno tecnológico del sistema

Salidas:

- Catálogo de requisitos

Técnicas:

- Reuniones

Participantes:

- Jefe de proyecto
- Usuarios / clientes
- Equipo de Soporte de IT CGAE
- Área de Explotación de IT CGAE

5.2.8.2. Especificación de requisitos de implantación

En esta tarea se especifican de forma detallada los requisitos de implantación, generalmente relacionados con la formación, infraestructura e instalación, con el fin de preparar y organizar, con la antelación suficiente, todos los recursos necesarios para la implantación e instalación del sistema de información.

Teniendo en cuenta las particularidades del sistema de información, se determinan los conocimientos o aptitudes adicionales que requieren los usuarios finales para operar con el nuevo sistema, al margen de la funcionalidad soportada por el mismo. Como consecuencia, se pueden establecer requisitos de formación indispensables, como condición previa, para el desarrollo del plan de formación que se elaborará en el proceso Implantación y Aceptación del Sistema (IAS).

Los requisitos de infraestructura e instalación hacen referencia a las necesidades especiales de equipamiento software, hardware y comunicaciones exigidos por el nuevo sistema, así como a los tipos de elementos implicados en la instalación, que deben tenerse en cuenta al especificar la estrategia de implantación, en el proceso Implantación y Aceptación del Sistema (IAS).

Entradas:

- Catálogo de requisitos
- Diseño de la arquitectura del sistema
- Entorno tecnológico del sistema

Salidas:

- Catálogo de requisitos

Técnicas:

- Reuniones

Participantes:

- Jefe de proyecto
- Representantes (a nivel directivo) de los usuarios / clientes
- Equipo de Soporte de IT CGAE

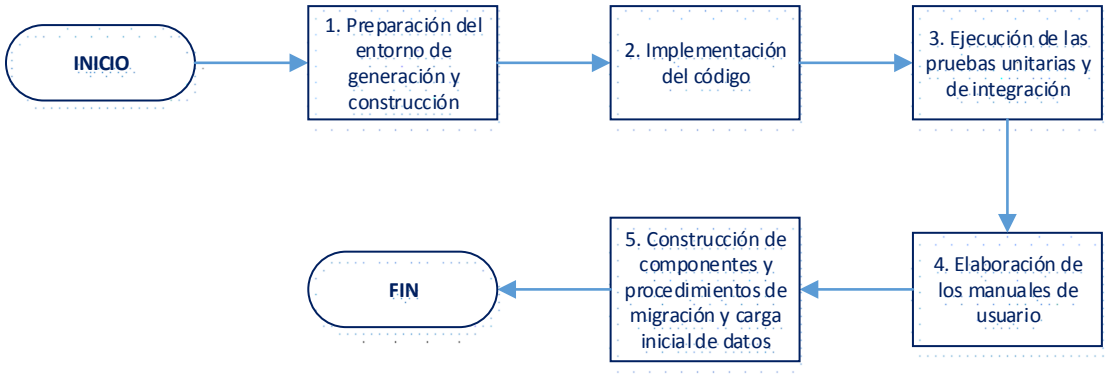
5.3. Construcción del Sistema de Información (CSI)

En esta fase se genera el código de los componentes del Sistema de Información, se desarrollan todos los procedimientos de operación y seguridad y se elaboran todos los manuales de usuario final y de explotación con el objetivo de asegurar el correcto funcionamiento del Sistema para su posterior implantación.

Para conseguir dicho objetivo, en este proceso se realizan las pruebas unitarias, las pruebas de integración de los subsistemas y componentes y las pruebas del sistema, de acuerdo al plan de pruebas establecido.

Asimismo, se define la formación de usuario final y, si procede, se construyen los procedimientos de migración y carga inicial de datos.

El siguiente diagrama representa el flujo de esta fase:



5.3.1. Preparación del entorno de generación y construcción

El objetivo de esta actividad es asegurar la disponibilidad de todos los medios y facilidades para que se pueda llevar a cabo la construcción del sistema de información. Entre estos medios, cabe destacar la preparación de los puestos de trabajo, equipos físicos y lógicos, gestores de bases de datos, bibliotecas de programas, herramientas de generación de código, bases de datos o ficheros de prueba, entre otros.

Las características del entorno de construcción y sus requisitos de operación y seguridad, así como las especificaciones de construcción de la estructura física de datos, se establecen en la actividad Generación de Especificaciones de Construcción (DSI 8), y constituyen el punto de partida para la realización de esta actividad.

Esta actividad se descompone en las siguientes tareas:

5.3.1.1. Implantación de la base de datos física o ficheros

En esta tarea hay que:

- Crear los elementos del sistema gestor de base de datos o sistema de ficheros
- Reservar el espacio de almacenamiento, definiendo, entre otros, los dispositivos físicos a emplear, tamaño de los bloques, tipo de registro físico, zona de desbordamiento, opciones de almacenamiento de datos, etc.
- Inicializar la base de datos o ficheros, cargando los datos considerados necesarios en el espacio de almacenamiento previamente definido

Entradas:

- Diseño de la arquitectura del sistema
- Entorno tecnológico del sistema
- Especificaciones de construcción del sistema de información

Salidas:

- Base de datos física o sistema de ficheros

Técnicas:

- Lenguaje SQL

Participantes:

- Analista(s) - diseñador(es)
- Desarrolladores
- Experto en base de datos

5.3.1.2. Preparación del entorno de construcción

En esta tarea se prepara el entorno en el que se construirán los componentes del sistema de información, contemplando aspectos tales como:

- Bibliotecas o librerías a utilizar
- Herramientas: generadores de código, editores, compiladores, verificadores sintácticos, montadores de enlace,
- Puestos de trabajo
- Implementación de los procedimientos de operación y seguridad propios del entorno de construcción, de acuerdo a los requisitos de seguridad y operación establecidos en la tarea *Especificación del entorno de construcción*

Entradas:

- Especificaciones de construcción del sistema de información

Salidas:

- Entorno de construcción

Técnicas:

- Control de versiones
- Gestión de dependencias

Participantes:

- Desarrolladores
- Técnicos de sistemas

5.3.2. Implementación del código

En esta tarea se genera el código correspondiente a cada uno de los componentes del sistema de información, identificados en la tarea *Definición de componentes y subsistemas de construcción*.

Para generar el código fuente se tienen en cuenta los estándares de nomenclatura, codificación y calidad utilizados por la organización y recogidos en el catálogo de normas de ITCGAE. En particular, se seguirán las buenas prácticas recogidas en el anexo C del presente documento.

Con el fin de verificar que el código fuente especifica de forma correcta el componente, se realiza su ensamblaje o compilación, verificando y corrigiendo los errores sintácticos, y el enlace del código objeto obtenido con las correspondientes bibliotecas.

Esta actividad se descompone en las siguientes tareas:

5.3.2.1. Implementación del código de componentes

En esta tarea se implementa el código correspondiente a cada uno de los componentes del sistema de información, identificados en la tarea *Definición de componentes y subsistemas de construcción*.

Para generar el código fuente se tienen en cuenta los estándares de nomenclatura, codificación y calidad utilizados por la organización y recogidos en el catálogo de normas de IT CGAE.

Con el fin de verificar que el código fuente especifica de forma correcta el componente, se realiza su ensamblaje o compilación, verificando y corrigiendo los errores sintácticos, y el enlace del código objeto obtenido con las correspondientes bibliotecas.

Entradas:

- Catálogo de normas de IT CGAE
- Especificaciones de construcción del sistema de información

Salidas:

- Producto *software*
    - Código fuente de los componentes

Técnicas:

- Programación orientada a objetos
- Lenguaje Java
- *Frameworks* de desarrollo
- Control de versiones
- Gestión de dependencias

Participantes:

- Desarrolladores

5.3.2.2. Generación del código de los procedimientos de operación y seguridad

El objetivo de esta tarea es generar los procedimientos de operación y administración del sistema de información, así como los procedimientos de seguridad y control de acceso, necesarios para ejecutar el sistema una vez que se haya implantado y esté en producción.

Para la generación de dichos procedimientos se tienen en cuenta, también, los estándares y normas de la instalación recogidos en el catálogo de normas de IT CGAE.

Entradas:

- Catálogo de normas de IT CGAE
- Diseño de la arquitectura del sistema
- Entorno tecnológico del sistema
- Procedimientos de operación y administración del sistema
- Procedimientos de seguridad y control de acceso
- Producto *software*

Salidas:

- Producto *software*
    - Código de procedimientos de operación y administración del sistema
    - Código de procedimientos de seguridad y control de acceso

Técnicas:

- Programación orientada a objetos
- Lenguaje Java
- *Scripting*
- Lenguaje SQL
- *Frameworks* de seguridad (autenticación y autorización)
- Control de versiones

Participantes:

- Desarrolladores
- Técnicos de sistemas
- Experto en base de datos

5.3.3. Ejecución de las pruebas unitarias y de integración

En esta actividad se realizan las pruebas unitarias de cada uno de los componentes del sistema de información, una vez codificados, con el objeto de comprobar que su estructura es correcta y que se ajustan a la funcionalidad establecida.

En el plan de pruebas se ha definido el entorno necesario para la realización de cada nivel de prueba, así como las verificaciones asociadas a las pruebas unitarias, la coordinación y secuencia a seguir en la ejecución de las mismas y los criterios de registro y aceptación de los resultados.

Las pruebas unitarias se irán ejecutando en paralelo a la actividad *Implementación del código de componentes*.

Una vez ejecutadas las pruebas unitarias para cada uno de los componentes, se procederá a ejecutar las pruebas de integración.

El objetivo de las pruebas de integración es verificar si los componentes o subsistemas interactúan correctamente a través de sus interfaces, tanto internas como externas, cubren la funcionalidad establecida, y se ajustan a los requisitos especificados en las verificaciones correspondientes.

La estrategia a seguir en las pruebas de integración se establece en el plan de pruebas, donde se habrá tenido en cuenta el plan de integración del sistema de información, siempre y cuando se haya especificado en la tarea *Definición de componentes y subsistemas de construcción*.

En la realización de estas pruebas es importante comprobar la cobertura de los requisitos, dado que su incumplimiento puede comprometer la aceptación del sistema por el equipo de operación responsable de realizar las pruebas de implantación del sistema, que se llevarán a cabo en el proceso Implantación y Aceptación del Sistema.

Los datos utilizados para las pruebas deberán ser lo más similares posible a los datos reales que se utilizarán en el entorno de producción. Cuando sea conveniente, en el caso de proyectos evolutivos sobre sistemas ya en producción, se solicitará al área de Explotación la realización de volcados de datos desde el entorno de producción al de pruebas para contar con datos más realistas. Al realizar estos volcados deberán tenerse en cuenta las consideraciones de privacidad derivadas de la Ley Orgánica de Protección de Datos de carácter personal (LOPD); en particular, si en los datos de producción a volcar existen datos personales, se deberán aplicar modificaciones sobre los datos en el entorno de prueba que eviten la identificación de sujetos reales (alteración de nombres y apellidos y/o documentos de identidad).

Esta actividad se descompone en las siguientes tareas:

5.3.3.1. Preparación del entorno de las pruebas unitarias

En esta tarea se preparan todos los recursos necesarios para realizar las pruebas unitarias de cada uno de los componentes del sistema de información.

Para ello, se asegura la disponibilidad del entorno y de los datos necesarios para ejecutar estas pruebas, se preparan las bibliotecas o librerías oportunas para la realización de las mismas, así como los procedimientos manuales o automáticos necesarios, conforme a la especificación del entorno definida en el plan de pruebas.

Se codifican las unidades de prueba automatizada, apoyándose en el *framework* de *testing* (JUnit). Esta tarea realmente debe realizarse en paralelo a la implementación de código, e incluso, idealmente, se deberían codificar las unidades de prueba antes que el propi código.

Entradas:

- Plan de pruebas

Salidas:

- Entorno de pruebas unitarias

Técnicas:

- Programación orientada a objetos
- Lenguaje Java
- *Framework* de *testing*
- Control de versiones

Participantes:

- Desarrolladores
- Técnicos de sistemas

5.3.3.2. Realización y evaluación de las pruebas unitarias

El objetivo de esta tarea es comprobar el correcto funcionamiento de los componentes del sistema de información, codificados en la actividad *Generación del código de los componentes y procedimientos*, conforme a las verificaciones establecidas en el plan de pruebas para el nivel de pruebas unitarias, en la actividad *Especificación técnica del plan de pruebas*.

Para cada verificación establecida, se realizan las pruebas con los casos de pruebas asociados, efectuando el correspondiente análisis y evaluación de los resultados, y generando un registro conforme a los criterios establecidos en el plan de pruebas.

Se distinguen aquí dos tipos de pruebas unitarias:

- **Automatizadas**: Se definen sobre un *framework* de *testing* (JUnit) y se programan para ser ejecutadas de forma automática en cada ciclo de construcción. Permiten evaluar si los métodos de los componentes desarrollados cumplen los comportamientos esperados, mediante aserciones que pueden ser verdaderas o falsas. Este tipo de pruebas son especialmente útiles en pruebas de regresión, para verificar que cambios en el código no provocan malfuncionamiento en métodos ya desarrollados. Siempre que sea posible, es preferible este tipo de pruebas a las manuales.
- **Manuales**: Las realiza el desarrollador interactuando directamente con el componente en su entorno de desarrollo. Este tipo de pruebas son necesarias para validar, por ejemplo, que la interfaz de usuario se comporta como se espera (ya que eso es más difícil de automatizar). En la realización de estas pruebas deberán observarse las mismas recomendaciones ya expuestas en el apartado 4.4.5.2 del presente documento.

Seguidamente, se analizan los resultados de las pruebas unitarias, evaluándose las mismas para comprobar que los resultados son los esperados. Si los resultados no son los esperados hay que proceder a realizar las correcciones pertinentes.

Entradas:

- Producto *software*
- Entorno de pruebas unitarias
- Plan de pruebas

Salidas:

- Resultado y evaluación de las pruebas unitarias

Técnicas:

- *Testing*
- Automatización de la construcción
- *Framework* de *testing*

Participantes:

- Desarrolladores
- Equipo de pruebas

5.3.3.3. Preparación del entorno de las pruebas de integración

En esta tarea se disponen todos los recursos necesarios para realizar las pruebas de integración de los componentes y subsistemas que conforman el sistema de información.

Para ello, se asegura la disponibilidad del entorno y de los datos necesarios para ejecutar estas pruebas, se preparan las bibliotecas o librerías que se estimen oportunas para la realización de las mismas, así como los procedimientos manuales o automáticos asociados, conforme a la especificación del entorno definida en el plan de pruebas.

Entradas:

- Plan de pruebas

Salidas:

- Entorno de pruebas de integración

Técnicas:

- *Testing*

Participantes:

- Desarrolladores
- Técnicos de sistemas
- Equipo de soporte

5.3.3.4. Realización y evaluación de las pruebas de integración

El objetivo de esta tarea es verificar el correcto funcionamiento de las interfaces existentes entre los distintos componentes y subsistemas, conforme a las verificaciones establecidas para el nivel de pruebas de integración.

Para cada verificación establecida, se realizan las pruebas con los casos de pruebas asociados, efectuando el correspondiente análisis e informe de los resultados de cada verificación, y generando un registro conforme a los criterios establecidos en el plan de pruebas.

Una vez realizadas las pruebas se procederá a su evaluación, en la que se recoge el grado de cumplimiento de las pruebas y que consiste en:

- Comparar los resultados obtenidos con los esperados
- Identificar el origen de cada problema detectado para poder remitirlo a quien proceda, determinar la envergadura de las modificaciones y qué acciones deben llevarse a cabo para resolverlo de forma satisfactoria.
- Indicar si el plan de pruebas debe volver a realizarse total o parcialmente, y si será necesario contemplar nuevos casos de prueba no considerados anteriormente

Entradas:

- Producto *software*
- Entorno de pruebas de integración
- Plan de pruebas

Salidas:

- Resultado y evaluación de las pruebas de integración

Técnicas:

- *Testing*
- Automatización de la construcción

Participantes:

- Analistas
- Desarrolladores
- Equipo de pruebas

5.3.4. Elaboración de los manuales de usuario

Esta actividad comprende la elaboración de los manuales de usuario del sistema. Es importante destacar que aunque esta actividad se represente en el diagrama de la fase en cuarto lugar, no tiene por qué realizarse necesariamente de forma secuencial con respecto de las actividades precedentes; los manuales pueden ir realizándose en paralelo con la actividad de pruebas o incluso, en algunos casos, con la de codificación (cuando el sistema comprenda varios módulos; se puede ir avanzando en el manual de un módulo mientras se implementan otros). En cualquier caso, una vez finalizada la actividad de pruebas, se deberán revisar los manuales (si se empezaron a redactar con anterioridad) para actualizarlos convenientemente.

Esta actividad incluye una única tarea:

5.3.4.1. Elaboración de los manuales de usuario

El objetivo de esta tarea es elaborar la documentación de usuario, tanto usuario final como de explotación, de acuerdo a los requisitos establecidos en la tarea *Especificación de requisitos de documentación de usuario*, y recogidos en el catálogo de requisitos.

Los requisitos de documentación especifican aspectos relativos a los tipos de documentos a elaborar y estándares a seguir en la generación de los mismos, y para cada uno de ellos:

- Formato y soporte en el que se desarrollarán
- Estructura
- Distribución y mantenimiento de la documentación y número de copias a editar.

Entradas:

- Catálogo de requisitos
- Producto *software*

Salidas:

- Manual(es) de usuario final
- Manual(es) de explotación

Técnicas:

- Ofimática

Participantes:

- Equipo de proyecto

5.3.5. Construcción de componentes y procedimientos de migración y carga inicial de datos

En esta actividad se realizará todo lo necesario para la carga inicial de datos del sistema así como, en su caso, la migración de datos desde un sistema o versión de sistema anterior. Esto incluye tanto el diseño, implementación y prueba de programas de carga / migración como la preparación de scripts de base de datos (scripts SQL) y de ficheros de datos iniciales a cargar.

En el caso de carga inicial de datos es conveniente que esta actividad se realice en paralelo a la construcción del sistema, a medida que sea necesario, para facilitar las pruebas del propio sistema.

En esta actividad podemos distinguir tres tareas:

5.3.5.1. Preparación del entorno de migración y carga inicial de datos

Se dispone el entorno en el que se van a construir los componentes y procedimientos de migración y carga inicial de datos, considerando las bibliotecas o librerías a utilizar, herramientas o utilidades específicas para la conversión, y compiladores, entre otros, cuya necesidad se habrá establecido en la tarea *Especificación del entorno de migración*. Obsérvese que, en general, se tratará del mismo entorno utilizado para la construcción del propio sistema, con la posible adición de algunos componentes o bibliotecas.

Asimismo, se determinan los datos necesarios para realizar las pruebas de los componentes y procedimientos asociados y se configura el entorno de acuerdo a dichas necesidades.

Entradas:

- Plan de migración y carga inicial de datos

Salidas:

- Entorno de migración

Técnicas:

- Instalación y configuración de entorno de desarrollo

Participantes:

- Equipo de proyecto
- Técnicos de sistemas

5.3.5.2. Generación del código de los componentes y procedimientos de migración y carga inicial de datos

El objetivo de esta tarea es la generación del código correspondiente a los procedimientos y componentes necesarios para llevar a cabo la migración, definidos en el plan de migración y carga inicial de datos obtenido en las tareas *Diseño de procedimientos de migración y carga inicial* y *Diseño detallado de componentes de migración y carga inicial*.

Para generar el código fuente se tienen en cuenta los estándares de nomenclatura y codificación utilizados por RedAbogacía y la guía de buenas prácticas recogida en el anexo C.

Entradas:

- Plan de migración y carga inicial de datos
- Catálogo de normas de IT CGAE

Salidas:

- Código fuente de los componentes de migración y carga inicial de datos
- Procedimientos de migración y carga inicial de datos

Técnicas:

- Descripción textual
- Diagramación
- Programación

Participantes:

- Equipo de proyecto

5.3.5.3. Realización y evaluación de las pruebas de migración y carga inicial de datos

El objetivo de esta tarea es efectuar las pruebas de los distintos componentes y procedimientos de migración y evaluar su resultado. Esta evaluación recoge el grado de cumplimiento de las mismas, y consiste en:

- Comparar los resultados obtenidos con los esperados
- Identificar el origen de cada problema detectado para poder remitirlo a quien proceda, determinar la envergadura de las modificaciones y qué acciones deben llevarse a cabo para resolverlo de forma satisfactoria
- Indicar si el plan de pruebas debe volver a realizarse total o parcialmente, y si será necesario contemplar nuevos casos de prueba no considerados anteriormente

Entradas:

- Plan de migración y carga inicial de datos
- Código fuente de los componentes de migración y carga inicial de datos
- Procedimientos de migración y carga inicial de datos

Salidas:

- Resultado de las pruebas de migración y carga inicial de datos
- Evaluación del resultado de las pruebas de migración y carga inicial de datos

Técnicas:

- Testing (pruebas unitarias y de integración)

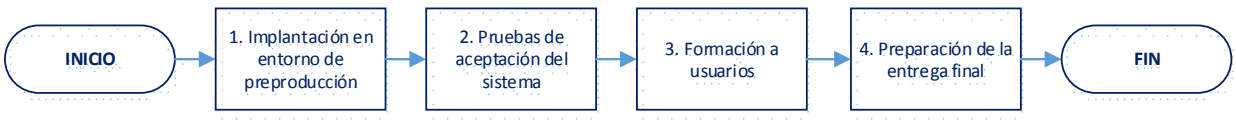
Participantes:

- Equipo de proyecto

5.4. Implantación y Aceptación del Sistema (IAS)

En esta fase se prepara la entrega final del Sistema de Información, se despliega la misma en el entorno de preproducción, se realizan las pruebas finales de aceptación sobre dicho entorno y, una vez obtenido el OK al sistema, se procede a su entrega para el entorno de producción.

El siguiente diagrama representa el flujo de esta fase:



5.4.1. Implantación en entorno de preproducción

En esta actividad se prepara la entrega del sistema construido para su despliegue en el entorno de preproducción y se solicita dicho despliegue al equipo de Explotación de IT.

Esta actividad incluye una única tarea:

5.4.1.1. Preparación y solicitud de entrega en preproducción

Se prepara la entrega conteniendo el proyecto software, los scripts de carga inicial y migración de datos y la documentación de despliegue, siguiendo la instrucción técnica **ITPGS.12-01-Proceso Gestión de Entregas de Software-Vx_xx** (donde x_xx representa la última versión liberada del documento). El documento de entrega deberá seguir el modelo normalizado de IT CGAE para este tipo de documento.

Entradas:

- Catálogo de normas de IT CGAE
- Sistema software construido
- Código fuente de los componentes de migración y carga inicial de datos
- Procedimientos de migración y carga inicial de datos

Salidas:

- Paquete de entrega a preproducción
- Solicitud de entrega a preproducción en la herramienta corporativa (EasyVista)

Técnicas:

- Gestión de entregas
- Descripción textual

Participantes:

- Equipo de proyecto

5.4.2. Pruebas de aceptación del sistema

Una vez implantado el sistema completo en el entorno de preproducción, se deberá realizar un ciclo completo de pruebas sobre el producto terminado de cara a asegurar su perfecto funcionamiento, y para obtener la aceptación final por parte de los responsables designados por el usuario / cliente.

Esta actividad incluye una única tarea:

5.4.2.1. Realización de las pruebas finales y de aceptación en preproducción

Las pruebas a realizar serán tanto funcionales como de seguridad, carga y rendimiento, así como pruebas de aceptación, de la misma forma ya descrita para MEDRA^{A} en el apartado 4.5.1

Entradas:

- Sistema en preproducción
- Plan de pruebas

Salidas:

- Sistema probado y aceptado

Técnicas:

- Testing (funcional, de seguridad, de carga y rendimiento)
- Actas de aceptación

Participantes:

- Jefe de proyecto
- Analista(s) funcional(es)
- Representantes (a nivel directivo) de los usuarios / clientes

5.4.3. Formación a usuarios

En los casos en que así se requiera, el equipo de proyecto deberá encargarse de impartir formación en el uso del sistema, bien directamente a los usuarios finales del mismo, bien a un equipo de formadores que a su vez impartan la formación a los usuarios.

La formación puede obviarse si la sencillez e *intuitividad* del sistema la hacen innecesaria, o si se han preparado mecanismos de autoformación, por ejemplo, a través de la plataforma de formación de IT CGAE.

Esta actividad incluye una única tarea:

5.4.3.1. Realización de la formación a usuarios finales

La formación a los usuarios finales se realizará sobre el entorno de preproducción ya instalado y con la carga inicial y, en su caso, la migración de datos realizada. Los contenidos de la formación deberán ajustarse al contenido del manual de usuario, aunque el orden y la profundidad en que se traten los temas pueden diferir por razones didácticas o de tiempo disponible.

Si se puede disponer de formadores especializados, el equipo de proyecto formará a os formadores, y estos a su vez a los usuarios finales.

Entradas:

- Sistema en preproducción
- Manual de usuario

Salidas:

- Usuarios formados

Técnicas:

- Técnicas didácticas: ejemplos prácticos, presentaciones,…

Participantes:

- Equipo de proyecto
- Usuarios
- Formadores (en caso de disponibilidad de formadores especializados)

5.4.4. Preparación de la entrega final

En esta actividad se prepara la entrega del sistema construido para su despliegue en el entorno de producción y se solicita dicho despliegue al equipo de Explotación de IT. Obsérvese que, si bien en el diagrama de la fase se representan en secuencia, esta actividad puede realizarse en paralelo a la actividad de *formación a usuarios*.

Esta actividad incluye una única tarea:

5.4.4.1. Preparación y solicitud de entrega en producción

Se prepara la entrega conteniendo el proyecto software, los scripts de carga inicial y migración de datos y la documentación de despliegue, siguiendo la instrucción técnica **ITPGS.12-01-Proceso Gestión de Entregas de Software-Vx_xx** (donde x_xx representa la última versión liberada del documento). El documento de entrega deberá seguir el modelo normalizado de IT CGAE para este tipo de documento.

Entradas:

- Catálogo de normas de IT CGAE
- Sistema software construido
- Código fuente de los componentes de migración y carga inicial de datos
- Procedimientos de migración y carga inicial de datos

Salidas:

- Paquete de entrega a producción
- Solicitud de entrega a producción en la herramienta corporativa (EasyVista)

Técnicas:

- Gestión de entregas
- Descripción textual

Participantes:

- Equipo de proyecto

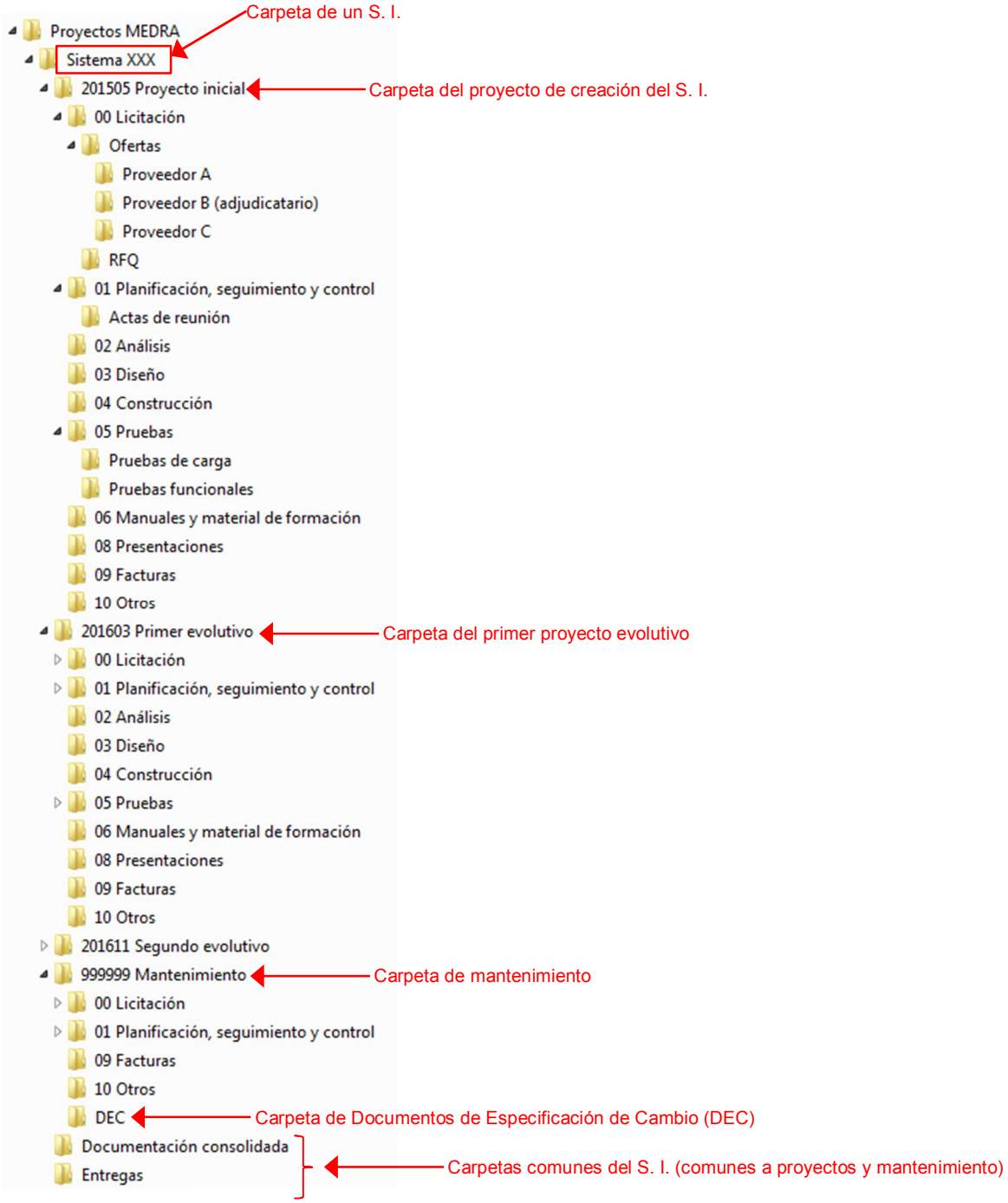
6. Organización y estructura de los proyectos

El presente capítulo describe cómo se organizará la documentación, código fuente y demás productos generados a lo largo del ciclo de vida de un proyecto de desarrollo de software en IT CGAE.

6.1. Carpetas de documentación – Proyectos de desarrollo

La documentación de los proyectos de desarrollo se almacenará en el recurso compartido habitualmente como unidad **H:** (ruta UNC \\192.168.20.1\It-cgae) debajo de la carpeta **3.-Desarrollo\Proyectos MEDRA,** existiendo una carpeta por cada sistema de información o aplicación.

La estructura de subcarpetas de la carpeta de cada S.I. será la siguiente:



Bajo la carpeta correspondiente a cada Sistema de Información o aplicación (cuyo nombre deberá corresponder con el nombre o acrónimo más comúnmente utilizado para ese sistema; p. ej.: *SIGA*, *Lexnet Abogacía*, etc.), existirá una carpeta para cada proyecto de desarrollo correspondiente a ese sistema, una carpeta para el mantenimiento (pequeñas tareas de mantenimiento "continuo" no gestionadas como proyectos), y carpetas comunes al S.I. (carpeta de Entregas, Documentación consolidada y otras que se pueda considerar necesario). Estas últimas contendrán documentación que sea común a todos los proyectos y al mantenimiento.

En particular, la subcarpeta **Entregas** contendrá todo el historial de entregas a PRE y PRO, de modo que en una misma carpeta se encontrarán todas las entregas cronológicamente, independientemente de que hayan sido generadas desde un proyecto o a partir de una tarea de mantenimiento.

La carpeta **Documentación consolidada** contendrá la documentación técnica cerrada (de análisis, diseño, operación, etc.) del sistema actualizada a la última versión que se encuentre en producción.

Dentro de las carpetas de cada proyecto o mantenimiento se encontrará la documentación de gestión (planificación, seguimiento, licitación si procede…) y la documentación técnica parcial específica del alcance de ese evolutivo o mantenimiento (por ejemplo, los requisitos específicos del evolutivo y las versiones de documentos técnicos correspondientes a esa versión del S.I.).

Las subcarpetas de proyecto se prefijarán con la fecha de inicio del proyecto en formato AAAAMM para que aparezcan ordenadas cronológicamente dentro de la carpeta del S.I. Asimismo, la subcarpeta de mantenimiento se prefijará con la etiqueta 999999 para que aparezca justo a continuación de las subcarpetas de proyecto.

La subcarpeta del primer proyecto (el proyecto por el cual se crea el S. I.) de denominará "AAAAMM Proyecto inicial". Las subcarpetas de los proyectos evolutivos subsiguientes se nombrarán (además de con AAAAMM) con un nombre significativo que identifique el alcance del evolutivo, si es posible (por ejemplo: "*Adaptación nuevo WS MJU*", o si no, simplemente como "*Primer evolutivo*", "*Segundo evolutivo*", etc.

Dentro de cada subcarpeta de proyecto o mantenimiento:

La carpeta **00 Licitación** sólo existirá si el proyecto de desarrollo o el mantenimiento se subcontrata y para ello se realiza un proceso de licitación (especificación de RFP/RFQ, petición, recepción y adjudicación de ofertas). Bajo la subcarpeta **Ofertas** se creará una carpeta por cada proveedor que se presente para almacenar su documentación correspondiente (oferta y documentación anexa). Cuando uno de ellos resulte seleccionado, se renombrará su carpeta añadiendo al final **(adjudicatario)**.

La carpeta **09 Facturas** también existirá únicamente cuando se haya contratado a un proveedor externo.

Salvo excepciones justificadas, no se crearán directamente bajo la carpeta de proyecto otras subcarpetas aparte de las especificadas en el árbol anterior; cualquier otra carpeta particular del proyecto que no encaje en ninguna subcarpeta se creará bajo **10 Otros**. Tampoco se archivarán documentos directamente bajo la carpeta de proyecto o mantenimiento, debiendo estar estos dentro de subcarpetas.

6.2. Documentación mínima exigida

Para cada proyecto de desarrollo deberá existir un conjunto de documentos que serán considerados los mínimos imprescindibles. Esto no impide que puedan existir en las carpetas de documentación más documentos que los aquí reseñados, o que estos contengan más apartados; los aquí citados son, como se indica, **un mínimo** a cumplir. Deberá existir proporcionalidad entre la extensión y complejidad del desarrollo software y la documentación que lo acompaña.

Se deberán utilizar, siempre que existan, plantillas normalizadas para los documentos. Cuando no exista plantilla para un tipo específico de documento, se partirá de la plantilla general básica de RedAbogacía **2013-Plantilla Word ITCGAE.dotx / 2013-Plantilla_Redabogacia_PPT.pptx**).

Las plantillas específicas de MEDRA se hallan en la ruta:

H:\3.- Desarrollo\MEDRA\Plantillas de documentos

Las plantillas generales de RedAbogacía se encuentran en la ruta:

H:\1.- General\Plantillas de documentos

Documento	Contenido mínimo	Carpeta	Observaciones
RFQ	Según plantilla normalizada	00 Licitación\RFQ	Sólo para proyectos subcontratados a proveedores
Oferta(s)	Lo exigido por la RFQ	00 Licitación\Ofertas*Carpeta_proveedor*	Sólo para proyectos subcontratados a proveedores
Plan de proyecto	Apartados: - Alcance - Arquitectura general del sistema - Equipo de proyecto - Participantes - Evaluación del paradigma metodológico - Planificación y cronograma	01 Planificación, seguimiento y control	La planificación reflejada en el plan de proyecto será la inicial
Planificación actualizada	Planificación en formato Gantt o similar	01 Planificación, seguimiento y control	Se actualizará periódicamente para reflejar la situación del proyecto a lo largo de su ejecución. Se deberá conservar el histórico de versiones de planificación

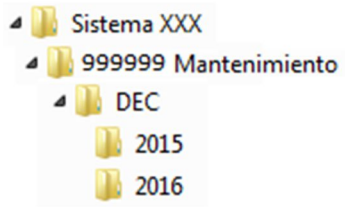
Documento	Contenido mínimo	Carpeta	Observaciones
Actas de reunión	Según plantilla normalizada	01 Planificación, seguimiento y control\Actas de reunión	Se deberán conservar actas de las reuniones de seguimiento, al menos de las más relevantes (cuando se tomen decisiones que afecten de forma importante al alcance o planificación del proyecto)
Catálogo de requisitos	Requisitos funcionales y no funcionales	02 Análisis	Más extenso y detallado en MEDRA^{C} que en MEDRA^{A}. Este catálogo puede ser un documento ofimático o bien residir en una herramienta CASE con soporte específico para requisitos
Análisis funcional	Apartados para MEDRA^{A}: - Descripción funcional general del sistema - *Mock-ups* de interfaz de usuario	02 Análisis	Más extenso y detallado en MEDRA^{C} que en MEDRA^{A}
	Apartados para MEDRA^{C}: - Especificación de casos de uso - Subsistemas (si aplica) - Especificación de interfaz de usuario		
Diseño técnico	Apartados: - Arquitectura del sistema - Requisitos de diseño - Catálogo de excepciones - Modelo de datos - Modelo de clases - Realización de casos de uso	03 Diseño	Más extenso y detallado en MEDRA^{C} que en MEDRA^{A}
Informe(s) de pruebas	No es necesario un documento muy exhaustivo ni formal, basta con un Excel que recoja las principales pruebas realizadas, especialmente las que han generado error	05 Pruebas\Pruebas funcionales	En cada subcarpeta se incluirán los informes de pruebas según su tipo (funcional o de carga / estrés)
		05 Pruebas\Pruebas de carga	
Manual de usuario		06 Manuales y material de formación	

Documento	Contenido mínimo	Carpeta	Observaciones
Manual de operación	Apartados: - Alcance (resumen) - Arquitectura del sistema (a alto nivel) - Parámetros de configuración - Recursos requeridos (directorios de trabajo, base de datos que utiliza, certificados, conexiones con otros sistemas, etc.) - Operaciones de mantenimiento	06 Manuales y material de formación	El alcance y la arquitectura deben ser resumidos y a alto nivel; se trata de que sólo con este documento alguien se haga una idea general de para qué sirve el sistema y cómo está implementado. Este documento debe reflejar el estado de la versión del sistema que está en producción.
Documento(s) de entrega	Según plantilla normalizada	07 Entregas*Carpeta de entrega*	Véase **ITPGS.12-01-Proceso Gestión de Entregas de Software-Vx_xx.docx** para la estructura de las carpetas de entrega

6.3. Carpetas de documentación – Mantenimientos

La documentación asociada a los "pequeños mantenimientos" o "mantenimientos continuos" (aquellos que no van asociados a una gestión completa de proyecto) se almacenará en el recurso compartido habitualmente como unidad **H:** (ruta UNC \\192.168.20.1\It-cgae) debajo de la carpeta **3.-Desarrollo\Proyectos MEDRA*Carpeta del S.I.*\999999 Mantenimiento,** como se puede apreciar en la figura del apartado 6.1

La carpeta de cada mantenimiento podrá contener las subcarpetas que se consideren necesarias siguiendo la misma nomenclatura que las carpetas de proyectos (p. ej. ***00 Licitación***, ***09 Facturas***, etc.). Además contendrá una subcarpeta denominada **DEC** que contendrá los documentos de especificación de cambios (véase apartado 7.2), agrupados a su vez en subcarpetas por años, como se muestra en la figura siguiente:



6.4. Nomenclatura y versionado de documentos

Los documentos de proyecto se deberán nombrar siguiendo los criterios de nomenclatura expuestos en el presente apartado.

El nombre de cualquier documento generado en los proyectos MEDRA (nos referimos aquí al nombre de archivo, no al título del documento) tendrá la siguiente estructura:

KKK..K_*nombreDescriptivo*[_vX.YY].ext

donde:

- KKK..K será un código representativo del servicio o sistema. Deberán utilizarse códigos o acrónimos reconocibles y ampliamente aceptados por ITCGAE y sus usuarios; por ejemplo: *SIGA*, *CEN* (Censo), *ACA*, etc. No debe tener una longitud específica, pero se recomienda que no sea mayor de 6 caracteres. Para documentos que no correspondan a un proyecto concreto se utilizará, en general, el código ITCGAE; o si el documento afecta a dos sistemas concretos (por ejemplo una especificación de interfaz entre dos sistemas) podrán concatenarse los códigos de ambos, separados por un "guion bajo".

  Cuando el alcance del documento se refiera a un evolutivo de un sistema, se podrá especificar en este código, para evitar confusiones, la versión del sistema a la que corresponde el documento. Por ejemplo, si se trata de un documento para la versión 2 de ACA el código podría ser: *ACA_V2*.
- *nombreDescriptivo* será un nombre descriptivo del contenido del documento. En particular, cuando se trate de documentos exigidos por la metodología, se establecerá el nombre recogido en la misma para ese tipo de documento; por ejemplo; *RFQ*, *AnálisisFuncional*, *Requisitos*, *DiseñoTécnico*, etc.
  En el caso de documentos que se refieran a eventos que ocurren en una fecha determinada (por ejemplo, actas de reunión), se concatenará la fecha del evento en formato AAAAMMDD separada por un guion bajo; p. ej.: *ActaReunión_20150908*.
  Para los documentos de especificación de cambios descritos en el apartado 7.2 de la metodología, el *nombreDescriptivo* se compondrá del prefijo **DEC** seguido del código de la petición de cambio en EasyVista, separados por guion bajo. **Ejemplo: SIGA_DEC_R1509_0086.docx**
- X.YY serán el número de versión y sub-versión del documento, según el criterio especificado más adelante. Obsérvese que este número sólo se especificará en el nombre del archivo cuando se trate de versiones históricas o de una versión en borrador. La última versión aprobada de un documento se archivará sin el número de versión codificado en su nombre. De este modo, cuando se cree un hipervínculo a un documento (por ejemplo, desde EasyVista, la Wiki de RedAbogacía, etc.), el vínculo se podrá crear al documento en su última versión aprobada y el enlace no se romperá aunque se actualice dicha versión.
- ext será la extensión del documento según el formato de archivo de que se trate (*.docx*, *.xslx*, *.pdf*, etc.)

El versionado de documentos seguirá las siguientes reglas:

- La primera versión de un documento se numerará como 1.00
- Mientras un documento se encuentra en fase de elaboración y revisión, sin haber sido dado por cerrado (y en su caso, aprobado), se considerará como un "borrador". En este estado, el documento se codificará con el número de versión en su nombre (como se ha indicado antes) y tanto en su portada como en los pies de página se deberá especificar que se trata de un borrador.
- Cuando se realice un cambio menor sobre un documento que ya haya sido circulado a otras personas aparte de su autor, se incrementará el número de "sub-versión", que se especificará siempre con 2 dígitos. El autor podrá incrementar también el número de "sub-versión" a su criterio, aun cuando no haya circulado el documento a otras personas. Se considera un cambio menor aquél que no suponga una modificación relevante del contenido del documento, a criterio de su autor.
- Cuando se realice un cambio de mayor entidad, se incrementará el número de versión y se inicializará el número de "sub-versión" a 00.
- Cuando un documento que se encontraba en estado de borrador se dé por cerrado (y, en su caso, aprobado), el autor del mismo:
    - Eliminará la palabra "borrador" de la portada y los pies de página
    - Eliminará el número de versión del nombre de archivo del documento
    - Archivará el documento en la subcarpeta que corresponda de la carpeta de proyecto
    - Enviará un correo electrónico a las personas interesadas, si procede, con un hipervínculo al documento tal y como ha quedado archivado en la carpeta de proyecto
    - Cualquier enlace que deba hacerse desde una herramienta corporativa a un documento del proyecto debería enlazarse utilizando el nombre de archivo sin versión, para que el enlace no se rompa al evolucionar las versiones y apunte siempre a la última versión cerrada del mismo
- Cuando se cierre una versión de un documento del cual ya existía una versión anterior en la carpeta de proyecto, el autor de la última versión procederá como sigue:
    - Moverá la versión preexistente a una subcarpeta denominada **Histórico** (dentro de la misma carpeta en la que estuviese el documento). SI fuese necesario, creará esa subcarpeta
    - Renombrará el archivo de la versión preexistente (dentro de **Histórico**) poniendo el sufijo con su número de versión correspondiente
    - Por último, guardará la nueva versión "cerrada" en la subcarpeta correspondiente (donde antes estaba la versión previa), tal y como se ha especificado en el punto anterior

6.5. Código fuente

El código fuente se almacenará en la herramienta corporativa de control de versiones (Subversion) que a la fecha del presente documento se encuentra accesible a través de la URL:

https://CGAE-ITA83:443/svn

Este repositorio será accesible al área de Desarrollo de ITCGAE y a los responsables de seguridad de la organización. Siempre que sea necesario se habilitará el acceso para la inspección y auditoría del código fuente.

Existirá un repositorio por cada "área funcional", entendiendo por tal el conjunto de aplicaciones o servicios que están "emparentados" funcionalmente. Cada repositorio se nombrará con el prefijo "rep" más un nombre descriptivo del área funcional. Por ejemplo:

- **repsiga**: Repositorio para el proyecto SIGA y otros relacionados con el mismo
- **repcenso**: repositorio para proyectos relacionados con el censo de colegiados
- **repmobile**: Repositorio para aplicaciones móviles
- **repabogacia**: Repositorio para proyectos de ámbito general o que no encajen en otro grupo
- etc…

Bajo cada repositorio existirá una carpeta por cada aplicación, servicio o módulo que suponga una unidad de compilación independiente; por ejemplo:

- repsiga
  - eCOM
  - SIGA
  - SIGADoc
  - SIGAServices

Cada una de esas carpetas contendrá a su vez tres subcarpetas:

- **trunk**: donde se almacenará la versión "principal" del código fuente. Se considerará la versión principal aquella que coincida con la que está en producción en un momento dado. Para proyectos nuevos que aún no han llegado a producción, la versión **trunk** será la única que exista (la que se está desarrollando)
- **branches**: donde se almacenarán versiones "rama". Cuando un sistema tenga una versión en producción y se acometa un evolutivo sobre la misma que vaya a tener una duración de varios días (no para pequeñas correcciones y parches), la nueva versión se iniciará como una rama (*branch*). Cuando la nueva versión suba a producción, se procederá de la siguiente forma:
  - se copiará o moverá el código existente en la rama hacia el principal (*trunk*)
  - si se va a iniciar inmediatamente el desarrollo de una nueva versión, se puede renombrar la rama con el código de la nueva versión para reutilizarla (en tal caso deberemos haber "copiado" y no "movido" la rama al **trunk**. En caso contrario, lo aconsejable es "mover" la rama para que así desaparezca de **branches**
- **tags:** donde se almacenarán etiquetados de versiones y parches. Cada vez que se realice la entrega a producción de una versión o un parche, se etiquetará el código fuente correspondiente con la misma codificación que la entrega (véase ITPGS.12-01-Proceso Gestión de Entregas de Software-Vx_xx.docx). Esto permitirá volver fácilmente a una versión / parche concreta.

Como ya se ha mencionado, los pequeños cambios o parches (en general, correctivos) que tengan un ciclo de desarrollo corto (pocos días) no generarán una nueva versión en una rama, sino que se realizarán sobre el código principal (*trunk*). Debe recordarse en cualquier caso etiquetar el código cada vez que se hace una entrega de versión de uno de estos parches para tener controlado el historial de versiones de código y poder volver en cualquier momento a una versión / parche concreto.

Para proyectos subcontratados a proveedores, se deberá acordar con los mismos mecanismos de entrega del código fuente que permitan actualizar dicho código periódicamente en la herramienta corporativa de IT CGAE. En cualquier caso, a la finalización del proyecto, deberá quedar en la herramienta corporativa almacenada la última versión entregada del código fuente.

7. Gestión de pequeños mantenimientos

Todo lo descrito en los capítulos 3 a 6 del presente documento es de aplicación a los proyectos de desarrollo de *software* llevados a cabo en IT CGAE, entendiendo por "proyecto": un conjunto ordenado de tareas que se desarrolla a lo largo de un periodo de tiempo relativamente largo (en general, un mínimo de un mes), en el que normalmente participan varias personas, y que tiene como objetivo el producir un nuevo sistema software, o una nueva versión de un sistema ya existente, incluyendo un conjunto amplio de cambios en su alcance.

Sin embargo, en muchas ocasiones es necesario realizar un cambio concreto en un sistema existente con cierta premura, sin esperar a que se planifique una nueva versión del sistema. En general, esta situaciones se producen cuando se detecta un error en el sistema en producción y se requiere una corrección rápida del mismo (aunque también podrían incluirse en este caso pequeños cambios de funcionalidad que, sin corresponder a correcciones de errores, se consideren urgentes por el beneficio que aportan al usuario).

Cuando se da esta circunstancia, no tiene sentido plantear la ejecución de un proyecto completo con todas sus implicaciones metodológicas para acometer un cambio menor. Por ello, el presente capítulo describe cómo tratar estos pequeños cambios de forma ordenada, pero sin toda la carga metodológica que requiere un proyecto de mayor envergadura.

7.1. Gestión de cambios

La corrección de un error, o el añadir o mejorar una pequeña funcionalidad a un sistema en producción implica un **cambio** en ese *software*. Por tanto, esa tarea deberá gestionarse de acuerdo al procedimiento de gestión de cambios estipulado para IT CGAE dentro de su Sistema Integrado de Gestión, que se alinea con la norma ISO 20.000 – Gestión del servicio de TI.

El manual del procedimiento de gestión de cambios para IT CGAE se encuentra recogido en el documento **PGS.11 Gestión de Cambios-VX.docx** (donde **X** representa la última versión liberada del documento).

En los casos en los que el cambio se derive de la detección de una incidencia en producción, se deberá tener también en consideración el procedimiento de gestión de incidencias (recogido en **PGS.08 Gestión de Incidencias-VX.docx**) y la instrucción técnica **ITPGS.08-01 Guía Técnica de Gestión de Incidencias_vX.docx** (donde **X** representa la última versión liberada de los documentos).

En los documentos citados se pueden encontrar especificados en detalle los pasos a seguir para la gestión de un cambio en un sistema en producción. Sin embargo, el procedimiento se refiere a cambios *en general*, y no es específico para los cambios que implican desarrollo de software, por lo que en los siguientes apartados del presente capítulo se dan indicaciones más específicas para este tipo de cambios.

7.2. Documentación del cambio

Como ya se ha indicado en el apartado 2.1 del presente documento, se deberá crear una petición de cambio en la herramienta de gestión (EasyVista) del tipo adecuado, y en la misma se documentará el alcance del cambio y la solución técnica a aplicar, de la siguiente forma:

- Si se trata de un cambio "menor", será suficiente con explicar el alcance del cambio en el campo "Justificación" y dar una breve descripción técnica del cambio a realizar en el campo "Nota" de EasyVista. Para describir el alcance del cambio se seguirá el siguiente esquema (sólo se incluirán los campos que sean de aplicación en cada caso):
  - **ORIGEN**: Indicar de quién procede la petición de cambio
  - **APLICAR EN**: Indicar en qué "punto funcional" hay que aplicar el cambio: es decir, en qué módulo, pantalla, botón, proceso, etc. *P. ej.: Censo > Buscar Sanciones > Editar*
  - **ESCENARIO**: Configuraciones previas o precondiciones que se tengan que dar para que se produzca la situación que requiere cambio
  - **INCIDENCIA**: Si el cambio deriva de una incidencia, describir brevemente lo que falla. Este campo no será necesario si la petición de cambio ya está relacionada con una incidencia en EasyVista
  - **MOTIVO:** Si el cambio no deriva de una incidencia, justificar brevemente por qué se solicita
  - **SE PIDE**: Descripción de en qué debe consistir (funcionalmente) el cambio
  - **SUGERENCIA**: El solicitante del cambio puede incluir aquí sugerencias sobre cómo implementarlo
- Si se trata de un cambio de mayor entidad, se deberá realizar un documento de especificación del cambio, que se archivará en la carpeta **DEC** de ***999999 Mantenimiento*** correspondiente al sistema *software* al que afecte, según se ha explicado en el apartado 0. Este documento deberá vincularse mediante un enlace (UNC) desde el campo "Notas" de la petición de cambio en EasyVista. El contenido de este documento será el siguiente (existe una plantilla específica para este tipo de documento, **ITCGAE_PlantillaDEC.dotx**, en la carpeta de plantillas de MEDRA):
  - **Alcance** (descripción de en qué consiste el cambio y a qué módulo(s) afecta)
  - **Requisitos**. Si el cambio es funcionalmente sencillo y queda suficientemente descrito en el apartado de Alcance, se puede omitir este apartado. Para cambios funcionalmente más complejos se recomienda enunciar los requisitos de forma desglosada.
  - **Propuesta de interfaz de usuario** (si el cambio implica una modificación relevante de la IU).
  - **Diseño técnico**. Se especificarán los elementos de código (clases, métodos, páginas, etc.) afectados por el cambio. Si la complejidad técnica del cambio es media o alta, se deberá acompañar de diagramas explicativos (de secuencia, colaboración, etc.). Se especificarán también los cambios del modelo de datos, si los hubiera.
  - **Pruebas**. Se especificarán las pruebas a realizar / realizadas para verificar la correcta implementación del cambio

  El nombre del documento de especificación de cambio se compondrá con el prefijo del sistema *software* al que afecta seguido del acrónimo DEC y del código de la petición de cambio en EasyVista, separados por guion bajo. **Ejemplo: SIGA_DEC_R1509_0086.docx.**

  En cualquier caso, en el campo "Justificación" de EasyVista habrá que indicar, al igual que en los cambios "menores", cuál es el ORIGEN y el MOTIVO del cambio.

Se considera un cambio “menor” aquél que cumpla todos los requisitos siguientes:

- Que conlleve un esfuerzo de desarrollo inferior a 1 jornada * hombre
- Que no implique cambios en el modelo de datos
- Que no implique cambios significativos en la arquitectura interna de la aplicación (modelo de clases, interfaces, etc.)
- Que no tenga un gran impacto en la interfaz de usuario
- Que no implique cambio en interfaces con otros sistemas

Además de documentarse de forma individual el cambio como se ha indicado, si el cambio implica una modificación en requisitos o en diseño del sistema, se deberán actualizar los documentos generales de especificación de requisitos y de diseño técnico del sistema en cuestión, de modo que se mantengan siempre actualizados.

7.3. Metodología de trabajo

Si el cambio se deriva de la corrección de un error, no será necesario validarlo y se gestionará como cambio preautorizado, según se especifica en el procedimiento de gestión del cambio.

En caso contrario, el cambio deberá haber sido solicitado por alguien con autoridad para definir un cambio sobre el sistema en cuestión: un usuario representativo, un comité de expertos, o un responsable de ITCGAE con capacidad de representación de los usuarios. O bien, si el cambio surge a propuesta de la propia área de Desarrollo de ITCGAE, el cambio deberá ser pre-validado por alguien con similares atribuciones en cuanto a definición de cambios. Estos cambios se tramitarán como tipo "ordinario", o bien "urgente" si se considera como tal por su impacto en el sistema de producción.

La solicitud de cambio, en cualquier caso, se registrará en el sistema de gestión (EasyVista).

Para cambios no preautorizados, el jefe de proyecto al que competa el sistema en cuestión dentro del área de Desarrollo deberá estudiar el cambio y decidir si es procedente o no (según establece el procedimiento de gestión de cambios).

SI el cambio no es de tipo "menor" (según la definición anteriormente dada), se desarrollarán en primer lugar los apartados 1 a 3 del documento de especificación del cambio (Alcance, requisitos y propuesta de IU). Estos apartados se deberán validar con el solicitante o responsable del cambio antes de proceder a su implementación.

Una vez validada la especificación del cambio se procederá a su implementación. En el código fuente modificado para implementar el cambio se deberá insertar una línea de comentario indicando el código asignado a la petición de cambio en el sistema de gestión. Si el cambio en el código abarca varias líneas, se insertarán líneas de comentario indicando el inicio y el fin del cambio en el código.

Una vez implementado el cambio se deberá proceder a realizar pruebas funcionales y de regresión sobre el sistema para asegurar:

- que el cambio realizado cumple las especificaciones dadas
- que no se han introducido errores en otras funcionalidades previamente existentes del sistema

Para esto último, se debe acudir a la documentación de diseño del sistema, a efectos de conocer las dependencias que pudieran existir del módulo(s) o clase(s) modificadas, y por tanto, los posibles efectos colaterales derivados de la modificación.

Se deberá evaluar si el cambio realizado podría tener un efecto relevante en el rendimiento de alguna funcionalidad del sistema. En caso afirmativo se deberán realizar pruebas de rendimiento (y en su caso, de carga y estrés) de la funcionalidad en cuestión para asegurarse de que se cumplen los acuerdos de nivel de servicio (SLA), si existen, o si no, unos criterios mínimos de rendimiento razonables.

Por último, finalizadas las pruebas realizadas sobre el sistema modificado, se procederá a entregar el parche o versión menor del sistema siguiendo el procedimiento (**PGS.12 Gestión de Entrega y Despliegue-V*X*.docx**) y la guía (**ITPGS.12-01-Proceso Gestión de Entregas de Software-V*X*.docx**) de gestión de entregas (donde ***X*** representa la última versión liberada de los documentos).

8. Anexo A. Guía rápida

El presente anexo presenta una guía rápida de la metodología MEDRA en forma de cuadros-resumen. Para cada fase de la metodología se describen los documentos (y otros objetos) y, en su caso, los apartados de esos documentos que se consideran **productos mínimos imprescindibles** y que por tanto deberán generarse para los proyectos de desarrollo basados en MEDRA.

8.1. Evaluación del paradigma metodológico

Fase	Productos mínimos imprescindibles	Observaciones
Evaluación del paradigma metodológico	Plan de proyecto, apartado "Evaluación del paradigma metodológico"	En proyectos subcontratados lo idóneo es decidir el paradigma antes de pedir la RFQ, para lo cual puede ser conveniente reunirse previamente con los proveedores candidatos.

8.2. MEDRA^{A} (paradigma ágil)

Fase	Productos mínimos imprescindibles	Observaciones
Actividades de inicio de proyecto	Plan de proyecto, apartados: - Equipo de proyecto - Participantes - Alcance - Arquitectura general del sistema - Planificación y cronograma	En el apartado "participantes" se identificará a las personas que actúen con el rol de "definidor-validador".
	Registro de petición de cambio en EasyVista	Nuevo servicio o modificación relevante
	Catálogo de requisitos	En MEDRA^{A} no es necesario detallar tanto los requisitos como en MEDRA^{C} por seguir el paradigma ágil
	Documento de análisis	Descripción a alto nivel y *mock-ups* de I.U.
	Documento de diseño técnico	Descripción a alto nivel
Cada *sprint*	Plan de proyecto, apartados: - Planificación y cronograma, subapartado específico para el sprint - Revisión de sprint (uno por cada sprint)	

Fase	Productos mínimos imprescindibles	Observaciones
	Registro de tareas en herramienta de gestión	Si el proyecto es subcontratado, será el proveedor quien registre las tareas en su propia herramienta de gestión
	Documento de diseño técnico	Se amplía en detalle el diseño de alto nivel que ya se había plasmado previamente en este documento
	Código fuente y binarios	
	Informe de pruebas	No es necesario un documento muy exhaustivo ni formal, basta con un Excel que recoja las principales pruebas realizadas, especialmente las que han generado error
	Paquete con la entrega	
	Documento de entrega según plantilla normalizada y registro en EasyVista	
Actividades de cierre del proyecto	Informes de pruebas	Los informes de pruebas que se realicen con herramientas automáticas serán los que produzcan las propias herramientas
	Acta de aceptación	
	Manual de usuario	
	Manual de explotación y seguridad	Si procede
	Paquete con la entrega	
	Documento de entrega según plantilla normalizada y registro en EasyVista	

8.3. MEDRA^{C} (paradigma en cascada)

Fase	Productos mínimos imprescindibles	Observaciones
ASI	Plan de proyecto, apartados: - Equipo de proyecto - Participantes - Alcance - Arquitectura general del sistema - Planificación y cronograma	
	Registro de petición de cambio en EasyVista	Nuevo servicio o modificación relevante
	Registro de tareas en herramienta de gestión	Si el proyecto es subcontratado, será el proveedor quien registre las tareas en su propia herramienta de gestión
	Catálogo de requisitos	En MEDRA^{C} los requisitos deben estar más detallados que en MEDRA^{A} porque son la base de lo que se va a construir
	Documento de análisis, apartados: - Especificación de casos de uso - Subsistemas (si aplica) - Especificación de interfaz de usuario	
	Plan de pruebas	
DSI	Registro de tareas en herramienta de gestión	Si el proyecto es subcontratado, será el proveedor quien registre las tareas en su propia herramienta de gestión
	Documento de diseño, apartados: - Arquitectura del sistema - Requisitos de diseño - Catálogo de excepciones - Modelo de datos - Modelo de clases - Realización de casos de uso	En la realización de casos de uso se describe (mediante diagramas de interacción UML) cómo interaccionan los objetos para implementar la funcionalidad requerida
	Plan de migración y carga inicial de datos	Si no se requiere migración, la carga inicial de datos puede especificarse en el documento de diseño y obviarse este plan
CSI	Registro de tareas en herramienta de gestión	Si el proyecto es subcontratado, será el proveedor quien registre las tareas en su propia herramienta de gestión
	Código fuente y binarios	

Fase	Productos mínimos imprescindibles	Observaciones
	Informe de pruebas	No es necesario un documento muy exhaustivo ni formal, basta con un Excel que recoja las principales pruebas realizadas, especialmente las que han generado error
	Manual de usuario	
	Manual de explotación y seguridad	Si procede
IAS	Paquete con la entrega	
	Documento de entrega según plantilla normalizada y registro en EasyVista	
	Informes de pruebas	Los informes de pruebas que se realicen con herramientas automáticas serán los que produzcan las propias herramientas
	Acta de aceptación	

9. Anexo B. Herramientas de apoyo a la metodología

El presente anexo recoge las herramientas a utilizar en distintas actividades definidas por la metodología.

NOTA: Estas serán las herramientas utilizadas cuando las actividades sean desempeñadas directamente por personal de RedAbogacía (interno o externo). En el caso de proyectos subcontratados a proveedores, el proveedor deberá comprometerse a utilizar estas herramientas u otras que proporcionen una funcionalidad equivalente y que permitan producir los entregables requeridos por la metodología.

Actividad	Herramienta	Observaciones
Redacción de documentos (ofimática)	Microsoft Office	En los casos en los que existan, se aplicarán las plantillas definidas, si no existe plantilla específica se usará la genérica de RedAbogacía. Las versiones finales de los documentos se podrán pasar a formato PDF.
Planificación de tareas	Microsoft Project Gantter for Google Drive	Dado que RedAbogacía dispone de un número limitado de licencias de MS Project y son muy costosas, se propone el uso alternativo de la herramienta gratuita Gantter for Google Drive (extensión de Google Chrome)
Especificación de requisitos	Microsoft Word con plantilla específica y macros	A futuro, posible importación en Enterprise Architect
Gestión de tareas de proyectos	JIRA Software	Véase el anexo D de este documento
Diagramación	Enterprise Architect	Diagramas contemplados en Enterprise Architect (UML, modelado web, DFD, modelo entidad/relación, etc.). Sólo para proyectos de desarrollo nuevos de gran envergadura.
	Microsoft Visio Dia	En proyectos en los que no se use EA o para diagramas de otros tipos. Los usuarios que no tengan licencia de Visio pueden utilizar la alternativa *freeware* Dia
Diseño de prototipos de I.U.	Balsamiq Mockups	Por determinar si se utilizará la versión Cloud, de escritorio o el plugin para JIRA
Definición de tests unitarios	JUnit	Para proyectos Java. Si se diera el caso de proyectos en otra tecnología, se utilizará la herramienta más adecuada a ese caso

Actividad	Herramienta	Observaciones
Implementación de código	Eclipse, Subversion	Eclipse como IDE, Subversion como sistema de control de versiones
Depuración javascript/css/html	Firebug	Complemento de Firefox para hacer depuraciones javascript/css/html
Ejecución de sentencias SQL contra la BD	Toad SQLDeveloper DBeaver	Existe una versión de Toad gratuita aunque limita el nº de usuarios. Otra alternativa gratuita es DBeaver.
Construcción y despliegue	Jenkins, Maven, Artifactory	Maven como gestor de dependencias y herramienta de construcción, Artifactory como repositorio de binarios (por confirmar), Jenkins para automatización de tareas
Construcción de mocks o dummies	SOAPUI, Mockito	SOAPUI para servicios web, Mockito para clases Java
Testing de servicios web	SOAPUI, PostMan	SOAPUI para servicios SOAP. PostMan para servicios REST (es una extensión de Chrome)
Testing funcional automatizado	Selenium	
Testing de carga y estrés	JMeter, HP LoadRunner, proveedor externo	Se contempla la posibilidad de externalizar las pruebas de carga a través de un proveedor externo como ALTEN o IP-LABEL. Para pruebas realizadas internamente en RedAbogacía, se puede utilizar JMeter o la versión gratuita de HP LoadRunner

10. Anexo C. Buenas prácticas de programación

El presente anexo recoge un catálogo de buenas prácticas orientadas a la actividad de programación de *software*, que deberán ser observadas y cumplidas por todos los desarrolladores que participen en proyectos de IT CGAE, tanto si se trata de personal interno como subcontratado.

1. **Pensar antes de actuar.** No lanzarse a codificar sin más; tener en cuenta:
    - ¿Sabemos lo que estamos haciendo? ¿Y para qué sirve? Si no, preguntar. El conocimiento funcional sobre lo que estamos desarrollando es fundamental, especialmente para saber probar lo que estamos desarrollando.
    - Simplicidad (principio KISS – *Keep It Simple, Stupid!*). Hacer lo más simple que pueda funcionar. Código más simple implica menos posibilidad de errores y mejor mantenibilidad.
    - Patrones de diseño (*Gang of Four*, etc.). Es importante conocerlos y considerar si los podemos aplicar. Son patrones bien conocidos que pueden aplicarse en muchos casos y evitan que tengamos que "reinventar la rueda" en cuestión de diseño.
    - Modularidad: implementar clases bien diseñadas, siguiendo los principios de orientación a objetos: encapsulación, herencia y polimorfismo. Diseñar siempre métodos pequeños, con interfaces bien definidas.
    - Posibilidades de reutilización. Aprovechar código existente, intentar generalizar el código nuevo, pero sólo si vamos a usar esa generalización; no caer en la tentación de querer generalizar demasiado, y acabar implementando código más complejo para cubrir casos que al final nunca se darán.
    - Fomentar la extensibilidad: usar interfaces en vez de clases para los tipos de datos que se usen como parámetros; uso del patrón de "inyección de dependencias".

2. **Codificar con calidad**:
    - Controlar todos los errores y excepciones, un usuario final no debería recibir nunca un mensaje de excepción no controlada. "Desconfiar" de los parámetros de entrada, comprobar siempre que los valores recibidos están en el rango y formato esperados.
    - Usar nombres de variables significativos y ajustados al uso real que tiene la variable. No reutilizar la misma variable para usos distintos, genera confusión y disminuye la legibilidad del código.
    - Utilizar constantes nombradas, no valores "a pelo".
    - Escribir código claro, legible. Poner comentarios cuando sea preciso; pero si no es preciso, mejor (¡señal de que el código es auto-explicativo!)
    - Utilizar las plantillas de formato (*formatters*) predefinidos en el IDE para IT CGAE para conseguir un formato homogéneo del código de todos los desarrolladores
    - MUCHO OJO con el "copiar y pegar": ¿sabemos lo que estamos haciendo? No utilizar ejemplos sacados de internet si no tenemos muy claro lo que hace el código.
    - Seguir el convenio estándar de Java para el nombrado de paquetes (http://docs.oracle.com/javase/tutorial/java/package/namingpkgs.html). En particular, los nombres de paquetes deberían comenzar por **org.itcgae**; a continuación llevarán el nombre de la aplicación o sistema y a partir de ahí el nombre de subsistema, paquete, componente, etc. *Por ejemplo:* `org.itcgae.lexnetabogacia.utils`.
      Análogamente, los espacios de nombres XML (usados, por ejemplo, en la definición de servicios web) deberán comenzar por **http://www.abogacia.es**. *Por ejemplo:* `http://www.abogacia.es/censo`.
      NUNCA se debe incluir en un nombre de paquete o espacio de nombres el nombre de la empresa proveedora del desarrollo.

3. **Seguridad desde el diseño:**
    - Pensar en la seguridad del sistema desde el momento en que lo estamos diseñando; no dejar los aspectos de seguridad como un refinamiento posterior. Al final es más costoso en esfuerzo y se genera código menos seguro.
    - Controlar los riesgos usuales de seguridad:
        - Como regla general, desconfiar de los datos que procedan directamente de la entrada del usuario
        - Control de *buffers* y *arrays*: asegurarse de que no se puede escribir en la memoria excediendo su tamaño
        - Utilizar datos bien "tipados"
        - Prevenir la inyección de SQL: NUNCA concatenar directamente en una sentencia SQL la entrada que proceda del usuario (por ejemplo, el contenido de un campo de edición). Utilizar mejor parámetros para construir las sentencias SQL.
        - Prevenir la inyección de código de *script* en páginas web: codificar en HTML el contenido de las variables de tipo cadena que vayan a ser volcadas a la página para evitar la inyección del *tag* `<script>`

4. **Probar en paralelo a la codificación:**
    - Probar método por método. Como deben ser pequeños y simples, es menos tedioso ir probando cada método a medida que se va codificando.
    - Probar con "mala intención". No probar sólo el caso "esperado"; hay que probar los casos límite (pasar parámetros a *null*, fechas imposibles, números negativos cuando se esperan positivos…)
    - Utilizar *frameworks* de pruebas unitarias (p. ej. *JUnit*) para automatizar pruebas
    - Probar la misma funcionalidad con distintos perfiles de usuario (de distinto nivel de privilegios, de distinto departamento / Colegio, etc.)

4. **Pruebas cruzadas:**
    - Cuando el código está acabado, el desarrollador A prueba el código de B y viceversa
    - Conviene que el código lo pruebe:
        - Alguien que sabe bien lo que debe hacer el código: para probar exhaustivamente todos los casos funcionales
        - Alguien que no lo sabe bien: para ver si un "mal uso" es capaz de "romper" el código

11. Anexo D. Uso de JIRA

El presente anexo recoge las indicaciones sobre cómo trabajar con la herramienta JIRA para la gestión de tareas del área de Desarrollo de RedAbogacía.

NOTA: A LA FECHA DE ESTA VERSIÓN DEL DOCUMENTO, ESTE ANEXO SE ENCUENTRA AÚN EN REVISIÓN Y PUEDE SER SUSCEPTIBLE DE CAMBIOS.

- Cada miembro del área de Desarrollo de ITCGAE contará con una cuenta en el sistema JIRA del área
- Existirá un grupo de usuarios en JIRA por cada uno de los grupos en que se divide funcionalmente el área de Desarrollo de RedAbogacía. Un usuario deberá pertenecer al menos a un grupo, y podrá pertenecer a varios. Cada grupo tendrá visibilidad sobre un conjunto de proyectos.
- Cada usuario podrá tener asignado uno o varios roles en uno o varios proyectos. Los roles definidos son: "Administrador", "*Product owner*" y "Desarrollador".
    - Los miembros con perfil de desarrollador tendrán asignadas sus tareas en la herramienta, y la utilizarán para informar el grado de avance de cada una de ellas
    - Los miembros con perfil de *Product owner* podrán asignar tareas a los miembros del sus equipos y realizar el seguimiento de las tareas sobre la herramienta
    - Los miembros con perfil de administrador podrán realizar algunas acciones de administración sobre los proyectos que administren
    - El perfil se asigna a cada usuario por cada proyecto, con lo que un mismo usuario puede tener distintos perfiles en distintos proyectos.
    - Adicionalmente, habrá un grupo reducido de miembros del área que tendrán rol de Administrador de JIRA (pertenecerán a los grupos "administrators" y "*jira-administrators*")
- Para proyectos en los que participa una empresa externa proveedora, se contemplan dos posibles escenarios:
    - Si la empresa dispone de su propio sistema de seguimiento de tareas y ésta está accesible a través de internet, el jefe de proyecto (y potencialmente, los analistas) de ITCGAE podrían disponer de una cuenta de usuario en dicho sistema y realizar el seguimiento sobre él
    - Otra opción es asignar cuentas de usuario en el JIRA departamental de Desarrollo de ITCGAE a los miembros del equipo de trabajo de la empresa proveedora y que gestionen sus tareas en este sistema. En tal caso deberá garantizarse un control de permisos adecuado para que los miembros de ese equipo no puedan acceder a información de otros proyectos (a través de una adecuada gestión de grupos)
- Se crearán tres categorías de proyectos:
    - Mantenimiento – Para sistemas en "mantenimiento continuo"
    - Desarrollo ágil – Para proyectos de desarrollo que siguen el paradigma ágil (MEDRA^{A})
    - Desarrollo en cascada – Para proyectos de desarrollo que siguen el paradigma en cascada (MEDRA^{C})
- Por cada proyecto de desarrollo que se vaya a ejecutar o por cada sistema software en situación de mantenimiento, se creará un elemento de tipo "proyecto" en JIRA, dentro de la categoría correspondiente. Al crear el proyecto se escogerá el tipo adecuado según la categoría a la que pertenezca:
    - Para proyectos de mantenimiento, tipo "*Desarrollo de software de Kanban*"
    - Para proyectos ágiles, tipo "*Desarrollo de software de Scrum*"
    - Para proyectos en cascada, tipo "*Desarrollo básico de software*"

- El responsable de crearlo será el Jefe de proyecto correspondiente. El nombre coincidirá con el del sistema que se está desarrollando o manteniendo.
    - Si se trata de un proyecto de desarrollo evolutivo de un sistema ya existente en producción, el proyecto se creará en JIRA como una nueva versión del proyecto anterior, utilizando la capacidad de JIRA para gestionar distintas versiones de un proyecto.
- Se crearán diferentes "esquemas de tipos de incidencias" en JIRA:
    - Uno para los sistemas en mantenimiento y para proyectos de desarrollo con paradigma ágil (esquema tipo SCRUM)
    - Otro para proyectos de desarrollo con paradigma en cascada

  A cada proyecto JIRA se le asociará el esquema que corresponda, según su categoría.
- Se definirán los siguientes "tipos de incidencias" en JIRA:
    - Error
    - Épica
    - Historia
    - Tarea
    - Subtarea
    - Análisis
    - Diseño
    - Programación
    - Pruebas integración
    - Entrega
    - Documentación

  Los tipos de incidencia se asociarán a los esquemas de tipos de incidencias según la tabla siguiente:

Tipo de incidencias	Esquema tipo SCRUM	Esquema en cascada
Error	X	X
Épica	X	
Historia	X	
Tarea	X	
Subtarea	X	
Análisis		X
Diseño		X
Programación		X
Pruebas integración		X
Entrega	X	X
Documentación	X	X

- Se definirán los siguientes "estados de peticiones" en JIRA:
    - Backlog
    - Seleccionada para desarrollo
    - En curso

- Listo
- En pruebas
- Reabierta
- Cerrada
- Rechazada

Las peticiones podrán ser cambiadas de estado por los perfiles de “Jefe de proyecto” (JP) “Desarrollador” (D) o por el equipo de QA (cuando aplique) según el siguiente diagrama de transición de estados: