UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

Especialidad cursada

TRABAJO FIN DE GRADO

Plantilla guía de TFG para la ESI-UCLM
Curso de LaTeX esencial

Jesús Salido Tercero

septiembre, 2021

UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA

Depto. del Tutor
(cont.)

Especialidad cursada

TRABAJO FIN DE GRADO

Plantilla guía de TFG para la ESI-UCLM
Curso de LaTeX esencial

Autor: Jesús Salido Tercero

Tutor(a): nombre y apellidos

Co-tutor(a): nombre y apellidos

septiembre, 2021

Plantilla guía de TFG

Este texto ha sido preparado con la plantilla LaTeX de TFG para la UCLM publicada por Jesús Salido en GitHub[1] y Overleaf [2] como parte del curso *«LaTeX esencial para preparación de TFG, Tesis y otros documentos académicos»* impartido en la Escuela Superior de Informática de la Universidad de Castilla-La Mancha.

[1] https://github.com/JesusSalido/TFG_ESI_UCLM, DOI: 10.5281/zenodo.4574562
[2] https://www.overleaf.com/latex/templates/plantilla-de-tfg-escuela-superior-de-informatica-uclm/phjgscmfqtsw

Tribunal:

Presidente: _____

Vocal: _____

Secretario(a): _____

Fecha de defensa: _____

Calificación: _____

Presidente	Vocal	Secretario(a)
Fdo.:	Fdo.:	Fdo.:

A mis estudiantes
Por contribuir a hacer de cada día un reto ilusionante

TRABAJO FIN DE GRADO - ESCUELA SUP. DE INFORMÁTICA (UCLM)

Plantilla guía de TFG

Jesús Salido Tercero
Ciudad Real, septiembre 2021

Resumen

En una página como máximo, el resumen explicará de modo breve la problemática que trata de resolver el trabajo *(el 'qué')*, la metodología para abordar su solución *(el 'cómo')* y los resultados obtenidos. En los trabajos cuyo idioma principal sea el inglés, el orden de Resumen y Abstract se invertirá.

En concreto este documento debe servir como guía para preparar, con LaTeX, el TFG en la Escuela Superior de Informática (ESI) de la Univ. de Castilla-La Mancha (UCLM) siguiendo la normativa de aplicación. Está disponible en GitHub y Overleaf. Por tanto, puede emplearse tanto en modo local en un equipo con LaTeX instalado (MiKTeX, TeX Live, etc.), o bien en línea empleando el servicio de edición Overleaf.

Este documento se aprovecha para proporcionar información sobre la elaboración de la memoria del TFG con ayuda de LaTeX empleando este documento como plantilla. Por este motivo, el documento sigue una estructura similar de secciones a la que debe presentar un TFG y muestras ejemplos de uso de distintos elementos y comandos de maquetación del documento.

Aunque la plantilla se ajusta a las necesidades y reglamentación de la ESI-UCLM, su adaptación es sencilla a otras titulaciones, instituciones y otros documentos de carácter académico. Esta plantilla permite de modo automático la elaboración de la memoria del documento en idioma inglés y puede ser utilizada en cualquier SO (Windows, Linux, Mac OSX, etc.).

Guided template for TFG

Jesús Salido Tercero
Ciudad Real, September 2021

Abstract

English version for the abstract.

Agradecimientos

Aunque es un apartado opcional, haremos bueno el refrán *«es de bien nacidos, ser agradecidos»* si empleamos este espacio como un medio para agradecer a todos los que, de un modo u otro, han hecho posible que el trabajo realizado *llegue a buen puerto*. Esta sección es ideal para agradecer a directores, profesores, mentores, familiares, compañeros, amigos, etc.

Estos agradecimientos pueden ser tan personales como se desee e incluir anécdotas y chascarrillos, pero *nunca deberían ocupar más de una página.*

Jesús Salido Tercero
Ciudad Real, 2021

Notación y acrónimos

NOTACION

Ejemplo de lista con notación (o nomenclatura) empleada en la memoria del TFG.[3]

A, B, C, D	:	Variables lógicas
f, g, h	:	Funciones lógicas
$\cdot$	:	Producto lógico (AND), a menudo se omitirá como en AB en lugar de $A \cdot B$
$+$	:	Suma aritmética o lógica (OR) dependiendo del contexto
$\oplus$	:	OR exclusivo (XOR)
$\overline{A}$ o A'	:	Operador NOT o negación

LISTA DE ACRÓNIMOS

Ejemplo de lista con los acrónimos empleados en el texto.

CASE	:	Computer-Aided Software Engineering
CTAN	:	Comprenhensive TEX Archive network
IDE	:	Integrated Development Environment
ECTS	:	European Credit Transfer and Accumulation System
OOD	:	Object-Oriented Design
PhD	:	Philosophiae Doctor
RAD	:	Rapid Application Development
SDLC	:	Software Development Life Cycle
SSADM	:	Structured Systems Analysis & Design Method
TFE	:	Trabajo Fin de Estudios
TFG	:	Trabajo Fin de Grado
TFM	:	Trabajo Fin de Máster
UML	:	Unified Modeling Language

[3] Se incluye únicamente con propósito de ilustración, ya que el documento no emplea la notación aquí mostrada.

Índice general

Índice de figuras

Índice de tablas

Índice de listados

Índice de algoritmos

CAPÍTULO 1
Introducción

Este capítulo aborda la motivación del trabajo. Se trata de señalar la necesidad que lo origina, su actualidad y pertinencia. Puede incluir también un estado de la cuestión (o estado del arte) en la que se revisen estudios o desarrollos previos y en qué medida sirven de base al trabajo que se presenta.

En este capítulo debería introducirse el *contexto disciplinar y tecnológico* en el que se desarrolla el trabajo de modo que pueda entenderse con facilidad el ámbito y alcance del TFG. Puesto que un TFG no tiene que ser necesariamente un trabajo con aportes novedosos u originales, solo es necesario la inclusión de *estado del arte* cuando este contribuya a aclarar aspectos clave del TFG o se desee justificar la originalidad del trabajo realizado.

Este capítulo suele finalizar indicando la estructura (capítulos) del documento y el contenido de cada una de las partes en que se divide. Por tanto, las secciones que suelen acompañar este capítulo son:

1. **Motivación**. Responde a la pregunta sobre la necesidad o pertinencia del trabajo.
2. **Objetivo**. Determina de modo claro el propósito del trabajo descrito que puede desglosarse en sub objetivos cuando el objetivo principal se puede descomponer en módulos o componentes. Es muy importante definir el objetivo de modo apropiado. El Capítulo 2 de esta guía explica cómo definir el objetivo.
3. **Contexto disciplinar** (o tecnológico). También puede denominarse *estado del arte* cuando se trata de comentar trabajos relacionados que han abordado la cuestión u objetivo que se plantea.
4. **Estructura del documento**.

1.1. REDACCIÓN DE LA MEMORIA

Durante la realización de la memoria del TFG es importante tener presente respetar la guía de estilo de la institución [3]. Por tanto, el empleo de plantillas para un sistema de procesamiento de textos (p.ej., Word o LaTeX) puede requerir su adaptación cuando la plantilla mencionada no haya sido suministrada en la institución a la que se dirige el trabajo.

Para redactar un trabajo académico de modo efectivo se deben tener presentes una serie de normas que ayuden a conseguir un resultado final que sea claro y de fácil lectura. Para obtener un documento con estas características se recomienda seguir una serie de pautas como las expuestas en el blog de Leonor Zozaya [16] o el apartado de comunicación eficaz del departamento de Legua y Estilo de la UOC [14], de lectura recomendada.

A la hora de redactar el texto se debe poner especial atención en no cometer plagio y respetar los derechos de propiedad intelectual [12]. En particular merece gran atención la inclusión de gráficos e imágenes procedentes de Internet que no sean de elaboración propia. En este sentido se recomienda consultar el manual de la Universidad de Cantabria en el que se explica de modo conciso cómo incluir imágenes en un trabajo académico [13].

A continuación en esta plantilla se muestran ejemplos en una memoria preparada con LaTeX. Todos estos ejemplos que se muestran en esta plantilla se han presentado en el curso de LaTeX [11] y tanto estos, como los recogidos en varias obras de referencia, se pueden emplear para adaptar este documento a las necesidades particulares de los estudiantes [7, 5, 2, 8, 6, 4, 15]. Entre las obras de consulta disponibles sobre LaTeX se recomiendan el uso de las obras gratuitas en español [9, 1] y la documentación disponible en la página web de Overleaf [10] (en inglés).

1.1.1. Organización de información

El contenido del trabajo final de estudios se organiza en capítulos que se subdividen en secciones. Con LaTeX este tipo de organización se realiza de modo inmediato y automáticamente se genera los estilos para los títulos de cada sección y su inclusión en la tabla de contenidos.

En LaTeX, los ajustes relativos a la generación del formato y estilos asociados a secciones del documento se realizan con el paquete `'titlesec'` empleado en esta plantilla.

En las secciones siguientes se comenta la inclusión mediante LaTeX de otros elementos de información y ejemplos de los mismos para poder utilizarlos en la elaboración de la memoria del TFG.

Listas

Existen dos tipos de listas: enumeraciones y listas con viñetas. En el primer tipo los elementos de la lista se preceden de una clave numérica o alfabética mientras que en el segundo tipo se emplea una viñeta. En ambos casos se pueden anidar los elementos para crear una jerarquía entre los elementos que forman la lista. En LaTeX se recomienda la inclusión del paquete `'enumitem'` que permite personalizar fácilmente las listas de un documento. A continuación se muestran algunos ejemplos:

Ejemplo de lista con viñetas.

- pera

☞ manzana

- naranja

Ejemplo de lista condensada con separación mínima, en varias columnas y configuración de la etiqueta.

(1) pera
(2) manzana
(3) naranja
(4) patata
(5) calabaza
(6) fresa

Ecuaciones matemáticas

Para escribir ecuaciones matemáticas con LaTeX se recomienda incluir algunos en el documento los paquetes siguientes: `amsmath`, `amsfonts`, `amssymb`.

La composición de ecuaciones requiere el uso de comandos especializados, por tanto para facilitar dicha tarea se aconseja el uso de programas especializados como MathType o asistentes como el incluido en editores como TeXstudio[1] o herramientas en línea.[2] Es muy sencillo incluir fórmulas matemáticas sencillas en el mismo texto en el que se escribe. Por ejemplo, $c^2 = a^2 + b^2$ que podría ser la ecuación representativa del teorema de Pitágoras (ver también ec. 1.1).

Las fórmulas también se pueden separar del texto para que aparezcan destacadas, así:

[1] https://www.texstudio.org/
[2] https://latex.codecogs.com/, http://www.sciweavers.org/free-online-latex-equation-editor

$$c^2 = \int \left(a^2 + b^2\right) \cdot dx$$

Pero si se desea, las ecuaciones pueden ser numeradas de forma automática e incluso utilizar referencias cruzadas a ellas:

$$a^2 = b^2 + c^2 \tag{1.1}$$

Tablas

A continuación se incluyen algunos ejemplos de tablas hechas con LaTeX y paquetes dedicados. Para la realización de tablas más complejas se recomienda la consulta de [1, 10] y el empleo de asistentes o herramientas en línea.[3]

Se debe observar que el título de las tablas se ubica en la parte superior de la tabla. Puesto que el contenido de la tabla es texto, tiene sentido leer primero el título para contextualizar el contenido de la tabla antes de su lectura.

Tabla 1.1: Ejemplo de uso de la macro `cline`

7C0	hexadecimal
3700	octal
11111000000	binario
1984	decimal

Ejemplo de tabla en la que se controla el ancho de la celda.

Tabla 1.2: Ejemplo de tabla con especificación de anchura de columna

Día	Temp Mín (°C)	Temp Máx (°C)	Previsión
Lunes	11	22	Día claro y muy soleado. Sin embargo, la brisa de la tarde puede hacer que las temperaturas desciendan
Martes	9	19	Nuboso con chubascos en muchas regiones. En Cataluña claro con posibilidad de bancos nubosos al norte de la región
Miércoles	10	21	La lluvia continuará por la mañana, pero las condiciones climáticas mejorarán considerablemente por la tarde

[3] https://www.tablesgenerator.com/

Figuras

A diferencia de lo que sucede en las tablas, el título de las figuras aparece en la parte inferior de esta. Para la inclusión de las figuras debe tenerse en cuenta que el contenido de cada una se encuentra en un fichero individual que debería tener el formato y resolución apropiados para garantizar la calidad del resultado final.

En esta sección se añaden ejemplos de muestra para la inclusión de figuras simples y otras compuestas de subfiguras mediante el empleo del paquete `'subcaption'`.

Figura 1.1: Fotografía a color (por J. Salido, CC BY-NC-ND)

Ejemplo de figura compuesta por dos subfiguras incuidas mediante paquete `'subcaption'`. Mediante el uso de etiquetas (`\label`) es posible incluir referencias cruzadas a subfiguras como la fotografía en blanco y negro de la Fig. 1.2b.

(a) Fotografía a color **(b)** Fotografía en blanco y negro

Figura 1.2: Ejemplo de inclusión de subfiguras en un mismo entorno (por J. Salido, CC BY NC ND)

En los trabajos académicos la inclusión de imágenes y figuras que no son propiedad del autor suscitan bastante controversia y son fuente de incumplimiento inadvertido de la ley de propiedad intelectual. Respecto a este hecho se recomienda tanto a estudiantes como tutores consultar documentación informativa sobre el uso correcto de figuras en documentos académicos [13]. Entre las «incorrecciones» más frecuentes al incluir figuras en los documentos académicos se observa:

- Abuso del derecho de cita. Se produce al incluir, con fines exclusivamente decorativos o ilustrativos de la explicación, una figura sujeta a derechos de uso restringido invocando el derecho de cita (incluso con correcta atribución de la obra).

- Incorrecta atribución de la obra. Es habitual confundir al autor de la obra con la fuente de origen de la misma. La fuente es precisa cuando se cita la obra original. Sin embargo, la licencia de muchas obras exige la atribución al autor y la inclusión de la licencia bajo la que se distribuye o hace uso de la misma (véase como ejemplo cómo se realiza una correcta atribución en las Fig. 1.1 y 1.2 mencionando al autor y la licencia Creative-Commons[4] bajo la que se rige el uso de la imagen y el mecanismo de título alternativo para que dicha atribución no aparezca en el índice de figuras).
- Supresión de los detalles de la licencia de uso. Al incluir obras de terceros debemos tener presente los términos de distribución de la misma e incluirlos junto a la atribución de su legítimo autor.

La inclusión de material de *dominio público* o sin restricciones de uso hace innecesaria la atribución al autor pero puede incluirse una nota de agradecimiento.[5]

Algoritmos y listados de código fuente

En los textos científicos relacionados con las TIC[6] (Tecnologías de la Información y Comunicaciones) suelen aparecer porciones de código en los que se explica alguna función o característica relevante del trabajo que se expone. Muchas veces lo que se quiere ilustrar es un algoritmo o método en que se ha resuelto un problema abstrayéndose del lenguaje de programación concreto en que se realiza la implementación. El paquete `algorithm2e` proporciona un entorno `algorithm` para la impresión apropiada de algoritmos tratándolos como objetos flotantes y con mucha flexibilidad de personalización.

Algoritmo 1.1: Cómo escribir algoritmos

Datos : este texto
Resultado : como escribir algoritmos con LaTeX2e

```
1  inicialización;
2  while no es el fin del documento do
3  |  leer actual;
4  |  if comprendido then
5  |  |  ir a la siguiente sección;
6  |  |  la sección actual es esta;
7  |  else
8  |  |  ir al principio de la sección actual;
9  |  end
10 end
```

La inclusión de porciones de código fuente se puede formatear de modo sencillo en LaTeX mediante el uso del paquete `'listings'`. A continuación, se muestran varios ejemplos.

Listado 1.1: Ejemplo de código fuente en lenguaje Java

```
1  // @author www.javadb.com
2  public class Main {
3  // Este método convierte un String a un vector de bytes
4
5  public void convertStringToByteArray() {
6
7  String stringToConvert = "This␣String␣is␣15";
8    byte[] theByteArray = stringToConvert.getBytes();
```

[4] https://creativecommons.org
[5] Incluyendo un texto como: *«Por cortesía de ...»*
[6] Por supuesto en un TFG (Trabajo Fin de Grado) o tesis de un centro superior de informática.

```
    System.out.println(theByteArray.length);
}

public static void main(String[] args) {
    new Main().convertStringToByteArray();
}
}
```

Listado 1.2: Ejemplo de código C

```
// Este código se ha incluido tal cual está en el fichero LaTeX
#include <stdio.h>

int main(int argc, char* argv[]) {
    puts("¡Hola mundo!");
}
```

Listado 1.3: Ejemplo escrito en Matlab

```
function f = fibonacci(n)
 % FIBONACCI Fibonacci sequence
 % f = FIBONACCI(n) generates the first n Fibonacci numbers.
 % Copyright 2014 Cleve Moler
 % Copyright 2014 The MathWorks, Inc.
f = zeros(n,1);
f(1) = 1;
f(2) = 2;
for k = 3:n
f(k) = f(k-1) + f(k-2);
end
```

1.1.2. Bibliografía

Todo el material ajeno se debe citar convenientemente sin contravenir los términos de las licencias de uso y distribución de dicho material. Esto se extiende al uso de diagramas y fotografías. El incumplimiento de la legislación vigente en materia de protección de la propiedad intelectual es responsabilidad exclusiva del autor independientemente de la cesión de derechos que este haya convenido.

La sección de *Bibliografía*, que si se prefiere puede titularse *Referencias*, incluirá un listado ordenado preferentemente por orden alfabético (primer apellido del autor principal) con todas las obras citadas en el texto. En la lista de referencias se especificará para cada obra: autores, título, editorial y año de publicación. Este formato se conseguirá en LaTeX mediante el uso del estilo estándar `'plain'` o cualquier otro derivado con estilo de citación numérica. En algunas titulaciones se obliga a una ordenación por orden de cita en el texto que con BibTeX se puede obtener mediante los estilos estándar `'unsrt'` e `'ieeetr'`.

Es muy importante tener en cuenta que solo se incluirán en esta sección las referencias bibliográficas citadas expresamente en el documento. Si se desea incluir fuentes consultadas, pero no citadas, se puede confeccionar con ellas una sección denominada *Material de consulta*, aunque estas referencias se pueden incluir opcionalmente a lo largo del documento como notas a pie de página.

En las titulaciones técnicas se empleará estilo de citación numérico con el número de la referencia entre corchetes. La cita podrá incluir el número de página concreto de la referencia que se desea citar. El uso correcto de la citación implica dejar claro al lector cuál es el texto, material o idea citado. Las obras referenciadas sin mención explícita o implícita al material concreto citado deberían considerarse material de consulta y por tanto ser agrupados como *Material de consulta* distinguiéndolas claramente de aquellas otras en las que si se recurre a la citación.

En las titulaciones que requieren un estilo de citación de tipo autor-año (no numérico), se puede incluir el paquete LaTeX `'apalike'` y especificar este mismo estilo en la sección de bibliografía en el argumento del comando `'bibliographystyle'`.

Cuando se desee incluir referencias a páginas genéricas de la Web sin mención expresa a un artículo con título y autor definido, dichas referencias podrán hacerse como notas al pie de página o como un apartado de fuentes de consultas dedicado a las *Direcciones de Internet*. Por el contrario los documentos electrónicos publicados en Internet se pueden incluir empleando el tipo de entrada `'misc'` como se muestra en la bibliografía que acompaña esta plantilla.

CAPÍTULO 2
Objetivo

Introduce y motiva la problemática (i.e. *¿cuál es el problema que se plantea y por qué es interesante su resolución?*)

Debe concretar y exponer detalladamente el problema a resolver, el entorno de trabajo, la situación y qué se pretende obtener. También puede contemplar las limitaciones y condicionantes a considerar para la resolución del problema (lenguaje de construcción, equipo físico, equipo lógico de base o de apoyo, etc.). Si se considera necesario, esta sección puede titularse *Objetivos del TFG e hipótesis de trabajo*. En este caso, se añadirán las hipótesis de trabajo que el alumno pretende demostrar con su TFG.

Una de las tareas más complicadas al proponer un TFG es plantear su Objetivo. La dificultad deriva de la falta de consenso respecto de lo que se entiende por *objetivo* en un trabajo de esta naturaleza. En primer lugar se debe distinguir entre dos tipos de objetivo:

(A) La *finalidad específica* del TFG que se plantea para resolver una problemática concreta aplicando los métodos y herramientas adquiridos durante la formación académica. Por ejemplo, *«Desarrollo de una aplicación software para gestionar reservas hoteleras* on-line*»*.

(B) El *propósito académico* que la realización de un TFG tiene en la formación de un graduado. Por ejemplo, la *adquisición de competencias específicas de la especialización* cursada.

En el ámbito de la memoria del TFG se tiene que definir el primer tipo de objetivo, mientras que el segundo tipo es el que se añade al elaborar la propuesta de un TFG presentada ante un comité para su aprobación. *Este segundo tipo de objetivo no debe incluirse en la memoria y en todo caso solo debe hacerse en la sección de conclusiones finales.*[1]

Un objetivo bien planteado debe estar determinado en términos del *«producto final»* esperado que resuelve un problema específico. Es por tanto un sustantivo que debería ser *concreto* y *medible*. El objetivo planteado puede pertenecer una de las categorías que se indica a continuación:

- *Diseño y desarrollo de «artefactos»* (habitual en las ingenierías). Por la naturaleza de los programas informáticos (software), los trabajos que implican su diseño suelen contemplar también el desarrollo o implementación de prototipos. Esto es menos frecuente en otras áreas de ingeniería en las que claramente se separa la fase de diseño o realización de un proyecto, frente a la ejecución del mismo (p.ej. ingeniería civil, arquitectura, etc.).
- *Estudio* que ofrece información novedosa sobre un tema (usual en las ramas de ciencias y humanidades).
- *Validación de una hipótesis* de partida (propio de los trabajos científicos y menos habitual en el caso de los TFG).

Estas categorías no son excluyentes, de modo que es posible plantear un trabajo cuyo objetivo sea el diseño y desarrollo de un «artefacto» y este implique un estudio previo o la validación de

[1] En lagunas titulaciones es obligatorio que la memoria explique las competencias específicas alcanzadas con la realización del trabajo.

alguna hipótesis para guiar el proceso. En este caso y cuando el objetivo sea lo suficientemente amplio puede ser conveniente su descomposición en elementos más simples hablando de *subobjetivos*. Por ejemplo, un programa informático puede descomponerse en módulos o requerir un estudio previo para plantear un nuevo algoritmo que será preciso validar. La descomposición de un objetivo principal en subobjetivos u objetivos secundarios debería ser natural (no forzada), bien justificada y sólo pertinente en los trabajos de gran amplitud.

Junto con la definición del objetivo del trabajo se puede especificar los *requisitos* que debe satisfacer la solución aportada. Estos requisitos especifican *características* que debe poseer la solución y *restricciones* que acotan su alcance. En el caso de un trabajo cuyo objetivo es el desarrollo de un «artefacto» los requisitos pueden ser *funcionales* y *no funcionales*.

Al redactar el objetivo de un TFG se debe evitar confundir los medios con el fin. Así es habitual encontrarse con objetivos definidos en términos de las *acciones* (verbos) o *tareas* que será preciso realizar para llegar al verdadero objetivo. Sin embargo, a la hora de planificar el desarrollo del trabajo si es apropiado descomponer todo el trabajo en *hitos* y estos en *tareas* para facilitar dicha *planificación*.

La categoría del objetivo planteado justifica modificaciones en la organización genérica de la memoria del trabajo. Así en el caso de estudios y validación de hipótesis el apartado de resultados y conclusiones debería incluir los resultados de experimentación y los comentarios de cómo dichos resultados validan o refutan la hipótesis planteada.

CAPÍTULO 3
Metodología

En este capítulo se debe detallar las metodologías empleadas para planificación y desarrollo del trabajo, así como explicar de modo claro y conciso cómo se han aplicado dichas metodologías.

A continuación se incluye una guía rápida que puede ser de gran utilidad en la elaboración de este capítulo.

3.1. GUÍA RÁPIDA DE LAS METODOLOGÍAS DE DESARROLLO DE SOFTWARE

3.1.1. Proceso de desarrollo de software

El **proceso de desarrollo de software** se denomina también **ciclo de vida del desarrollo del software** (*SDLC, Software Development Life-Cycle*) y cubre las siguientes actividades:

1.- **Obtención y análisis de requisitos** (*requirements analysis*). Es la definición de la funcionalidad del software a desarrollar. Suele requerir entrevistas entre los ing. de software y el cliente para obtener el 'qué' y 'cómo'. Permite obtener una *especificación funcional* del software.

2.- **Diseño** (*SW design*). Consiste en la definición de la arquitectura, los componentes, las interfaces y otras características del sistema o sus componentes.

3.- **Implementación** (*SW construction and coding*). Es el proceso de codificación del software en un lenguaje de programación. Constituye la fase en que tiene lugar el desarrollo de software.

4.- **Pruebas** (*testing and verification*). Verificación del correcto funcionamiento del software para detectar fallos lo antes posible. Persigue la obtención de software de calidad. Consisten en pruebas de *caja negra* y *caja blanca*. Las primeras comprueban que la funcionalidad es la esperada y para ello se verifica que ante un conjunto amplio de entradas, la salida es correcta. Con las segundas se comprueba la robustez del código sometiéndolo a pruebas cuya finalidad es provocar fallos de software. Esta fase también incorpora la *pruebas de integración* en las que se verifica la interoperabilidad del sistema con otros existentes.

5.- **Documentación** (*documentation*). Persigue facilitar la mejora continua del software y su mantenimiento.

6.- **Despliegue** (*deployment*). Consiste en la instalación del software en un entorno de producción y puesta en marcha para explotación. En ocasiones implica una fase de *entrenamiento* de los usuarios del software.

7.- **Mantenimiento** (*maintenance*). Su propósito es la resolución de problemas, mejora y adaptación del software en explotación.

3.1.2. Metodologías de desarrollo software

Las metodologías son el modo en que las fases del proceso software se organizan e interaccionan para conseguir que dicho proceso sea reproducible y predecible para aumentar la productividad y la calidad del software.

Una metodología es una colección de:

A. **Procedimientos** (indican cómo hacer cada tarea y en qué momento),

B. **Herramientas** (ayudas para la realización de cada tarea), y

C. **Ayudas documentales**.

Cada metodología es apropiada para un tipo de proyecto dependiendo de sus características técnicas, organizativas y del equipo de trabajo. En los entornos empresariales es obligado, a veces, el uso de una metodología concreta (p. ej. para participar en concursos públicos). El estándar internacional ISO/IEC 12270 describe el método para seleccionar, implementar y monitorear el ciclo de vida del software.

Mientras que unas intentan sistematizar y formalizar las tareas de diseño, otras aplican técnicas de gestión de proyectos para dicha tarea. Las metodologías de desarrollo se pueden agrupar dentro de varios enfoques según se señala a continuación.

1. **Metodología de Análisis y Diseño de Sistemas Estructurados** (*SSADM, Structured Systems Analysis and Design Methodology*). Es uno de los paradigmas más antiguos. En esta metodología se emplea un modelo de desarrollo en cascada (*waterfall*). Las fases de desarrollo tienen lugar de modo secuencial. Una fase comienza cuando termina la anterior. Es un método clásico poco flexible y adaptable a cambios en los requisitos. Hace especial hincapié en la planificación derivada de una exhaustiva definición y análisis de los requisitos. Son metodologías que no lidian bien con la flexibilidad requerida en los proyectos de desarrollo software. Derivan de los procesos en ingeniería tradicionales y están enfocadas a la reducción del riesgo. Emplea tres técnicas clave:
   - Modelado lógico de datos (*Logical Data Modelling*),
   - Modelado de flujo de datos (*Data Flow Modelling*), y
   - Modelado de Entidades y Eventos (*Entity Event Modelling*).
2. **Metodología de Diseño Orientado a Objetos** (*OOD, Object-Oriented Design*). Está muy ligado a la OOP (Programación Orientada a Objetos) en que se persigue la reutilización. A diferencia del anterior, en este paradigma los datos y los procesos se combinan en una única entidad denominada *objetos* (o clases). Esta orientación pretende que los sistemas sean más modulares para mejorar la eficiencia, calidad del análisis y el diseño. Emplea extensivamente el Lenguaje Unificado de Modelado (UML) para especificar, visualizar, construir y documentar los artefactos de los sistemas software y también el modelo de negocio. UML proporciona una serie diagramas de básicos para modelar un sistema:
   - Diagrama de Clase (*Class Diagram*). Muestra los objetos del sistema y sus relaciones.
   - Diagrama de Caso de Uso (*Use Case Diagram*). Plasma la funcionalidad del sistema y quién interacciona con él.
   - Diagrama de secuencia (*Sequence Diagram*). Muestra los eventos que se producen en el sistema y como este reacciona ante ellos.
   - Modelo de Datos (*Data Model*).
3. **Desarrollo Rápido de Aplicaciones** (*RAD, Rapid Application Developmnent*). Su filosofía es sacrificar calidad a cambio de poner en producción el sistema rápidamente con la funcionalidad esencial. Los procesos de especificación, diseño e implementación son simultáneos. No se realiza una especificación detallada y se reduce la documentación de diseño. El sistema se diseña en una serie de pasos, los usuarios evalúan cada etapa en la que proponen cambios y nuevas mejoras. Las interfaces de usuario se desarrollan habitualmente mediante sistemas interactivos de desarrollo. En vez de seguir un modelo de desarrollo en cascada sigue un modelo en espiral (Boehm). La clave de este modelo es el desarrollo continuo que ayuda a minimizar los riesgos. Los desarrolladores deben definir las características de mayor prioridad. Este tipo

de desarrollo se basa en la creación de prototipos y realimentación obtenida de los clientes para definir e implementar más características hasta alcanzar un sistema aceptable para despliegue.

4. **Metodologías Ágiles**. *"[...] envuelven un enfoque para la toma de decisiones en los proyectos de software, que se refiere a métodos de ingeniería del software basados en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan con el tiempo según la necesidad del proyecto. Así el trabajo es realizado mediante la colaboración de equipos auto-organizados y multidisciplinarios, inmersos en un proceso compartido de toma de decisiones a corto plazo. Cada iteración del ciclo de vida incluye: planificación, análisis de requisitos, diseño, codificación, pruebas y documentación. Teniendo gran importancia el concepto de "Finalizado"(Done), ya que el objetivo de cada iteración no es agregar toda la funcionalidad para justificar el lanzamiento del producto al mercado, sino incrementar el valor por medio de "software que funciona"(sin errores). Los métodos ágiles enfatizan las comunicaciones cara a cara en vez de la documentación. [...]"*[1]

3.1.3. Proceso de testing

1. *Pruebas modulares* (pruebas unitarias). Su propósito es hacer pruebas sobre un módulo tan pronto como sea posible. Las *pruebas unitarias* que comprueban el correcto funcionamiento de una unidad de código. Dicha unidad elemental de código consistiría en cada función o procedimiento, en el caso de programación estructurada y cada clase, para la programación orientada a objetos. Las características de una prueba unitaria de calidad son: *automatizable* (sin intervención manual), *completa*, *reutilizable*, *independiente* y *profesional*.
2. *Pruebas de integración*. Pruebas de varios módulos en conjunto para comprobar su interoperabilidad.
3. *Pruebas de caja negra*.
4. *Beta testing*.
5. *Pruebas de sistema y aceptación*.
6. *Training*.

3.1.4. Herramientas CASE (*Computer Aided Software Engineering*)

Las herramientas CASE están destinadas a facilitar una o varias de las tareas implicadas en el ciclo de vida del desarrollo de software. Se pueden dividir en la siguientes categorías:

1. Modelado y análisis de negocio.
2. Desarrollo. Facilitan las fases de diseño y construcción.
3. Verificación y validación.
4. Gestión de configuraciones.
5. Métricas y medidas.
6. Gestión de proyecto. Gestión de planes, asignación de tareas, planificación, etc.

IDE (Integrated Development Environment)

- Notepad++
- Visual Studio Code
- Atom
- GNU Emacs
- NetBeans
- Eclipse
- Qt Creator
- jEdit
- ItelliJ IDEA

[1] Fuente: Wikipedia

Depuración

- GNU Debugger

Testing

- JUnit. Entorno de pruebas para Java.
- CUnit. Entorno de pruebas para C.
- PyUnit. Entorno de pruebas para Python.

Repositorios y control de versiones

- Git
- Mercurial
- Github
- Bitbucket
- SourceTree

Documentación

- LaTeX
- Markdown
- Doxygen
- DocGen
- Pandoc

Gestión y planificación de proyectos

- Trello
- Jira
- Asana
- Slack
- Basecamp
- Teamwork Projects
- Zoho Projects

3.1.5. Fuentes de información adicional

- Top 6 Software Development Methodologies. Maja Majewski. Planview LeanKit, 2019.
- 12 Best software development methodologies with pros and cons. acodez, 2018.
- Software Development Methodologies. Association of Modern Technologies Professionals, 2019.

CAPÍTULO 4

Resultados

En esta sección se describirá la aplicación del método de trabajo presentado en el capítulo 3, mostrando los elementos (modelos, diagramas, especificaciones, etc.) más importantes.

Este apartado debe explicar cómo el empleo de la metodología permite satisfacer tanto el objetivo principal como los específicos planteados en el TFG así como los requisitos exigidos (según exposición en cap. 2).

4.1. COSTES, PLANIFICACIÓN Y PRESUPUESTO

Si el TFG consiste en el desarrollo e implementación de un prototipo, la memoria debe incluir el coste del prototipo considerando tanto el hardware como los recursos humanos necesarios para su desarrollo. Esta explicación se puede incluir en secciones dentro del capítulo de resultados. Si además el TFG contempla la puesta en marcha de la solución diseñada, la planificación y presupuesto —de la puesta en marcha— pueden constituir capítulos específicos subsiguientes en la memoria, ya que en sí misma la puesta en marcha constituye un proyecto de ejecución separado del diseño.

Cuando se tiene en cuenta la puesta en marcha de un proyecto de ingeniería, la planificación y presupuesto que se realizan de modo previo a su ejecución son críticos para gestionar los recursos que permitan alcanzar los objetivos de calidad, temporales y económicos previstos para el proyecto.

Es muy importante que todas las justificaciones aportadas se sustenten no solo en juicios de valor sino en evidencias tangibles como: historiales de actividad, repositorios de código y documentación, porciones de código, trazas de ejecución, capturas de pantalla, demos, etc.

CAPÍTULO 5

Conclusiones

En este capítulo se realizará un juicio crítico y discusión sobre los resultados obtenidos. *Cuidado, esta discusión no debe confundirse con una valoración del enriquecimiento personal que supone la realización del trabajo como culminación de una etapa académica.* Aunque de gran importancia, esta última valoración debe quedar fuera de la memoria del trabajo y solo debe ahondarse en ella ante requerimiento explícito del comité en el acto de defensa.

Si es pertinente deberá incluir información sobre trabajos derivados como publicaciones o ponencias en preparación, así como trabajos futuros *(solo si estos están iniciados o planificados en el momento que se redacta el texto)*. Evitar hacer una lista de posibles mejoras. Contrariamente a lo que alguno pueda pensar generalmente aportan impresión de trabajo incompleto o inacabado.[1]

5.1. JUSTIFICACIÓN DE COMPETENCIAS ADQUIRIDAS

Es muy importante recordar que según la normativa vigente en la ESI, el capítulo de conclusiones debe incluir *obligatoriamente* un apartado destinado a justificar la aplicación en el TFG de competencias específicas (una o más) adquiridas en la tecnología específica cursada.

En el TFG se han aplicado las competencias correspondientes a la Tecnología Específica de *[poner lo que corresponda]*:

Código de la competencia 1: *[Texto de la competencia 1]*. Explicación de cómo se ha aplicado en el TFG.

... otras más si las hubiera.

[1] Puede reflexionarse en ello por si en la defensa del trabajo se pregunta sobre estas posibles mejoras.

Bibliografía

[1] Alexánder Borbón and Walter Mora. *Edición de textos científicos con LaTeX. Composición, diseño editorial, gráficos, Inkscape, Tikz y presentaciones Beamer.* Instituto Tecnológico de Costa Rica, 2 edition, 2021.

[2] Bernardo Cascales, Pascual Lucas, José M. Mira, Antonio J. Pallarés, and Salvador Sánchez-Pedreño. *El libro de LaTeX.* Pearson Education, 2005. ISBN: 9788420537795.

[3] Escuela Superior de Informática, Universidad de Castilla-La Mancha. Guía de estilo y formato para Trabajos Fin de Grado. URL: https://pruebasaluuclm.sharepoint.com/sites/esicr/tfg/SiteAssets/SitePages/Inicio/20190304-GuiaEstiloFormatoTFG.pdf, March 2019. Último acceso: sep. 2021.

[4] Michel Goossens, Frank Mittelbach, Sebastian Rahtz, Denis Roeget, and Herbert Voß. *The LaTeX graphics companion.* Addison-Wesley Reading, MA, second edition, 2007. ISBN: 9780321508928.

[5] George A. Grätzer. *First steps in LaTeX.* Springer Verlag, October 1999. ISBN: 0817641327.

[6] George A. Grätzer. *More math into LaTeX.* Birkhauser, fourth edition, 2007. ISBN: 978-3319237954.

[7] Leslie Lamport. *LaTeX: A document preparation system.* Addison-Wesley, second edition, June 1994. ISBN: 978-0201529838.

[8] Frank Mittelbach and Michel Goossens. *The LaTeX companion.* Addison, Reading, Mass, second edition, 2004. ISBN: 0-201-36299-6.

[9] Tobias Oetiker, Hubert Partl, Irene Hyna, and Elisabeth Schlegl. *La introducción no-tan-corta a LaTeX2e*, 2014.

[10] Overleaf. Overleaf knowledge base. URL: https://www.overleaf.com/learn, 2021. Último acceso: sep. 2021.

[11] Jesús Salido. Curso: LaTeX esencial para preparación de tfg, tesis y otros documentos académicos. URL: http://visilab.etsii.uclm.es/?page_id=1468, 2010. Último acceso: sep. 2021.

[12] Universidad Carlos III de Madrid. Guía temática del TFG. URL: https://uc3m.libguides.com/TFG, 2021. Último acceso: sep-2021.

[13] Universidad de Cantabria. Cómo usar imágenes en trabajos. Artículo técnico disponible en URL: https://web.unican.es/buc/Documents/Formacion/guia_imagenes.pdf, 2018. Último acceso: sep. 2021.

[14] Universitat Oberta de Catalunya. Comunicación eficaz y redacción. URL: https://www.uoc.edu/portal/es/servei-linguistic/redaccio/10-recomanacions/index.html. Último acceso: sep. 2021.

[15] WikiMedia. LaTeX Wikibook. URL: http://en.wikibooks.org/wiki/LaTeX, 2010. Último acceso: sep. 2021.

[16] Leonor Zozaya. Redacción de textos. recomendaciones para presentar trabajos académicos. Blog disponible en URL: http://redaccion.hypotheses.org/, 2017. ISSN: 2444-8885.

ANEXOS

ANEXO A

El primer anexo

Los anexos se incluirá de modo opcional material suplementario que podrá consistir en breves manuales, listados de código fuente, esquemas, planos y en general aquello que complementa a la memoria, pero no se incluye en el texto principal. Se recomienda que no sean excesivamente voluminosos, aunque su extensión no está sometida a la regulación por normativa, ya que esta afecta únicamente al texto principal de la memoria.