

Universidad Carlos III

Curso Ingeniería del Software 2020-21

Práctica   
Curso 2020-21

Título

Fecha: **xx/xx/xx** - ENTREGA: **1/2/FINAL**

GRUPO: **81/82/83** EQUIPO: **01/02…**

Alumnos: **Nombre apellido / Nombre Apellido**

**Nombre apellido / Nombre apellido**

PÁGINA DE ESTADO DE DOCUMENTO

Registro de Cambios

|  |  |  |  |
| --- | --- | --- | --- |
| Versión | Fecha | Autor | Cambios en la versión |
| 0 | 30-09-2011 | Ana García | Primer borrador. |
| 1 | 15-10-2011 | Juan Gómez | Presentado como primera entrega. Se completan las secciones del capítulo 1 que habían quedado en blanco en el primer borrador. |
| 1.1 | 20-10-2011 | Ana García | Corregido según observaciones de los revisores. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Horas invertidas en el proyecto

|  |  |  |  |
| --- | --- | --- | --- |
| NOMBRE | INDIVIDUAL | EQUIPO | TOTAL |
| Ana García | 40 | 15 | **55** |
| Juan Gómez | 43 | 11 | **54** |
| Isabel López | 47 | 16 | **63** |
| Pedro Fernández | 50 | 18 | **68** |
| TOTAL | **180** | **60** | **240** |

Distribución de responsabilidades

|  |  |
| --- | --- |
| NOMBRE | Descripción de sus principales contribuciones al proyecto |
| Ana García | Ha trabajado principalmente en las secciones 1 y 2, ha revisado el resto de secciones, coordinación. |
| Juan Gómez | Requisitos funcionales y modelo conceptual, revisión del proyecto del otro equipo |
| Isabel López | Requisitos no funcionales y modelo de implementación, listas y tablas |
| Pedro Fernández | Especificación del diseño de componentes, revisión del proyecto del otro equipo, listas y tablas |

Si se desea, la tabla puede detallarse más usando una matriz de asignación de responsabilidades (matriz RACI: *responsible, accountable, consulted, informed*).

TABLA DE CONTENIDO

1 Introducción 6

1.1 Propósito del documento 6

1.2 Visión general del documento 6

1.3 Referencias 6

2 Descripción general 7

2.1 Perspectiva del producto 7

2.2 Alcance del software 7

2.3 Capacidades generales 7

2.4 Restricciones generales 7

2.5 Características de los usuarios: roles y capacidades 8

2.6 Entorno operacional 8

3 Requisitos 9

3.1 Justificación de la clasificación de requisitos 9

3.2 Justificación de la plantilla de requisitos 9

3.3 Requisitos funcionales 9

3.4 Requisitos no funcionales 10

3.5 Vocabulario del dominio 10

4 Arquitectura 11

4.1 Modelo conceptual 11

4.2 Modelo de implementación 12

4.3 Especificación del diseño de componentes 12

5 Listas y tablas 15

5.1 Consistencia entre requisitos: conflictos, redundancias, acoplamientos 15

5.2 Trazabilidad requisitos – modelo conceptual (clases) 15

5.3 Trazabilidad requisitos – modelo de implementación (componentes) 15

CONSIDERACIONES SOBRE LAS ENTREGAS

* + La **primera entrega** abarca los capítulos 1-2-3 y una primera versión de la sección 5.1.
  + La **segunda entrega** se completa con los capítulos 4-5. La sección 5.1 se corrige y amplía según sea necesario.
  + La **entrega final** revisa y corrige todo el contenido de acuerdo con las observaciones recibidas. La primera entrega revisada puede adelantarse si se desea con la segunda entrega.

Eliminar esta página del trabajo final

# Introducción

Este capítulo debe dar una visión general del conjunto del documento.

## Propósito del documento

Esta sección debe:

1. Definir el propósito del documento en particular (el documento, *no el proyecto*).
2. Especificar el perfil del lector al que va dirigido el documento.

## Visión general del documento

Esta sección debe:

1. Describir lo que contiene el resto del documento.
2. Explicar cómo está organizado el documento.

## Referencias

Esta sección debe incluir una lista completa de *todos los documentos citados* en la práctica (estándares, manuales, páginas web, herramientas utilizadas, etc.), identificados por su autor, título y fecha, u otros datos análogos que permitan su localización unívoca (título de la revista u organización que lo ha publicado, URL y fecha de descarga, etc.).

# Descripción general

Este capítulo describe el contexto en el que debe encuadrarse el producto final. No contiene los requisitos específicos del sistema como tales, pero sí información que los hace más fáciles de comprender y les sirve de introducción.

## Perspectiva del producto

Esta sección pone el producto final en perspectiva frente a otros productos relacionados con él. Explica el *contexto evolutivo* del producto: si el sistema sustituye a otro anterior, o es evolución de un antepasado, o está previsto que en el futuro será a su vez sustituido por uno nuevo, etc. Todos los productos mencionados deben ser adecuadamente referenciados.

Además, en esta sección pueden incluirse estudios realizados sobre otros *productos similares* al que queremos desarrollar. Realizar estudios sobre sistemas similares al que se va a desarrollar es una posibilidad interesante, en algunos casos incluso aconsejable. No obstante, si el estudio es muy extenso, deberá quedar en un anexo o en un documento separado.

## Alcance del software

Esta sección debe:

1. Identificar por su nombre el o los productos software que serán generados.
2. Explicar el objetivo general que se persigue con la aplicación, lo que deberá hacer el futuro software (y, si es necesario, también *lo que no deberá hacer*).
3. Describir los beneficios, objetivos y metas del proyecto de forma tan precisa como sea posible, pero sin llegar al nivel de concreción propio de los requisitos (ver capítulo 3).
4. Consideraciones éticas: valoración ética de los medios, fines y beneficios del proyecto (aproximadamente un párrafo donde se examine la responsabilidad social del proyecto).

## Capacidades generales

Esta sección debe incluir una descripción de los principales *servicios que deberá ofrecer el sistema* y por qué son necesarios. También deberá describir el *proceso al que el software dará soporte*: la descripción de este proceso justifica y hace comprensibles los requisitos que más tarde se especificarán de modo completo, y permite realizar un análisis crítico de los mismos. Nótese bien la diferencia entre el sistema software como tal, y el proceso u organización al que este sistema da servicio (véase 2.6 Entorno operacional).

## Restricciones generales

Esta sección debe describir cualquier circunstancia que limite las opciones del desarrollador para construir el sistema. Esta sección no debe usarse para imponer requisitos específicos o restricciones de diseño específicas, pero deberá explicar por qué determinados requisitos o restricciones existen.

## Características de los usuarios: roles y capacidades

Esta sección deberá describir aquellas características de los usuarios que afecten a los requisitos. Una tabla con los roles de usuarios y los servicios o funciones que requiere cada rol (= capacidades) es muy útil para lograr una concepción global del sistema. Es análogo a un modelo de casos de uso de alto nivel, pero sin necesidad del diagrama, ya que la representación en forma de tabla es suficiente.

Es posible que mucha gente interactúe con el sistema en la fase de operación y mantenimiento. Algunas de esas personas son usuarios, operadores y personal de mantenimiento. Determinadas características de ese personal como su nivel educativo, idioma, experiencia y conocimiento técnico impondrán importantes restricciones al software.

Aunque el software sea usado con mucha frecuencia, puede que determinadas personas lo usen sólo ocasionalmente. Los usuarios habituales se convertirán en expertos mientras que los ocasionales seguirán siendo hasta cierto punto novatos. Es importante clasificar a los usuarios y estimar la cantidad más probable de usuarios de cada categoría. Si no es posible hacer una estimación en números absolutos, una estimación en números relativos también será útil.

Conviene decir algo sobre el tipo general de interfaz que requieren los usuarios: si es una interfaz clásica de ordenador o más bien se requiere un terminal específico, si debe ser fijo o móvil, u otras características especiales (acceso por voz, pantalla táctil, etc.). No se trata de detallar completamente la interfaz (esto es una tarea posterior de diseño), sino de dar algunas ideas generales que marcan el tipo de software que hay que desarrollar.

## Entorno operacional

Esta sección describe el universo artificial y natural en el que el sistema software va a funcionar, es decir, los otros sistemas informáticos, dispositivos mecánicos y agentes naturales con los que deberá interactuar el sistema informático. Esta descripción narrativa puede ir acompañada de *diagramas de contexto*, para resumir interfaces externas, y *diagramas de actividad* para ilustrar cómo la actividad externa se integra con el sistema informático. También deberá especificarse la naturaleza de las interacciones con sistemas externos.

Si el documento define un producto que forma parte de un proyecto o sistema software mayor, entonces esta sección debe también describir brevemente las actividades que serán soportadas por los sistemas externos y definir las interfaces de comunicación con esos sistemas.

# Requisitos

Este capítulo debe contener la especificación completa de los requisitos del sistema software que se desea desarrollar. Los clientes deben validar el conjunto de requisitos, y el control de calidad debe verificar que el sistema software satisface estos mismos requisitos. Por este motivo los requisitos deben expresarse de forma clara, precisa, completa, correcta, consistente y verificable.

Es *esencial* elaborar el vocabulario del dominio, de modo que en los requisitos se utilice en todo momento un vocabulario controlado. Los requisitos y vocabulario del dominio se elaboran en paralelo.

## Justificación de la clasificación de requisitos

En esta sección se deben explicar las razones que se han seguido para elegir la clasificación de requisitos adoptada. Los requisitos deben estar adecuadamente ordenados y estructurados en apartados y subapartados, desarrollando la clasificación elemental en requisitos funcionales (por ejemplo en áreas temáticas) y no funcionales (por ejemplo por tipos de requisitos no funcionales). Una lista plana de requisitos habitualmente no es suficientemente aclaratoria.

Por tanto, en esta práctica deben añadirse *obligatoriamente* más subdivisiones de acuerdo con una clasificación bien estructurada de los requisitos, siguiendo los principios de modularidad y anidamiento, y debe explicarse la clasificación adoptada. Específicamente para los requisitos funcionales, se debe proponer una clasificación de los requisitos en *áreas temáticas* (es decir, agrupar los requisitos según el “tema” al que se refieren), tomando como pauta el modelo conceptual (ver 4.1), que se encontrará ya en fase inicial de desarrollo. Para los requisitos no funcionales puede ser útil una clasificación a priorien *tipos de requisitos* (por ejemplo la propuesta por el estándar ESA, o la clasificación propuesta en la asignatura), pero esta clasificación debe ser flexible y adaptada al problema concreto.

## Justificación de la plantilla de requisitos

Es muy conveniente utilizar una *plantilla* (en forma de tabla) para cada requisito individual, especificando por separado los distintos atributos de cada requisito. Debe justificarse el conjunto de atributos que se ha elegido para configurar esta plantilla.

El *plan de pruebas de verificación* es un elemento obligatorio de esta práctica. Las pruebas de verificación deben especificarse para cada requisito, como un apartado propio dentro de la plantilla.

## Requisitos funcionales

Esta sección debe contener los requisitos funcionales. Los requisitos deben ser redactados en lenguaje natural. Esto los hace más comprensibles para los no especialistas, aunque abre la puerta a que surjan inconsistencias, ambigüedades e imprecisiones. Estas propiedades no deseables pueden ser evitadas con el uso de un lenguaje técnico adecuado para la especificación de requisitos.

## Requisitos no funcionales

Esta sección debe contener los requisitos no funcionales, igualmente redactados en lenguaje natural. Pueden afectar a todo el conjunto de requisitos funcionales, o bien sólo a una parte de los mismos, en cuyo caso debe especificarse de qué parte se trata (por ejemplo, requisitos de seguridad relacionados con determinado grupo de funcionalidades del software).

## Vocabulario del dominio

El vocabulario del dominio (también denominado a veces “glosario de términos”) se subdivide en dos apartados, que es muy importante distinguir bien.

### Vocabulario del modelo conceptual

El primero de ellos (*vocabulario del modelo conceptual*) define todos los términos significativos y específicos del problema que aparecen en los requisitos. Los términos pueden ser simples o compuestos, nominales o verbales (ejemplos: “cuenta”, “estación de control”, “facturar”, “reasignar zona”). Su elaboración comienza con la captura de requisitos, y evoluciona con el análisis de requisitos y el modelado conceptual. Es una herramienta muy útil para elaborar los requisitos y el modelo conceptual, ya que éste debe recoger el vocabulario relevante utilizado en los requisitos. El vocabulario del dominio establece un puente entre los requisitos y los modelos, y es además esencial para utilizar en todo momento un vocabulario controlado.

### Vocabulario técnico

El segundo apartado (*vocabulario técnico*) incluye otros términos (típicamente referidos al diseño o a las tecnologías empleadas en el sistema) que se ha juzgado conveniente definir, relativos a artefactos de implementación que no tienen relación directa con el modelo conceptual, aunque aparezcan en los requisitos. No es necesario definir todos los términos técnicos que ya se conocen por el contexto de la asignatura (por ejemplo, ‘interfaz’), sino sólo aquellos que son específicos del proyecto (por ejemplo, un sistema especial de transmisión de información). También se deben definir aquí los acrónimos y abreviaturas empleados en el documento.

A diferencia del vocabulario del modelo conceptual, los términos del vocabulario técnico no aparecen en el modelo conceptual.

# Arquitectura

## Modelo conceptual

Esta sección debe contener un modelo cuyo propósito es especificar la *arquitectura de información* del sistema que se desea construir, mediante un conjunto de diagramas de clases adecuadamente explicados; se pueden completar con diagramas de objetos si se juzga pertinente.

El modelo de información, o modelo conceptual, debe estar adecuadamente *justificado* a partir de los requisitos. No tiene sentido que en él aparezcan clases, atributos, operaciones y otros elementos que no hayan aparecido anteriormente en los requisitos. Igualmente, no tiene sentido que en los requisitos se mencionen conceptos importantes que no aparezcan reflejados de ninguna manera en el modelo conceptual. La referencia al *vocabulario del dominio* (ver sección 3.4) es nuevamente un puente importante que vincula los requisitos con el modelo conceptual.

Es importante notar que la mera yuxtaposición de imágenes (diagramas) resulta muy poco explicativa e insuficiente; por otra parte, si el modelo se intentase explicar sólo mediante palabras, la explicación resultaría compleja y difícil de seguir. En otras palabras, hay que encontrar la combinación adecuada de diagramas y texto que se explican y complementan mutuamente. Analogía: imagina que quieres contar una excursión con fotografías; no te conformarías con poner una tras otra las fotografías, sino que cada una de ellas requiere una explicación textual, que la pone en relación con otras fotografías, llama la atención sobre sus aspectos más relevantes, etc.

Los diagramas de clases deben ayudar a entender el sistema. En consecuencia a estos diagramas no se les debe añadir tanta información que sean difíciles de comprender, ni tan poca información que no añadan nada a lo que ya se dice en los requisitos. Estos diagramas deben tener el grado de detalle suficiente para conservar su carácter comunicativo, en tanto que constituyen una vista gráfica de los requisitos. Por tanto, deberán contener todas las clases conceptuales (es decir, las que aparecen mencionadas en los requisitos), pero no las que sean exigencia exclusivamente del diseño posterior. Así mismo, estas clases aparecerán con los atributos que sean relevantes desde el punto de vista conceptual. Para aumentar la claridad y potencia expresiva, puede ser conveniente mostrar un diagrama global en el que sólo aparezcan las clases, y varios diagramas parciales en los que aparezcan clases con atributos.

Con el fin de resaltar que este modelo representa la arquitectura de información del sistema, y no el diseño de la implementación, en esta práctica *queda prohibido incluir operaciones en las clases*. En otros contextos es perfectamente legítimo y útil representar las operaciones de las clases, pero cuando el objetivo es representar la arquitectura de información, las operaciones resultan contraproducentes.

Debe mencionarse la *herramienta de modelado* utilizada para crear los diagramas (ver 1.3 Referencias).

## Modelo de implementación

En esta sección se presenta y justifica la *arquitectura de desarrollo* elegida (descomposición del sistema en subsistemas y componentes), y se especifican las dependencias entre los distintos componentes que hayan resultado de la descomposición. La descomposición se hace habitualmente a varios niveles (componentes que a su vez se descomponen en subcomponentes, etc.), pero en esta práctica sólo se trata la descomposición de primer nivel, centrada en la vista externa (interfaces) de los componentes, y las relaciones entre ellos.

Se deben utilizar los diagramas y vistas que sean necesarios y las oportunas explicaciones textuales. Vale lo dicho en la sección 4.1 sobre la adecuada combinación de imágenes y palabras

La descomposición arquitectónica elegida debe *justificarse* teniendo en cuenta los criterios vistos en clase: simplicidad, extensibilidad, modificabilidad, eficiencia, y tal vez otros que se consideren relevantes. También deben considerarse los requisitos no funcionales a la hora de justificar la arquitectura. La clasificación de requisitos por áreas temáticas (ver 3.1) puede también proporcionar pistas importantes para esta descomposición. Si existen varias opciones arquitectónicas, entonces hay que compararlas y escoger de modo razonado la que mejor se adapte a los requisitos del sistema.

No hay obligación de seguir una arquitectura predefinida, ni hay una descomposición que *a priori* sea mejor que las demás (por ejemplo MVC, modelo-vista-controlador), sino que la decisión depende del contexto. Lo que hay que hacer es justificar la elección de la arquitectura, razonando por qué es adecuada para resolver el problema planteado. La arquitectura empleada puede ser un estilo arquitectónico estándar, pero también puede ser una variante o un híbrido de estilos, o puede estar sólo vagamente inspirada en uno de ellos. Por lo mismo, tampoco basta que la arquitectura elegida se corresponda con un estilo estándar: lo que hay que hacer es justificar que es adecuada para resolver el problema.

Como en el caso anterior, debe mencionarse la *herramienta de modelado* utilizada para crear los diagramas (ver 1.3 Referencias).

## Especificación del diseño de componentes

En esta sección se especifican las propiedades de cada uno de los componentes identificados en la sección anterior. Cada componente deberá tener un nombre identificador único. El identificador deberá reflejar el propósito y la funcionalidad del componente, siendo a la vez breve y significativo. Si es necesario utilizar abreviaturas, deben utilizarse coherentemente y sin ambigüedad, y deben quedar documentadas. Los componentes deberán tener identificadores mutuamente consistentes: por ejemplo, si un componente se llama Publicador\_Registros, entonces es posible que exista uno que se llame Editor\_Registros, pero no Edición\_Registros, ni Editor-Registros, ni EditorRegistros.

El elemento esencial de la especificación de cada componente son sus interfaces, que obviamente deben coincidir perfectamente con las interfaces identificadas en el modelo de implementación (sección 4.2).

### Componente A (repetir para cada componente)

Por cada componente se repite la misma estructura con los siguientes apartados:

Tipo

Debe indicarse esencialmente si es ejecutable o no ejecutable. Componentes no ejecutables serían archivos de configuración, plantillas de páginas web, etc. El contenido de los demás apartados de la descripción del componente depende del tipo de componente: si es ejecutable, habrá que hacer énfasis en el Procesamiento (por ejemplo, métodos ofrecidos por las clases de implementación); si no es ejecutable, en los Datos (por ejemplo, información contenida en una página web). Los apartados de la descripción no son fijos, sino que deben adaptarse al tipo de componente.

Propósito [Función, Procesamiento, Datos, Recursos]

El propósito del componente se define esencialmente trazándolo contra los requisitos que el componente implementa. Esta trazabilidad hacia atrás justifica la existencia del componente y explica su propósito. Este apartado contiene una breve descripción textual del propósito del componente, basada en esos requisitos. El detalle de la trazabilidad se especifica en la sección 5.3.

Dependencias (interfaces requeridas)

Se definen las dependencias respecto a interfaces proporcionadas por otros componentes del sistema, o por sistemas externos. En el primer caso, dado que toda *interfaz requerida* tiene su contrapartida en una *interfaz proporcionada*, no es necesario repetir la especificación de la interfaz en ambos lugares; basta con especificarla como interfaz proporcionada y referenciarla adecuadamente en el otro lugar donde es requerida. En el segundo caso, sí es necesario especificar completamente la interfaz requerida, puesto que la interfaz proporcionada queda fuera del sistema.

Interfaces (Interfaces proporcionadas)

De modo general una interfaz es un conjunto de operaciones que ofrecen un servicio coherente. En diseño orientado a objetos se va más allá: una interfaz *define un tipo*, que proporciona un conjunto coherente de operaciones sobre las instancias compatibles con ese tipo.

La definición de interfaces permite aislar componentes y lograr un diseño modular y mantenible mediante interacciones bien definidas. Para definir correctamente el tipo especificado por la interfaz, se enumeran las operaciones pertenecientes a la interfaz, así como los contratos que deben satisfacer cada una de las operaciones. Para especificar los contratos debe utilizarse la técnica de *diseño por contratos* vista en el curso.

# Listas y tablas

## Consistencia entre requisitos: conflictos, redundancias, acoplamientos

## Trazabilidad requisitos – modelo conceptual (clases)

## Trazabilidad requisitos – modelo de implementación (componentes)

En estas tres secciones pueden utilizarse tablas de doble entrada o tablas de 2-3 columnas. La tabla de doble entrada tiene el peligro de ser excesivamente dispersa, por lo que a menudo las tablas simplificadas de pocas columnas son una solución más conveniente.