# Proyecto Docente

Rubén Béjar Hernández

10 de mayo de 2017

# Índice general

**Introducción** **1**

**1. Contexto legal e institucional** **3**
1.1. La Universidad en España . . . . . 3
1.1.1. La Ley Orgánica de Universidades . . . . . 4
1.1.2. La Ley de Ordenación del Sistema Universitario de Aragón . . . 6
1.2. El Espacio Europeo de Educación Superior . . . . . 6
1.2.1. La dimensión del Espacio Europeo de Educación Superior . . . 7
1.2.2. Implantación del Espacio Europeo de Educación Superior en España . . . . . 8
1.2.3. El marco de calificaciones europeo . . . . . 8
1.2.4. Objetivos para el periodo 2015-2020 . . . . . 10
1.3. La Universidad de Zaragoza . . . . . 12
1.3.1. Breve historia . . . . . 13
1.3.2. La Escuela de Ingeniería y Arquitectura . . . . . 13
1.3.3. El Departamento de Informática e Ingeniería de Sistemas . . . . 14
1.3.4. El Área de Lenguajes y Sistemas Informáticos . . . . . 15

**2. Contexto profesional** **19**
2.1. La informática . . . . . 19
2.2. La ingeniería . . . . . 20
2.3. La regulación de la ingeniería informática . . . . . 22
2.4. El sector TIC . . . . . 23

**3. Contexto académico** **26**
3.1. Estructura de los estudios universitarios de ingeniería informática . . . 26
3.2. El grado en ingeniería informática en la Universidad de Zaragoza . . . . 32
3.3. El máster universitario en ingeniería informática en la Universidad de Zaragoza . . . . . 36

**4. La enseñanza-aprendizaje en la educación superior** **38**
4.1. El aprendizaje . . . . . 38
4.2. Los estudiantes . . . . . 41
4.3. Los equipos de estudiantes . . . . . 43
4.4. Metodología de enseñanza-aprendizaje . . . . . 45
4.4.1. La lección magistral . . . . . 45

4.4.2. Las preguntas . . . 46
4.4.3. Las prácticas de laboratorio . . . 47
4.4.4. La enseñanza con grupos pequeños . . . 47
4.4.5. El aprendizaje basado en problemas y proyectos . . . 48
4.4.6. El *e-learning* y otras tendencias . . . 49
4.5. La evaluación . . . 51

**5. La materia de gestión de proyectos de software 53**
5.1. Los proyectos . . . 53
5.1.1. El origen de las técnicas de gestión de proyectos: el método del camino crítico y la planificación . . . 54
5.1.2. El alcance de la gestión de proyectos . . . 55
5.1.3. Fundamentos teóricos de la gestión de proyectos . . . 56
5.1.4. El triángulo de hierro . . . 58
5.1.5. Los proyectos de software en la actualidad . . . 59
5.1.6. Otros tipos de proyectos de ingeniería . . . 62
5.1.7. El futuro . . . 62
5.2. La gestión de proyectos software en el Computer Science Curricula de ACM/IEEE . . . 63
5.3. La gestión y dirección de proyectos de software en ingeniería informática en la universidad española . . . 64
5.3.1. En la Universidad de Zaragoza . . . 65
5.3.2. En otras universidades españolas . . . 66

**6. La asignatura de Proyecto Software 69**
6.1. La asignatura en la titulación . . . 70
6.2. El programa de la asignatura . . . 70
6.2.1. Introducción a la gestión de proyectos . . . 70
6.2.2. Gestión de configuraciones . . . 71
6.2.3. Gestión de configuraciones. Control de versiones con Git . . . 72
6.2.4. Proceso de desarrollo y equipo humano . . . 75
6.2.5. Métricas y estimaciones . . . 76
6.2.6. Planificación, gestión de riesgos, y el plan de gestión del proyecto software . . . 78
6.2.7. Aseguramiento de la calidad. El modelo de madurez CMMI y la norma ISO 9001 . . . 80
6.2.8. El marco legislativo del software . . . 82
6.3. El proyecto en equipo . . . 83
6.3.1. Ideas . . . 84
6.3.2. Solicitud de trabajo . . . 84
6.3.3. Propuesta técnica y económica . . . 84
6.3.4. Contrato . . . 85
6.3.5. El plan de gestión del proyecto . . . 85
6.3.6. Reuniones con los equipos . . . 87
6.3.7. Presentación comercial . . . 91
6.3.8. Presentación técnica . . . 92

6.4. Metodologías de enseñanza-aprendizaje . . . . . 92
6.5. Planificación temporal de la asignatura . . . . . 92
6.6. La evaluación . . . . . 94
6.7. Bibliografía comentada . . . . . 94

**7. La asignatura de Gestión de Proyecto Software 97**
7.1. La asignatura en la titulación . . . . . 98
7.2. El programa de la asignatura . . . . . 99
7.2.1. El planteamiento de producto . . . . . 99
7.2.2. Scrum 1 - Introducción . . . . . 100
7.2.3. Scrum 2 - Principios ágiles . . . . . 100
7.2.4. Scrum 3 - Sprints . . . . . 101
7.2.5. Scrum 4 - Requisitos e historias de usuario . . . . . 102
7.2.6. Scrum 5 - La pila del producto . . . . . 103
7.2.7. Scrum 6 - Estimación y velocidad . . . . . 104
7.2.8. Scrum 7 - Planificación . . . . . 105
7.2.9. Scrum 8 - Los sprints en detalle . . . . . 106
7.2.10. Scrum 9 - Los roles en detalle . . . . . 107
7.2.11. Scrum 10 - Arquitectura y deuda técnica . . . . . 109
7.2.12. Scrum 11 - Tests en proyectos ágiles . . . . . 110
7.2.13. Scrum 12 - Gestión de configuraciones en proyectos ágiles: introducción a la entrega continua . . . . . 111
7.2.14. Técnicas de gestión 1 - Gestión de riesgos . . . . . 112
7.2.15. Técnicas de gestión 2 - Gestión del tiempo . . . . . 114
7.2.16. Técnicas de gestión 3 - Gestión del coste . . . . . 115
7.2.17. Técnicas de gestión 4 - Gestión de la financiación . . . . . 116
7.3. El proyecto en equipo . . . . . 118
7.3.1. Organización temporal del proyecto . . . . . 118
7.4. Metodologías de enseñanza-aprendizaje . . . . . 119
7.5. Planificación temporal de la asignatura . . . . . 120
7.6. La evaluación . . . . . 122
7.7. Bibliografía comentada . . . . . 123

**Bibliografía 125**

# Índice de figuras

1.1. Marco Español de Cualificaciones para la Educación Superior (MECES) y equivalencia con el EQF. Tomado de `https://ec.europa.eu/ploteus/sites/eac-eqf/files/cuadro-meces.pdf` . . . . . 10

5.1. Joseph Priestley: Chart of Biography (1765) . . . . . 55
5.2. El triángulo de hierro (licencia CC BY-SA 3.0 por John Manuel Kennedy T.) . . . . . 58
5.3. Resolución de proyectos 2004-2012. (c) Standish Group (2013) . . . . . 59
5.4. Sobrecostes, retrasos y características completadas. (c) Standish Group (2013) . . . . . 60
5.5. Resolución de proyectos: pequeños frente a grandes. (c) Standish Group (2013) . . . . . 60
5.6. Resultado de proyectos según metodologías. (c) 2014 Scott W. Ambler, www.ambysoft.com/surveys/ . . . . . 61
5.7. Resultado de proyectos desde distintos puntos de vista según metodologías. (c) 2014 Scott W. Ambler, www.ambysoft.com/surveys/ . . . . . 61

6.1. (c) http://xkcd.com/1597/, under a Creative Commons Attribution Non Commercial 2.5 License . . . . . 73
6.2. Un ejemplo de repositorio Git con dos ramas mezcladas . . . . . 74

# Índice de tablas


1.1. Asignaturas de LSI en la Universidad de Zaragoza . . . . . 15
1.1. Asignaturas de LSI en la Universidad de Zaragoza . . . . . 16
1.1. Asignaturas de LSI en la Universidad de Zaragoza . . . . . 17
1.1. Asignaturas de LSI en la Universidad de Zaragoza . . . . . 18

3.1. Módulos Máster Ingeniería Informática . . . . . 28
3.1. Módulos Máster Ingeniería Informática . . . . . 29
3.2. Módulos Grado Ingeniería Informática . . . . . 30
3.2. Módulos Grado Ingeniería Informática . . . . . 31
3.2. Módulos Grado Ingeniería Informática . . . . . 32
3.3. Asignaturas de formación básica . . . . . 33
3.4. Asignaturas de formación común . . . . . 34
3.5. Asignaturas optativas . . . . . 34
3.6. Asignaturas específicas de cada especialidad . . . . . 35
3.7. Asignaturas obligatorias del máster . . . . . 36
3.8. Asignaturas optativas del máster . . . . . 37

4.1. Acciones para motivar a los estudiantes . . . . . 42
4.1. Acciones para motivar a los estudiantes . . . . . 43

6.1. Resultados de aprendizaje de Proyecto Software en la EINA (tomados de la ficha de la asignatura) . . . . . 69
6.2. Competencias adquiridas en Proyecto Software en la EINA (tomados de la ficha de la asignatura) . . . . . 70
6.3. Calendario por semanas . . . . . 93

7.1. Resultados de aprendizaje de Gestión de Proyecto Software en la EINA (tomados de la ficha de la asignatura) . . . . . 97
7.1. Resultados de aprendizaje de Gestión de Proyecto Software en la EINA (tomados de la ficha de la asignatura) . . . . . 98
7.2. Competencias adquiridas en Gestión de Proyecto Software en la EINA (tomados de la ficha de la asignatura) . . . . . 98
7.3. Hitos del proyecto . . . . . 120
7.4. Calendario por semanas . . . . . 121

# Concurso

Este Proyecto Docente ha sido escrito para la participación en el concurso público para la provisión de plazas de Profesor Titular de Universidad en la Universidad de Zaragoza, convocado el 13 de marzo de 2017 y publicado en el BOE número 71, de 24 de marzo de 2017. En concreto, este proyecto se presenta para el concurso público de la plaza 2017_16, del área de conocimiento de Lenguajes y Sistemas Informáticos y el Departamento de Informática e Ingeniería de Sistemas, con actividades docentes e investigadoras de Proyecto Software y Gestión de Proyecto Software.

# Agradecimientos

Quiero mostrar mi agradecimiento a los profesores con los que he compartido la impartición de las asignaturas Proyecto Software y Gestión de Proyecto Software en el Grado en Ingeniería Informática de la Universidad de Zaragoza, y antes en la Ingeniería Informática en esa Universidad. Especialmente a los doctores Pedro R. Muro-Medrano y F. Javier Zarazaga Soria. Sin su trabajo en el diseño, preparación e impartición de las versiones anteriores de estas asignaturas, y sin la experiencia que me han aportado a lo largo de los años, mi Proyecto Docente, focalizado en estas asignaturas, no sería lo que es.

# Introducción

La carrera profesional como profesor de universidad tiene algunos elementos distintivos. Por una parte, es necesario haber empleado muchos años en la formación necesaria para adquirir el conocimiento y las habilidades necesarias hasta alcanzar el grado académico de doctor, pero por otra parte no se recibe mucha formación, ni muy formal, sobre "cómo ser profesor", ni en la vertiente docente, ni en la investigadora, ni en la de gestión universitaria[1]. El resultado es que todo eso (diseñar un curso, dar una clase a un grupo grande, resolver un conflicto que surja durante una revisión de examen, ser miembro de un tribunal, escribir y someter un manuscrito junto a una carta de presentación del mismo, comportarse en una cena de un congreso, seguir los rituales adecuados en una reunión con personal directivo de una empresa, coordinar una petición de proyecto, participar en la dirección de un departamento o centro etc.) se aprende sobre la marcha, bajo demanda, y lo que se aprende depende mucho de la gente que hayas tenido más cerca y de tu experiencia directa.

No es que esta aproximación no funcione, de hecho es la aproximación que ha funcionado para todos los profesores de hoy en día, pero cuesta mucho tiempo y esfuerzo, y el resultado es un poco como el valor al soldado, que se presupone y solo se reconoce y distingue en casos realmente extraordinarios. Tampoco es que otros profesionales no se enfrenten a situaciones parecidas (a casi nadie le enseñan en la carrera como comportarse en un reunión con un alto ejecutivo de un banco o un alto cargo de la administración pública, por ejemplo) pero la triple vertiente de la labor del profesor (docencia, investigación y gestión) la hace especialmente complicada en nuestro caso.

Las leyes reconocen las diversas facetas de los profesores de universidad, y buscan facilitar que puedan cumplirse con libertad y efectividad. Ya la propia Constitución Española, en su artículo 20, reconoce y protege los derechos a expresar y difundir libremente los pensamientos, ideas y opiniones, a la producción y creación literaria, artística, científica y técnica, a la libertad de cátedra y a comunicar o recibir libremente información veraz.

La Ley Orgánica 6/2001, de 21 de Diciembre, de Universidades (LOU), señala que la actividad y la autonomía de la Universidad se fundamentan en el principio de libertad académica, manifestado en las libertadas de cátedra, de investigación y de estudio, y que su función principal es el servicio público de la educación superior mediante la investigación, la docencia y el estudio. También indica que la investigación científica, técnica y artística es fundamento esencial de la docencia y una herramienta básica para el desarrollo de la sociedad mediante la transferencia de sus resultados a

[1] Y aún menos en las que podríamos denominar "secundarias", como la divulgación científica y las relaciones con el entorno profesional no universitario

la misma, que es uno de los objetivos esenciales de la universidad y que se reconoce la libertad de investigación en el ámbito universitario. Con respecto a la docencia, derecho y deber de los profesores de las universidades, la LOU señala dedicaciones para cada figura contractual, y por ejemplo el PDI funcionario dedicará, con carácter general, 24 créditos ECTS cada curso a esta actividad. Este derecho y deber se ejercerá con libertad de cátedra, sin más límites que los establecidos en la Constitución y en las leyes, y los derivados de la organización de las enseñanzas en sus Universidades.

La investigación es, según la LOU, también derecho y deber del personal docente e investigador de las universidades, y por tanto las universidades facilitarán la compatibilidad de esta con la docencia, e incentivarán el desarrollo de una trayectoria profesional que permita una dedicación más intensa a la actividad docente o a la investigadora. La LOU también señala respecto a la investigación, que la transferencia del conocimiento resultante de la misma es función de las universidades, que determinarán y establecerían los medios necesarios para facilitar la prestación de este servicio social por parte del personal docente e investigador.

Vistas las múltiples facetas del profesor de universidad, y la complejidad de estas, es una consecuencia normal que un proyecto que trate de recoger sus aspectos más fundamentales sea un documento largo, complejo y aún así nunca terminado o definitivo.

El proyecto docente que se presenta en este documento contextualiza la docencia desde las perspectivas legal, institucional, profesional y académica, establece unos fundamentos metodológicos de enseñanza-aprendizaje, y desarrolla el programa de dos asignaturas, Proyecto Software y Gestión de Proyecto Software, partiendo de una visión general de la materia de la gestión de proyectos de software. Estas dos asignaturas se imparten en la actualidad en el Grado en Ingeniería Informática, en la Escuela de Ingeniería y Arquitectura (EINA) de la Universidad de Zaragoza.

Las razones por las que he seleccionado estas dos asignaturas son principalmente dos. La primera es que he participado en su diseño, impartición y mejora continuada en el grado en ingeniería informática que de la EINA desde que este grado se puso en marcha, y creo que es interesante poder transmitir esta experiencia. Y la segunda es que en mi trabajo de transferencia de resultados de investigación, incluyendo especialmente mi participación en GeoSLab, empresa spin-off de la Universid de Zaragoza, me ha aportado una experiencia directa sobre la gestión de proyectos de software en la empresa que creo que contribuye a que mi trabajo como docente en esa materia sea mejor, al poder transmitir parte de esa experiencia de primera mano a mis estudiantes.

# Capítulo 1

# Contexto legal e institucional

Este capítulo describe el contexto legal e institucional que enmarca la labor universitaria objeto de este proyecto docente. El contexto legal en España se recoge con toda completitud en [Alcubilla, 2016], que es una colaboración entre Crue Universidades Españolas y el Boletín Oficial del Estado (BOE) con toda la normativa legal y reglamentaria sobre educación superior, y que se actualiza conforme se producen cambios. Sus principales elementos se describen en la sección 1.1. La sección 1.2 describe el contexto más amplio que proporciona el Espacio Europeo de Educación Superior que, desde su puesta en marcha con la declaración de Bolonia en 1999, ha sido la referencia para el diseño de la enseñanza superior en Europa. Finalmente, la sección 1.3 describe el contexto institucional más cercano, que es el formado por la Universidad de Zaragoza y la Escuela, Departamento y Área de Conocimiento donde se enmarca este proyecto docente.

## 1.1. La Universidad en España

El principio de autonomía universitaria, que es fundamental para una institución que debe ser tanto formativa como crítica con la sociedad, está recogido en la Constitución española. Esta autonomía incluye la libertad académica (de cátedra, de investigación y de estudio), pero también la autonomía de gestión y administración de los recursos propios.

Actualmente, la ley española enuncia las potestades que incluye dicha autonomía en la Ley Orgánica 6/2001, de 21 de Diciembre, de Universidades (LOU) (actualizada varias veces desde su redacción inicial), que sustituye a la LRU de 1983, aunque el régimen jurídico aplicable a las Universidades no empieza y termina en dicha Ley Orgánica, ni tampoco en los Estatutos de cada Universidad, sino que se completa con varias disposiciones de distintos rangos normativos y que incluyen normas sobre enseñanzas universitarias (acceso a la Universidad, ordenación académica, titulaciones y enseñanzas universitarias con acceso a profesiones reguladas), normas sobre los estudiantes, el personal docente e investigador, los centros universitarios, la investigación etc. Además de todo lo anterior, falta considerar las leyes autonómicas de Universidades.

### 1.1.1. La Ley Orgánica de Universidades

La Ley Orgánica 6/2001, de 21 de Diciembre, de Universidades (LOU), con las modificaciones posteriores introducidas sobre todo por la Ley Orgánica 4/2007, de 12 de Abril (y otras) comienza exponiendo las razones sobre su necesidad, que se resumen en la necesidad de que la Universidad se adapte a varios cambios: lo que la sociedad espera de ella en el contexto de la sociedad del conocimiento[1], la autonomía de las Universidades que ayudó a que un plazo de 20 años se triplicaran las existentes, y el proceso de descentralización universitaria por el que las administraciones educativas autonómicas recibieron las transferencias de las competencias en materia de enseñanza superior.

La LOU reconoce que las Universidades desempeñan un papel nuclear en el desarrollo cultural, económico y social, y que por ello es necesario reforzar su capacidad de liderazgo y flexibilizar sus estructuras, de manera que cada una pueda desarrollar planes adecuados a sus características, pudiendo decidir su estructura de profesorado, oferta de estudios y sus procesos de gestión e innovación. A cambio las Universidades deben poder abordar lo que la sociedad exige de ellas: docencia de calidad e investigación de excelencia. Para ello la LOU pretende facilitar la movilidad de estudiantes y profesores, profundizar en la creación y transmisión del conocimiento como hilo conductor de la actividad académica, responder a los retos de la enseñanza superior no presencial gracias a las tecnologías de la información ylas comunicaciones e integrarse en el espacio universitario europeo.

Esta ley se plantea a sí misma el reto de enlazar la autonomía universitaria con la rendición de cuentas a la sociedad que la financia e impulsa, y el reto de enlazar los distintos niveles competenciales: el de la Universidad, el de las Comunidades Autónomas y el de la Administración General del Estado. También pretende la mejora de la calidad del sistema universitario, estableciendo para ello la creación de la Agencia Nacional de Evaluación de la Calidad y Acreditación. Esta agencia evaluará tanto las enseñanzas como la actividad investigadora, docente y de gestión, y los servicios y programas de las Universidades. Las enseñanzas y títulos también se regulan, y los planes de estudio serán evaluados tras su implantación.

La LOU refuerza el compromiso de los poderes públicos con promover la investigación básica aplicada en las Universidades en beneficio del interés general, y con que las innovaciones científicas y técnicas se transfieran con rapidez y eficacia al conjunto de la sociedad, teniendo en cuenta por primera vez las empresas de base tecnológica como una estructura para que las Universidades difundan y exploten sus resultados en la sociedad.

Respecto al profesorado, la LOU establece un sistema de selección más abierto, competitivo y transparente, y el desarrollo de una carrera académica equilibrada y coherente con nuevas figuras contractuales e incentivos según parámetros de calidad. Para ello establece que el PDI de las Universidades públicas estará formado por funcionarios de los cuerpos docentes universitarios y personal contratado, establece varias figuras de contratación (Ayudantes, Ayudantes Doctores, Contratados Doctores, Asociados, Visitantes y Eméritos) y deja en dos las figuras de los cuerpos docentes univer-

[1] Aquella en la que el progreso, especialmente el económico, se basa en la producción, transmisión y difusión del conocimiento a través de las TIC

sitarios (Catedráticos de Universidad y Profesores Titulares de Universidad). Aunque originalmente existe una figura de contratación adicional, la del Profesor Colaborador, la Ley Orgánica 4/2007, de 12 de abril, la elimina.

Sobre las funciones de la Universidad, la LOU dice que esta es la encargada de proporcionar el servicio público de la educación superior a través de la investigación, la docencia y el estudio, y que sus funciones principales son:

- Crear, desarrollar, transmitir y criticar la ciencia, la técnica y la cultura.
- Formar para el ejercicio de actividades profesionales que requieran la aplicación de conocimientos y métodos científicos, técnicos y para la creación artística, y también la formación a lo largo de toda la vida
- Difundir, valorar y transferir el conocimiento al servicio de la cultura, la calidad de vida y el desarrollo económico

**Enseñanzas y títulos**

La LOU señala que una misión esencial de la Universidad es la enseñanza para el ejercicio de profesiones que requieren conocimientos científicos, técnicos o artísticos. Indica que la docencia es tanto un derecho como un deber de los profesores de las Universidades, que la ejercerán con libertad de cátedra, solo con los límites establecidos en la Constitución y en las leyes y los derivados de la organización de la enseñanza en sus Universidades. La actividad y dedicación docente serán criterios relevantes para determinar la eficiencia en la actividad profesional del personal docente.

Las Universidades impartirán enseñanzas que conducirán a la obtención de títulos oficiales y válidos en toda España, y también a otros títulos. Las directrices y condiciones para la obtención de los títulos oficiales serán establecidos por el Gobierno, y para poder impartir enseñanzas conducentes a ellos las universidades deberán poseer autorización de su Comunidad Autónoma y la verificación del Consejo de Universidades de que su plan de estudios se ajusta a estas directrices y condiciones.

Las enseñanzas universitarias se estructurarán en tres ciclos, Grado, Máster y Doctorado, cuya superación dará derecho a los títulos oficiales correspondientes.

**El personal docente e investigador**

La LOU establece que las universidades podrán contratar PDI en régimen laboral según las modalidades reguladas por esa Ley (Ayudante, Profesor Ayudante Doctor, Profesor Contratado Doctor, Profesor Asociado y Profesor Visitante), así como contratar Profesores Eméritos. Salvo en el caso de Profesores Visitantes, la contratación se hará mediante concurso público, y se considerará mérito preferente el tener la acreditación para participar en los concursos de acceso a los cuerpos docentes universitarios. Son las Comunidades Autónomas las que establecerán el régimen del PDI contratado en las universidades, en los términos de la LOU y en el marco de sus competencias.

Respecto al profesorado universitario funcionario, este pertenecerá al cuerpo de Catedráticos de Universidad o al de Profesores Titulares de Universidad. El acceso a los cuerpos de funcionarios docentes universitarios exigirá la obtención de una acreditación

nacional que garantice la calidad en el proceso de selección valorando los méritos y competencias de los aspirantes. Esta acreditación se hará mediante examen de la documentación que presenten los solicitantes por parte de comisiones de al menos siete integrantes de los cuerpos de funcionarios docentes universitarios de reconocido prestigio docente e investigador.

### 1.1.2. La Ley de Ordenación del Sistema Universitario de Aragón

La Ley 5/2005, de 14 de Junio, de Ordenación del Sistema Universitario de Aragón contiene una regulación del sistema universitario de Aragón, entendido como el formado por las universidades creadas o reconocidas mediante Ley, así como los centros públicos y privados en el ámbito de la enseñanza universitaria en Aragón. Esta Ley parte del derecho estatal y de otras Leyes de la Comunidad Autónoma, y regula aquellos aspectos en los que la Comunidad Autónoma de Aragón, dadas sus competencias, puede incidir específicamente. Por ejemplo, no hace una exposición detallada sobre la normativa de los miembros de los cuerpos docentes universitarios, ya recogida en la Ley nacional, pero sí que establece los fundamentos legales para que el Gobierno de Aragón pueda reglamentar en algunos aspectos que le atañen, como en el caso del profesorado contratado.

La Universidad de Zaragoza se reconoce como el principal referente y garante del servicio público de la educación superior y la investigación y esta Ley incide especialmente en los principios relativos a su financiación. La Ley establece distintos tipos de financiación, aunque la concreción del modelo queda para un acuerdo del Gobierno de Aragón, que podrá ser periódicamente modificado según las condiciones económicas y las necesidades de la Universidad.

Esta Ley tiene también como gran objetivo la creación de la Agencia de Calidad y Prospectiva Universitaria de Aragón, que es un órgano para impulsar y desarrollar iniciativas de evaluación continuada y promoción de la calidad del sistema universitario aragonés.

## 1.2. El Espacio Europeo de Educación Superior

El proceso de Bolonia ha revolucionado la cooperación en Europa en educación superior. En la celebración del 800 aniversario de la Universidad de París, cuatro ministros de educación Europeos (de Francia, Alemania, Italia y Reino Unido) compartieron la visión de que la segmentación de la educación superior en Europa era perjudicial, lo que plasmaron el 25 de Mayo de 1998 en la declaración de la Sorbona [Allegre et al., 1998]. Una año después, en Bolonia, 30 países firman la declaración de Bolonia [Joint declaration of the European Ministers of Education, 1999], en la que formalizan la puesta en marcha de un proceso voluntario para crear el Espacio Europeo de Educación Superior (EEES).

El proceso de Bolonia nace para fortalecer la competitividad y el atractivo de la educación superior en Europa, y facilitar la movilidad de estudiantes y trabajadores

gracias a un sistema de estudios de grado y postgrado con títulos y programas fácilmente legibles. Sin embargo, el proceso ha ido ampliándose con diversas declaraciones ministeriales por los que la estructura de títulos ha pasado a un sistema de tres ciclos o se ha pasado a hacer un fuerte énfasis en los resultados del aprendizaje:

- En 2001 en Praga, se pasa a 33 países, y se expanden los objetivos: aprendizaje continuo durante toda la vida, estudiantes como participantes activos, mejora del atractivo y la competitividad del EEES.
- En 2003 en Berlín, se alcanzan los 40 países, y de nuevo crecen los objetivos: enlazar el EEES con el Espacio de Investigación Europeo, y promover el aseguramiento de la calidad. Además se crean ciertas estructuras para soportar el proceso entre las reuniones ministeriales.
- En 2005 en Bergen, se subraya la importancia de involucrar a estudiantes, instituciones de educación superior, personal académico y empleadores, así como en la mejora de la investigación, sobre todo en relación con los programas de doctorado (tercer ciclo).
- En 2007 en Londres, se llega a 46 países participantes, se evalúan los progresos realizados hasta la fecha (movilidad, estructura de grados, reconocimiento, marcos de calificación, aprendizaje continuo, aseguramiento de calidad, dimensión social) y se establecen como prioridades la movilidad, la dimensión social, la recolección de datos y la empleabilidad.
- En 2009 en Lovaina, se establecen las áreas de trabajo principales para la siguiente década, que muestran una nueva orientación del proceso hacia una aproximación más en profundidad de las reformas orientada a completar la implementación del proceso de Bolonia.
- En 2010 se celebra la conferencia Aniversario en Budapest y Viena, por los 10 años el proceso. Allí se produce el lanzamiento oficial del EEES, lo que significa que el objetivo de la declaración de Bolonia se había conseguido. Sin embargo, no todos los objetivos originales se habían conseguido, así que el proceso de Bolonia entra en una nueva fase, de consolidación y operacionalización.
- En 2012 en Bucarest, se declara que la reforma de la educación superior puede ayudar a Europa a generar crecimiento y empleos de manera sostenible, para lo que se acuerda focalizarse en tres objetivos: proporcionar educación superior a más estudiantes, equipar a los estudiantes con habilidades que les faciliten la empleabilidad e incrementar la movilidad estudiantil.

### 1.2.1. La dimensión del Espacio Europeo de Educación Superior

El informe de implementación de Bolonia en 2015 [European Commission/EACEA/Eurydice, 2015] recoge las cifras más actuales disponibles sobre la implantación del EEES en Europa, algunas de las cuales es interesante resaltar para comprender mejor su dimensión:

- En el año académico 2011-2012 hay unos 37,2 millones de estudiantes de enseñanza terciaria en el EEES, aunque el tamaño de la población estudiantil es muy variado entre los distintos países del mismo, desde los 960 de Liechtenstein hasta los casi 8 millones de Rusia. De hecho, cinco países (Rusia, Turquía, Alemania, Reino Unida y Ucrania) representan algo más del 54 % del total, y otros cuatro (Francia, Polonia, España e Italia) tienen más de 1.900.000 estudiantes, mientras que hay 18 países con menos de 200.000.

- Hay 26 países que tienen entre 11 y 100 instituciones de enseñanza superior (públicas y privadas, académicas y orientadas profesionalmente[2]), entre ellos España (82 Universidades, entre públicas y privadas, en 2015). Hay 7 países que tienen entre 101 y 200 (entre las 124 de Portugal y las 184 de Turquía) y cuatro países con más de 200: Francia, Alemania, Polonia y Rusia.

- El gasto público en educación superior es (en 2011) de un 1,32 % del PIB en la mediana de los países del EEES. El contraste entre los que más gastan (como Dinamarca, con un 2,44 %) y los que menos (como Georgia, con un 0,30 %) es llamativo. España está en ese año en el 1,29 %, un poco por debajo de la mediana del EEES.

### 1.2.2. Implantación del Espacio Europeo de Educación Superior en España

El sistema de educación español ha quedado configurado en tres ciclos denominados Grado, Máster y Doctorado, en línea con los acuerdos de la conferencia de Bergen de 2005. Todos los grados consisten de 240 créditos ECTS, que incluyen todos los tipos de aprendizaje con sus correspondientes evaluaciones y que aproximadamente corresponden con 25 horas de trabajo del estudiante. El grado es suficiente para la entrada en el mercado laboral. Un Máster puede tener entre 60 y 120 créditos ECTS, y termina con la producción y defensa pública de un proyecto o disertación final. Los másters ofrecen estudios especializados y de nivel superior, y proporcionan un valor añadido sobre los grados. Los programas de Doctorado consisten de un periodo de aprendizaje, 60 créditos que pueden ser de estudios de Máster, o actividades especialmente designadas para ello, y un periodo de investigación supervisado por un director de tesis, y duran entre 3 y 4 años, aunque se en general se puede solicitar su extensión si hay causas justificadas.

El curso 2005-2006 marca la introducción de los primeros Másters y programas de doctorado totalmente adaptados al EEES en España. Los primeros Grados totalmente adaptados comienzan en el curso 2008-2009 (163 programas de Grado en toda España).

### 1.2.3. El marco de calificaciones europeo

El *European Qualifications Framework* (EQF, marco de calificaciones europeo), es un mecanismo diseñado para traducir los títulos nacionales a un conjunto europeo

[2]Aunque esta distinción, donde existe, cada vez es menos clara y aunque puede existir una distinción formal entre estas instituciones, no la hay entre los títulos que otorgan

de descriptores comunes, facilitando así su lectura y su comparación. El núcleo del EQF es la descripción de 8 niveles de referencia[3] que describen lo que la persona sabe, entiende y es capaz de hacer (los resultados de aprendizaje):

- Nivel 1: Conocimiento básico, general, habilidad para llevar a cabo tareas simples y capacidad para trabajar o estudiar bajo supervisión directa en un contexto estructurado.
- Nivel 2: Conocimiento básico de un campo de estudio, habilidad para usar información relevante para llevar a cabo tareas y problemas rutinarios usando herramientas y reglas simples, y capacidad para trabajar o estudiar bajo supervisión con cierta autonomía.
- Nivel 3: Conocimiento de hechos, principios, procesos y conceptos generales de un campo de estudio, un conjunto de habilidades cognitivas y prácticas para la resolución de problemas y tareas seleccionando y aplicando métodos, herramientas, materiales e información básicos, y la capacidad de tomar responsabilidades para completar tareas en un campo de trabajo adaptando el comportamiento a distintas circunstancias.
- Nivel 4: Conocimiento de un contexto amplio de un campo de trabajo, habilidades prácticas y cognitivas para generar soluciones a problemas específicos en un campo de trabajo, la competencia para auto-gestionarse en contextos de trabajo que son normalmente predecibles pero sujetos al cambio, y la capacidad para supervisar trabajo rutinario de otros.
- Nivel 5: Conocimiento comprensivo y especializado en un campo de estudio, sabiendo los límites de ese conocimiento, la habilidad de desarrollar soluciones creativas a problemas abstractos y la competencia para gestionar y supervisar en contextos donde hay cambios impredecibles, así como a revisar y mejorar las prestaciones de uno mismo y otros.
- Nivel 6: Conocimiento avanzado de un campo, incluyendo la comprensión crítica de principios y teorías, habilidades avanzadas demostrando la maestría e innovación necesarias para resolver problemas complejos e impredecibles en un campo de trabajo y la competencia para gestionar actividades y proyectos complejos, tomar decisiones en entornos impredecibles y gestionar individuos y grupos.
- Nivel 7: Conocimiento altamente especializado, parte del cuál está en la frontera del conocimiento en un campo, comprensión crítica de los problemas en el conocimiento de ese campo y de las interfaces con otros campos, la habilidad para resolver problemas especializados necesarios en la investigación y/o en la innovación y la habilidad para gestionar y transformar contextos complejos, impredecible y que necesitan nuevas aproximaciones estratégicas, así como la capacidad para contribuir al conocimiento y la práctica de la profesión.

[3]Su definición completa está en `https://ec.europa.eu/ploteus/content/descriptors-page`

- Nivel 8: Conocimiento en la frontera más avanzada de un campo de trabajo y de la interfaz con otros campos, las habilidades más avanzadas y especializadas, incluyendo la síntesis y la evaluación, necesarias para resolver problemas críticos en investigación y/o innovación y extender el conocimiento y la práctica profesional actuales, y la habilidad para demostrar autoridad, innovación, autonomía, integridad profesional y académica, y el compromiso sostenido para el desarrollo de nuevas ideas o procesos en la frontera de un contexto de trabajo, incluyendo la investigación.

La Figura 1.1 muestra las equivalencias entre los títulos en España y los niveles del EQF. La enseñanza universitaria empieza a partir del nivel 6, con el grado que corresponde con el nivel 6[4], el máster con el nivel 7 y el doctorado con el nivel 8.

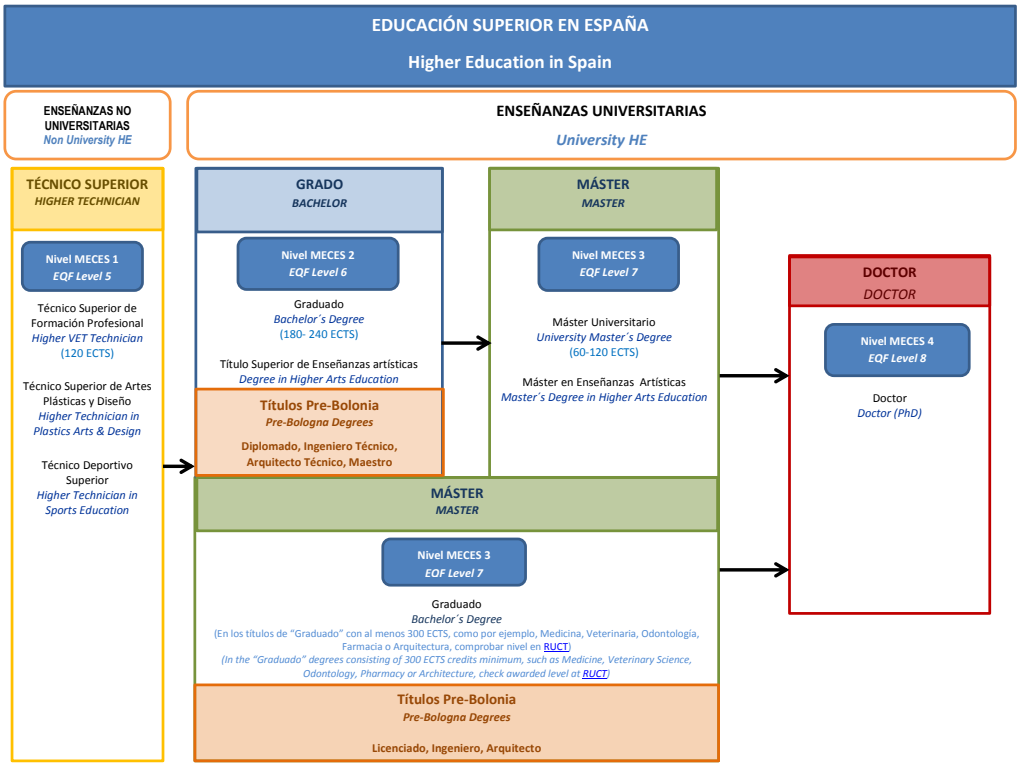



Figura 1.1: Marco Español de Cualificaciones para la Educación Superior (MECES) y equivalencia con el EQF. Tomado de `https://ec.europa.eu/ploteus/sites/eac-eqf/files/cuadro-meces.pdf`

### 1.2.4. Objetivos para el periodo 2015-2020

La última conferencia ministerial del EEES hasta la fecha ha tenido lugar en 2015, en Yerevan [EHEA ministerial conference, 2015]. Allí se ha reconocido que las reformas

[4]Salvo en aquellos grados con al menos 300 ECTS, como medicina o arquitectura, que pueden ser equivalentes al nivel 7.

de Bolonia han ayudado a estudiantes y graduados a moverse dentro del EEES con el reconocimiento de sus cualificaciones y periodos de estudio, que les han proporcionado los conocimientos, habilidades y competencias necesarios para continuar sus estudios o entrar en el mercado laboral de Europa. Sin embargo, las reformas estructurales no se han implementado de manera uniforme, y las herramientas han sido a veces usadas incorrectamente, o de manera superficial o burocrática, y por tanto es necesario continuar con la mejora de los sistemas de educación superior.

De cara al 2020, se quiere conseguir que todos los países que forman el EEES confíen mutuamente en sus sistemas de educación superior, que el reconocimiento automático de las cualificaciones sea una realidad para que estudiantes y graduados puedan moverse con facilidad, y que la educación superior contribuya de manera efectiva a construir sociedades inclusivas, basadas en valores democráticos y derechos humanos, donde las oportunidades educativas proporcionen competencias y habilidades necesarias para la ciudadanía Europea, para la innovación y para el empleo. Los objetivos principales son:

- Mejorar la calidad y la relevancia del aprendizaje y la enseñanza: promover la innovación pedagógica en entornos de aprendizaje centrados en los estudiantes y con todos los beneficios potenciales de las tecnologías digitales. Promover un enlace más fuerte entre la enseñanza, el aprendizaje y la investigación en todos los niveles de estudio, e incentivar a instituciones, profesores y estudiantes para intensificar las actividades que desarrollen la creatividad, la innovación y el emprendimiento. Todo esto soportado por descripciones transparentes de los resultados de aprendizaje y cargas de trabajo, rutas de aprendizaje flexibles y métodos adecuados de enseñanza y evaluación.

- Potenciar la empleabilidad de los graduados a través de sus vidas laborales, considerando un mercado laboral cambiante, caracterizado por los desarrollos tecnológicos y la aparición de nuevos perfiles laborales. Asegurar que al final de los ciclos de estudio los graduados poseen competencias adecuadas para su empleabilidad inmediata, y también para que puedan después desarrollar las nuevas competencias que necesitarán más adelante.

- Hacer que los sistemas de educación superior sean más inclusivos, facilitando la integración de inmigrantes, teniendo en cuenta los cambios demográficos y la necesidad del aprendizaje continuo a lo largo de la vida laboral, mejorando el equilibrio de género y proporcionando oportunidades a los estudiantes de entornos más desfavorecidos.

- Implementar reformas estructurales acordadas como requisito para el éxito a largor plazo del EEES: una estructura de grados y sistema de créditos comunes, estándares y guías de aseguramiento de la calidad comunes y la cooperación para programas de movilidad y conjuntos. Es necesario desarrollar políticas más efectivas para el reconocimiento de los créditos obtenidos en el extranjero y del aprendizaje anterior. Además es necesario que todos los países tomen estas medidas, puesto que la no implementación de algunas en algunos de ellos miman la credibilidad y la funcionalidad del EEES en su conjunto.

## 1.3. La Universidad de Zaragoza

Los Estatutos de la Universidad de Zaragoza, aprobados por Decreto 1/2004, de 13 de Enero, del Gobierno de Aragón y modificados por Decreto 27/2011, de 8 de Febrero, describen y regulan, dentro de sus competencias, la naturaleza y fines de la misma, su estructura (departamentos, facultades y escuelas, institutos de investigación y otros), su gobierno y representación, la docencia e investigación, la comunidad universitaria (personal docente e investigador (PDI), estudiantes y personal de administración y servicios), servicios e infraestructuras y su régimen económico y financiero.

La Universidad de Zaragoza es una administración pública con personalidad jurídica, autonomía académica, económica, financiera y de gobierno para ejercer el servicio público de la educación superior y, por tanto, al servicio de la sociedad. Entre sus fines principales se encuentran la transmisión de conocimientos y formación en educación superior, la creación, mantenimiento y crítica del saber en la ciencia, la cultura, la técnica y las artes, la formación de profesionales cualificados, el fomento y difusión de la cultura, la promoción de la transferencia y aplicación de conocimientos en la innovación, progreso y bienestar de la sociedad, el fomento de su proyección externa, especialmente en el marco del EEES, la Investigación y Latinoamérica, el fomento de un marco de pensamiento para que derechos humanos, solidaridad inter-generacional, desarrollo sostenible y paz sean objeto de investigación, formación y difusión, la promoción del desarrollo integral de las personas y la aceptación, defensa y promoción de los principios y valores democráticos y constitucionales.

Respecto a las estructuras principales de la Universidad, los departamentos son las unidades de docencia e investigación que coordinan las enseñanzas en varios centros y ámbitos del conocimiento (las áreas de conocimiento de su PDI), y que apoyan las actividades docentes e investigadores de los profesores; las facultades y escuelas organizan las enseñanzas de grado y los másteres oficiales y los institutos universitarios de investigación son los centros dedicados a investigación, desarrollo, asesoramiento e innovación científica, técnica y cultural, y la creación artística.

Con los datos disponibles en el año 2016[5] (algunos de ellos son de 2015 y otros de 2014), la comunidad universitaria de la Universidad de Zaragoza está formada por 40857 miembros, repartidos en 56 departamentos, 18 centros propios y 5 centros adscritos, a los que hay que sumar 5 institutos de investigación propios, 1 adscrito, 3 mixtos y 3 centros de investigación. La oferta académica incluye 54 grados, 55 másteres universitarios, 43 programas de doctorado y 93 estudios propios.

Respecto a la investigación y transferencia, existen 214 grupos de investigación reconocidos a los que pertenecen 2804 investigadores, hay 55 cátedras institucionales y de empresa y se han puesto en marcha un total de 32 empresas *spin off* y *start up*.

La Universidad de Zaragoza cuenta con un presupuesto total de 246,2 millones de euros (año 2015).

[5] `https://www.unizar.es/institucion/conoce-la-universidad/datos-basicos`

### 1.3.1. Breve historia

Como se recoge en Universidad de Zaragoza [2011], la Universidad de Zaragoza tiene su origen en un estudio de Artes creado por la iglesia en el siglo XII y elevado a la categoría de "Universitas magistrorum" en 1474 por el papa Sixto IV. Sin embargo, no es hasta Septiembre de 1542 cuando el emperador Carlos V firma un privilegio que eleva el estudio de Artes al rango de "Universidad general de todas las ciencias", donde se cursarían estudios de Teología, Derecho Canónico y Civil, Medicina y Filosofía. En 1554 el papa Julio III aprueba su fundación a través de una bula, y es por esto que la Universidad de Zaragoza es la única en España que tiene en su sello la imagen de San Pedro. Pedro Cerbuna en 1583 aporta los medios económicos necesarios para la nueva universidad, que se inauguraría en 1583.

Tras la Primera Guerra Mundial, la Universidad se renueva en profundidad, y en 1921 se aprueban unos estatutos autónomos y se empieza a impartir Doctorados. En los años 70, se incorporan nuevas enseñanzas y se expande la Universidad a otras localidades. En los ochenta se recupera la autonomía universitaria, y la Universidad de Zaragoza se adapta al ámbito de la Comunidad Autónoma de Aragón, aprobando estatutos en los años 1985, 2004 y 2011 y creando facultados en Huesca y Teruel. El siglo XXI está marcado por la adaptación al Espacio Europeo de Educación Superior, siendo el 2008/2009 el primer curso en que este se pone en marcha en la Universidad.

En sus 500 años la Universidad ha tenido más de 150 rectores, ha acogido al botánico y economista Ignacio de Asso, al primer director de la Real Academia de la Historia Montiano o al Premio Nobel Santiago Ramón y Cajal entre muchos otros, y ha concedido el Doctorado Honoris Causa (su máxima condecoración) a personajes como Luis Buñuel, Rigoberta Menchú o José Antonio Labordeta.

### 1.3.2. La Escuela de Ingeniería y Arquitectura

La Escuela de Ingeniería y Arquitectura de la Universidad de Zaragoza procede de la integración de dos facultades, el Centro Politécnico Superior (CPS) y la Escuela Universitaria de Ingeniería Técnica Industrial de Zaragoza (EUITIZ), proceso que se impulsa en la primavera de 2009.

El origen lo podemos encontrar en la la Escuela Técnica Superior de Ingenieros Industriales (ETSII) de Zaragoza, creada en 1974, donde se impartían inicialmente los estudios de Ingeniería Industrial (especialidades en Ingeniería Mecánica e Ingeniería Eléctrica), y en la Escuela Universitaria de Ingeniería Técnica Industrial de Zaragoza (EUITIZ), que en 1972 había sido integrada con la de Logroño. La ETSII tenía su sede en el edificio Interfacultades del campus universitario de la plaza de San Francisco, mientras que la EUITIZ estaba en dos edificios de la calle Corona de Aragón, muy cerca de este campus.

En 1984 cambian los planes de estudios y, por ejemplo, aparecen dos intensificaciones de la Ingeniería Industrial en la especialidad eléctrica, en Electrotecnia y en Electrónica Informática y Control, y en Diseño de Máquinas, Construcciones Industriales y Calor y Fluidos en la especialidad mecánica. En 1986 la ETSII se traslada al barrio del Actur y se intensifican las tareas de I+D y de cooperación con empresas. En 1988 se define un plan estratégico para la ETSII, que aconseja la creación de un Centro

Politécnico Superior, algo que se produce en Agosto de 1989. El objetivo es consolidar y ampliar el centro, y dotarlo de una estructura y recursos suficientes para ampliar la oferta en titulaciones de ingeniería y profundizar en las tareas de investigación y relación con las empresas del entorno. Esto permite empezar a impartir Ingeniería de Telecomunicaciones en 1990, Ingeniería Informática en 1992 e Ingeniería Química en 1994. En 1998 se pone la primera primera del nuevo edificio para la EUITIZ, construido cerca del Centro Politécnico Superior en lo que ahora se denomina campus Río Ebro.

El siguiente gran cambio llega con la adaptación al Espacio Europeo de Educación Superior, con la división entre grados y másteres que reserva para estos últimos el máximo nivel de atribuciones. Esto desdibuja las razones que tradicionalmente habían justificado la existencia de dos clases de centros (unos para ingenerías y arquitecturas técnicas y otros para "superiores") y conlleva la integración señalada al principio de esta sección.

Actualmente en la EINA hay unos 6000 estudiantes matriculados y pertenecen a ella unos 650 profesores. Se pueden cursar 36 titulaciones entre grados, másteres oficiales, estudios propios y titulaciones de ingeniería en fase de extinción.

### 1.3.3. El Departamento de Informática e Ingeniería de Sistemas

En 1987, la Universidad de Zaragoza pasa de una organización interna basada en Cátedras a una basada en Departamentos. En este proceso se constituye, entre otros departamentos, el de Ingeniería Eléctrica e Informática, que incluye las áreas de conocimiento de Electrónica, Ingeniería de Sistemas y Automática, Ingeniería Electrónica, Lenguajes y Sistemas Informáticos y Tecnología Electrónica. Los años siguientes se van incorporando nuevas áreas de conocimiento, como la de Arquitectura y Tecnología de Computadores, Ciencias de la Computación e Inteligencia Artificial, Ingeniería Telemática y Teoría de la Señal y Comunicaciones.

El importante número de áreas involucradas y la implantación de los planes de estudio de Ingeniería de Telecomunicación e Ingeniería Informática, llevan a solicitar la división del departamento en 1992, proceso que concluye en Abril de 1995 con la aprobación de la Junta de Gobierno de la Universidad de Zaragoza la división del Departamento de Ingeniería Eléctrica e Informática en los de Ingeniería Eléctrica, Electrónica y Comunicaciones y el de Informática e Ingeniería de Sistemas.

El Departamento de Informática e Ingeniería de Sistemas (DIIS) así formado incluye las áreas de conocimiento de Arquitectura y Tecnología de Computadores, Ciencias de la Computación e Inteligencia Artificial, Ingeniería de Sistemas y Automática y Lenguajes y Sistemas Informáticos, que mantiene hasta el día de hoy.

Actualmente el DIIS tiene 148 miembros de personal docente e investigador (entre profesores permanentes, no permanentes, investigadores y becarios de investigación) y 10 miembros de personal de administración y servicios. Su sede está en la Escuela de Ingeniería y Arquitectura, pero el PDI del DIIS imparte docencia en 10 centros de la Universidad de Zaragoza, repartidos en 28 titulaciones de grado y 5 másteres oficiales [DIIS, 2015].

### 1.3.4. El Área de Lenguajes y Sistemas Informáticos

Las áreas de conocimiento están definidas en el artículo 71 de la LOU, como "aquellos campos del saber caracterizados por la homogeneidad de su objeto de conocimiento, una común tradición histórica y la existencia de comunidades de profesores e investigadores, nacionales o internacionales."

El área de Lenguajes y Sistemas Informáticos (LSI) en la Universidad de Zaragoza es parte del DIIS, y los profesores adscritos a ella imparten asignaturas en 8 centros de esta universidad [DIIS, 2015]. La amplitud de las materias que engloba actualmente este área de conocimiento se aprecia en la Tabla 1.1, creada a partir de datos del plan de ordenación docente del año 2015/2016.

Tabla 1.1: Asignaturas de LSI en la Universidad de Zaragoza

| **Centro** | **Titulación** | **Asignatura** |
|---|---|---|
| Escuela de Ingeniería y Arquitectura (Z) | Grado Ingeniería Eléctrica | Informática |
| | Grado Ingeniería Electrónica y Automática | Fundamentos de Informática |
| | Grado Ingeniería Mecánica | Fundamentos de Informática |
| | Grado Ingeniería Química | Fundamentos de Informática |
| | Grado Tecnologías Industriales | Fundamentos de Informática |
| | Grado Tecnologías y Servicios de Comunicación | Fundamentos de Informática |
| | | Programación de Redes y Servicios |
| | | Análisis y diseño de software |
| | Grado Ingeniería Informática | Programación 1 |
| | | Programación 2 |
| | | Prog. Sist. Concurrentes |
| | | Teoria de la Computación |
| | | Est. Dat. Alg. |
| | | Interacción persona Ordenador |
| | | Tecnología de Programación |
| | | Bases de Datos |
| | | Ingeniería del Software |
| | | Inteligencia Artificial |
| | | Sistemas de Información |
| | | Sistemas Distribuidos |
| | | Proyectos |
| | | Seguridad Informática |
| | | Sistemas de Información Distribuidos |
| | | Bioinformática |
| | | Metodologías ágiles y calidad |
| | | Ingeniería de Requisitos |
| | | Arquitecturas Software |
| | | Verificación y Validación |
| | | Ingeniería Web |
| | | Sistemas Legados |
| | | Laboratorio de Ing. del Sw |
| | | Gestión de Proyectos Software |
| | | Metodologías ágiles y calidad |

Tabla 1.1: Asignaturas de LSI en la Universidad de Zaragoza

| **Centro** | **Titulación** | **Asignatura** |
|---|---|---|
| | | Sistemas y Tecnologías web |
| | | Algoritmia básica |
| | | Procesadores de Lenguajes |
| | | Aprendizaje Automático |
| | | Algoritmia para problemas difíciles |
| | | Recuperación de Información |
| | | Informática Gráfica |
| | | Visión por Computador |
| | | Videojuegos |
| | | Bioinformática |
| | | Sistemas de Información 2 |
| | | Bases de datos 2 |
| | | Almacenes y Minería de Datos |
| | | Sistemas de Ayuda a la Toma de Decisiones |
| | | Sistemas de Información Distribuidos |
| | | Laboratorio de Sistemas de Información |
| | | Diseño Centrado Usuario. Diseño para la Multimedia |
| | Grado Diseño Industrial | INFORMÁTICA(T) |
| | | Comu. Multimedia |
| | | Composición y Edición de Imágenes |
| | | Entornos 3D interactivos |
| | Grado Arquitectura | Informática(A) |
| | Máster Universitario Ingeniería Informática | Administración y dirección estratégica de empresas |
| | | Gestión de la innovación en TI |
| | | Computación Gráfica-Entornos Inmersivos-Multimedia |
| | | Sistemas Inteligentes |
| | | Manipulación y Análisis de Grandes Volúmenes de Datos |
| | | Sistemas Empotrados Ubicuos |
| | | Calidad en el Desarrollo de Software, Servicios e Infraestructuras TI |
| | | Computación de Altas Prestaciones |
| | | Tecnologías y Modelos para el Desarrollo de Aplicaciones Distribuidas |
| | | Sistemas BioInspirados |
| Facultad de Ciencias Sociales y del Trabajo (Z) | | |
| | Grado Trabajo Social | Tecnologías |
| | Grado Relaciones Laborales y Recursos Humanos | Tecnologías |
| | Máster Secundaria | Diseño curricular de matemáticas, informática y tecnología |
| | | Fundamentos de diseño instruccional de matemáticas, informática y tecnología |
| | | Contenidos disciplinares de informática |
| | | Diseño y organización de actividades de aprendizaje de informática y tecnología |

Tabla 1.1: Asignaturas de LSI en la Universidad de Zaragoza

| **Centro** | **Titulación** | **Asignatura** |
|---|---|---|
| | | Evaluación, innovación e investigación docente |
| | | Prácticum II y III |
| Facultad de Economía y Empresa (Z) | Grado en Marketing e Investigación de Mercados | Las TIC y su aplicación al Marketing |
| | | Sistemas de información y bases de datos |
| | Grado en Administración y Dirección de Empresas | Las TIC en la empresa |
| Escuela Universitaria Politécnica de Teruel (T) | Grado Ingeniería Informática | Programación I |
| | | Teoría de la Computación |
| | | Programación de Sist. Conc. Y Dist. |
| | | Estructuras de Datos y Algoritmos |
| | | Ingeniería del Software |
| | | Inteligencia Artificial |
| | | Sistemas de Información |
| | | Sistemas de Ayuda a la Toma de Decisiones |
| | | Seguridad Informática |
| | | Almacenes y Minería de Datos |
| | | Sistemas Legados |
| | | Ingeniería Web |
| | | Programación II |
| | | Interacción persona Ordenador |
| | | Tecnología de Programación |
| | | Bases de Datos |
| | | Proyecto Software |
| | | Bases de Datos II |
| | | Sists. de Información II |
| | | Sistemas y Tecnologías web |
| | | Diseño centrado en el usuario. Diseño para la multimedia |
| | | Comercio Electrónico |
| Escuela Politécnica Superior (H) | Grado en Ingeniería Agroalimentaria y del Medio Rural | Informática |
| Facultad de Ciencias de la Salud y del Deporte (H) | Grado en Odontología | Informática aplicada a la odontología |
| | Grado de Nutrición Humana y Dietética | Tecnologías de la información y comunicación en ciencias de la salud |
| Facultad de Empresa y Gestión Pública (H) | Grado en Gestión y Administración Pública | Informática de gestión |

Tabla 1.1: Asignaturas de LSI en la Universidad de Zaragoza

| **Centro** | **Titulación** | **Asignatura** |
|---|---|---|
| Campus Virtual | Asignaturas Virtuales - G9 | Implementación de una tienda web sencilla mediante .net |

# Capítulo 2

# Contexto profesional

En la actualidad vivimos en una sociedad de la información, caracterizada porque la información es un activo crítico en la economía, la política y la cultura. Esto es posible por la existencia de las tecnologías de la información y las comunicaciones (TIC), que han posibilitado una explosión en la cantidad y disponibilidad de la información y han alterado en mayor menor grado casi todas las actividades humanas.

Algunos indicadores del crecimiento e impacto de las TIC se recogen en el informe anual de la *International Telecommunication Union* [ITU, 2015] y reflejan el gran alcance de este sector: por ejemplo, el número de hogares con conexión a Internet en el Mundo a pasado de un 18,4 % en 2005 a un 49 % en 2015, mientras que el número de personas que usan Internet ha pasado de unos 1.024 millones en 2005 a 3.207 millones en 2015. Este mismo informe recoge un indicador, el *ICT Development Index* (IDI), que se usa para comparar el desarrollo en TIC entre países, y a lo largo del tiempo, y hacer una clasificación en la que España aparece en el puesto 26 en 2015 (estaba en el puesto 30 en 2010), ligeramente por encima de la media de la Europa más desarrollada.

La informática es la disciplina que sostiene la existencia de las TIC, que a su vez son el soporte de la sociedad de la información y el contexto profesional en el que se tienen que desenvolver los profesionales de la informática. Es por eso, que la sección 2.1 hace un breve e impreciso intento de al menos situarla, dada la dificultad de definirla. La sección 2.2 aporta algunas pinceladas sobre los elementos principales de la ingeniería, puesto que es la ingeniería informática el contexto de este proyecto docente. Dado el carácter profesional del contexto aportado en este capítulo, la sección 2.3 hace un breve repaso sobre la regulación legal de la ingeniería informática en España y en otros países. Finalmente la sección 2.4 aporta algunas cifras significativas sobre el sector TIC tanto en Europa como en España y en Aragón.

## 2.1. La informática

La informática es una disciplina que podríamos considerar que se origina en un artículo de 1936 escrito por el científico británico Alan Turing, que describía un computador hipotético. Anticipando la combinación de teoría e ingeniería de este campo, Turing diseñó y construyó algunos de los primeros computadores.

La informática es complicada de definir, por su relativa juventud, por su alto ritmo

de cambio, por la amplia diferencia de escala entre los fenómenos que trata, y por tener sus raíces en varias disciplinas (por ejemplo, distintas áreas de las matemáticas, física e ingeniería eléctrica) que a veces proponen formas distintas de considerar los mismos temas. Denning et al. [1989] recoge un conjunto de definiciones de informática dadas por diversos autores y propone una que es lo bastante amplia y directa como para resultar de utilidad: "...es un estudio sistemático de los procesos algorítmicos que describen y transforman información: su teoría, análisis, diseño, eficiencia, implementación y aplicación. La cuestión fundamental es ¿qué puede ser (eficientemente) automatizado?".

En 2012 se envía un cuestionario sobre los estudios de doctorado en informática en Europa a 86 experimentados profesores europeos de 28 países, de los que 68 contestan [Nagl, 2013]. Aunque no es el elemento principal del cuestionario, se les pregunta sobre lo que ellos consideran que caracteriza a la informática, y algunas de sus respuestas son muy interesantes porque dan una idea de la amplitud de esta disciplina y las distintas perspectivas bajo las que se puede considerar:

- La informática es analítica, y busca comprender y analizar los sistemas de comunicación y procesamiento, naturales e imaginados, también incluye el estudio de los artefactos y tiene una vibrante industria alrededor.

- La informática es constructiva, en su mayor parte consiste en construir algo, un sistema, un diseño no trivial, una prueba. Debería ser formal, pero las soluciones prácticas, la experiencia y la intuición desempeñan un papel. Los resultados teóricos deberían discutir la aplicabilidad, y los resultados prácticos ser formales donde sea posible. No es construir una solución detrás de otra, es una discusión intelectual sobre ideas, variedad de soluciones, aprendizaje y mejora.

- La informática tiene un ciclo de investigación específico. Contiene aspectos matemáticos, de ingeniería, de ciencias naturales y hoy en día también sociales. El núcleo es el pensamiento algorítmico y la solución de problemas constructiva.

- La informática tiene impacto, puede afectar profundamente la forma en que la gente vive, trabaja y se entretiene. Este recorrido tan corto entre la informática como disciplina científica y la gran escala de sus efectos hace que la informática sea atractiva para los estudiantes más brillantes. Deberíamos resaltar el potencial único de innovación que tiene la informática para preservar su atractivo.

- La informática es interdisciplinar. Es un 55 % ingeniería, un 25 % matemáticas y ciencias naturales, un 10 % administración de empresas y un 10 % artes y similares.

## 2.2. La ingeniería

Un abogado, un sacerdote y un ingeniero van a ser guillotinados. El cuello del abogado se apoya en la guillotina, pero cuando el verdugo tira de la palanca, la hoja se bloquea y no cae. Al abogado le dicen que debe haber sido una forma de justicia superior y que se le perdona la vida. Después es el turno del sacerdote, pero de nuevo

se bloquea la hoja. Se asume la intervención de poderes superiores y se le perdona la vida también. Cuando le llega el turno al ingeniero, apoya la nuca en la guillotina, mira hacia arriba y justo antes de que el verdugo tire de la palanca, el ingeniero dice —espera, veo cuál es el problema, y sé cómo arreglarlo [Petroski, 2011, p. 175].

El origen de la palabra ingeniero, parece que está en el latín *ingenium*, que significa un talento, habilidad o disposición naturales y, en este contexto, el ingenio para crear máquinas o dispositivos. La esencia de la ingeniería es el diseño; sin diseño, no habría ingeniería y el resto de las actividades de la ingeniería están al servicio del diseño [Koen, 2003, p. 28] [Petroski, 2011, p. 66]. El diseño es un proceso de síntesis, de creación, que es lo opuesto al análisis que es de comprensión.

Herbert A. Simon describió la ingeniería como la "ciencia de lo artificial". Rogers [1983, p. 51] la define como "la práctica de organizar el diseño y construcción de cualquier artificio que transforma el mundo físico alrededor de nosotros para alcanzar alguna necesidad reconocida" y señala que un ingeniero usará distintas tecnologías para lograrlo Hughes [2009] indica que "el método de la ingeniería —es decir, el proceso de diseño— tiene un fin práctico explícito [...] Para ponerlo crudamente, el proceso de diseño es análogo al método científico y una necesidad en ingeniería —el problema a resolver— es análoga a una hipótesis científica". Esta analogía aparece también en [Rogers, 1983, p. 55] que señala la precisión que es a menudo un requisito en la ciencia, no es típicamente alcanzable en la tecnología, porque el científico suele trabajar con un sistema bien definido y con un número limitado de variables, mientras que el tecnólogo debe trabajar con modelos "a escala" aproximados del dispositivo que está construyendo, y las teorías que desarrolle serán solo aplicables en estos modelos, puesto que no tendrá forma de saber si esos resultados pueden extrapolarse al dispositivo final, ni conocerá con ninguna precisión las condiciones en las que ese dispositivo final operará en la práctica.

Koen [2003, p. 28] define el método de la ingeniería como "el uso de heurísticas para causar el mejor cambio en una situación pobremente comprendida dentro de los recursos disponibles". Esta amplia definición, centrada en el proceso y no en el resultado, y ciertamente aplicable a muchos tipos de actividades, ayuda a situar el trabajo de los ingenieros: cuando una situación requiere un cambio, no todos los resultados son igualmente deseables, queremos el mejor cambio posible, tenemos recursos limitados y el conocimiento sobre el sistema antes, durante y después del cambio es incompleto, inconsistente o inabarcable durante el tiempo disponible para el problema, entonces hace falta un ingeniero[1]. Koen [2003, p. 57] va incluso algo más allá y propone que la principal regla del método de la ingeniería es "en cada ocasión, elegir la mejor heurística entre lo que el ingeniero tenga como la vanguardia de la mejor práctica de la ingeniería".

Que la heurística sea sugerida como la base del método de la ingeniería hará sin duda fruncir algunos ceños. Al fin y al cabo, las heurísticas no garantizan soluciones, pueden contradecirse entre ellas, y suelen abarcar solo el contexto inmediato donde se aplican. Por otra parte, reducen el tiempo de búsqueda para resolver problemas y

[1]Por supuesto, hemos visto en la sección 2.3 que ser ingeniero tiene distintos requisitos legales en distintos países. Pero esos no nos dicen mucho sobre el trabajo de los ingenieros. Tampoco es de gran ayuda para distinguir lo que es la ingeniería que muchos profesionales se auto-denominan a sí mismo ingenieros sin tener la titulación o credenciales requeridas legalmente para ello.

las búsquedas son esenciales en la resolución de problemas, no solo para buscar una solución, sino como un proceso que permite obtener información sobre la estructura del problema que será valiosa para descubrir una solución [Simon, 1996, p. 127]. De hecho, todo lo que se ha aprendido sobre cómo las personas resuelven problemas apunta a la conclusión de que esta involucra nada más que varias combinaciones de prueba, error y selectividad basada en heurísticas [Simon, 1996, p. 195]. Petroski [2011, p. 317-318] lo describe muy bien:

> la sabiduría convencional sostiene que la ingeniería no es sino ciencia aplicada [...] Esto está, de hecho, lejos de la verdad, y el diseño de una estructura, máquina o sistema de ingeniería comienza típicamente no con fórmulas matemáticas o principios científicos, sino con la concepción y el bosquejo de una idea. Solo cuando la idea está articulada en dibujos o palabras, las herramientas de las matemáticas y los principios de la ciencia se pueden usar para responder a preguntas específicas que convierten el diseño conceptual en uno detallado.
>
> Más a menudo que no, el diseño resultante tiene tal complejidad de detalle que no se puede traducir directamente en ecuaciones o fórmulas, ni sus partes se pueden compartimentalizar en principios científicos simples. Juicios y conjeturas son necesarias para manipular y reorganizar los componentes del diseño, modelos o prototipos que pueden entonces ser analizados y probados para ver si son conformes a los requisitos del problema inicial. Si no lo son, y teniendo en cuenta lo lejos que están, el diseño puede ser modificado por nuevos juicios y conjeturas y una nueva ronda de análisis y prueba puede llevarse a cabo. Este es el método iterativo de prueba y error [...] En diseño moderno, este método puede automatizarse con un computador, pero no ser totalmente reemplazado por este.

## 2.3. La regulación de la ingeniería informática

Tras la implantación en España del Espacio Europeo de Educación Superior, la ingeniería informática queda fuera del Real Decreto 1837/2008, de 8 de Noviembre, por el que se incorpora al ordenamiento jurídico español normativa europea relativa al reconocimiento de cualificaciones profesionales.

En 2016 la situación sigue más o menos igual, aunque se han aprobado algunas regulaciones relativas a la informática. La proposición no de ley 161/002878, aprobada el 11 de Febrero de 2015 con el apoyo de todos los grupos parlamentarios, insta al Gobierno a adoptar las medidas necesarias para que la ingeniería informática tenga el mismo nivel de definición académico que el resto de ingenierías, aunque por ahora todavía no se han producido avances significativos a este respecto. Hoy en día en España, las competencias establecidas en la resolución 12977/2009, de 8 de Junio, y listadas en la Sección 3.1 definen indirectamente lo que debe saber, y saber hacer, un ingeniero informático.

El acceso de los ingenieros a la profesión se hace de distintas maneras en el mundo[2].

[2] `https://www.icai.es/articulo-revista/el-acceso-de-los-ingenieros-al-ejercicio-de-la-profesion-`

En general hay tres modelos: los ingenieros profesionales con licencia, los ingenieros certificados o acreditados profesionalmente y el modelo basado exclusivamente en el título universitario.

El modelo basado en la licencia, con Estados Unidos como principal referente[3], asume que además de unos estudios superiores acreditados, es necesario un aprendizaje tutorizado y práctica documentada y compulsada, y exámenes en su caso, para poder hacer un pleno ejercicio de la profesión con garantías para la sociedad. Es el modelo más cercano al de los antiguos gremios profesionales. Sus características habituales son que el título de ingeniero profesional está protegido por la ley, el título académico típicamente no, la profesión está regulada y hay atribuciones reservadas exclusivamente a los ingenieros profesionales, la colegiación es obligatoria, la especialización se adquiere por la práctica tutorada y certificada más que por los estudios, y existe un código ético estricto al que se comprometen los ingenieros profesionales.

En el modelo basado en la certificación y el registro, con el Reino Unido como ejemplo principal[4], la profesión no está regulada y por tanto no existen atribuciones profesionales ni colegios propiamente dichos (aunque sí que hay asociaciones profesionales reconocidas por el estado). La competencia profesional se certifica y registra, en distintos niveles y de manera voluntaria, por una entidad acreditada para ello e independiente de las universidades. La certificación supone el reconocimiento de unas competencias y/o una especialización por encima de la del ingeniero no certificado. No se puede acceder a la certificación si no se tiene un título profesional reconocido y que dé este acceso.

En el modelo basado únicamente en el título universitario, la profesión está regulada, existen colegios profesionales y los ingenieros deben colegiarse para realizar determinados trabajos en los que asumen atribuciones. El único requisito para la colegiación es un título universitario reconocido, no se necesita ninguna prueba adicional. Este modelo se da en España, en la mayoría de los países de América Latina y en algunos de África.

En Europa la situación es muy variada. En general la profesión no está demasiado regulada (Alemania, Francia), o no lo está en absoluto (Reino Unido, Holanda, Bélgica, Suecia, Finlandia), aunque por ejemplo en Italia o en Portugal sí que se encuentra regulada. En cualquier caso, lo habitual en los países europeos es que la ingeniería informática no se trate como una excepción, y tenga la misma regulación que el resto de ingenierías.

## 2.4. El sector TIC

### En Europa

En su documento "Estrategia de I+D e innovación para las TIC en Europa: una apuesta de futuro" (documento COM número 116 de 2009, versión final)[5], la Comisión de las Comunidades Europeas señala que el mercado mundial de las TIC es de 2 billones

[3]También se da en Canadá, la India y Japón entre otros países.

[4]También se usa en Hong Kong, Malasia, Singapur, Nueva Zelanda y Australia.

[5]`http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2009:0116:FIN:ES:PDF`

de euros, que crece al 4 % anual, y que representan un 30 % de la I+D mundial, siendo Europa un 34 % del total de ese mercado. También señala que las TIC son esenciales para afrontar algunos de los desafíos de la sociedad europea, como el envejecimiento de la población o la sostenibilidad de la asistencia sanitaria, la seguridad, la privacidad y el decremento de emisiones de carbono y la salida de la crisis económica.

En 2012 el sector TIC en la Unión Europea emplea a un 2,76 % de la población ocupada en Europa (unos 6,18 millones de personas) y contribuye a un 3,99 % de su PIB, siendo el 92,27 % de esto debido a las empresas de servicios de TIC (ventas y reparaciones de equipamiento informático, electrónico y de comunicaciones, publicación de software, telecomunicaciones, programación de software, consultoría, procesamiento de datos, alojamiento y portales web,), y solo un 7,73 % a las de fabricación (de componentes electrónicos, computadores y periféricos, equipos de comunicaciones, electrónica de consumo y medios magnéticos y ópticos) [Mas y de Guevara Radoselovics, 2015].

## En España

El Consejo General de Colegios Profesionales de Ingeniería Informática ha publicado en 2015 un estudio sobre la situación laboral de los profesionales del sector de tecnologías de la información (TIC) en España [CCII, 2015]. En este estudio, se refleja que entre los profesionales de este sector podemos hablar de pleno empleo, puesto que solo el 5,9 % estaba desempleado en 2014, y además con una tasa de temporalidad inferior a la media en el conjunto de los trabajadores (el 16 % frente al 24 %). En plena crisis económica estas cifras se pueden considerar muy buenas, aunque no se puede dejar de notar que a partir del 2008 hay un estancamiento en el número de ocupados en este sector.

Este estudio señala que en Aragón trabajan el 4,6 % de los trabajadores del sector TIC de España, lejos del 26 % de Cataluña o del 23,4 % de la Comunidad Autónoma de Madrid, aunque la diferencia se pueda deber principalmente (pero no enteramente, especialmente en el caso de Cataluña) a la mayor población de estas comunidades.

El 58,9 % de los trabajadores en el estudio son ingenieros, licenciados o másteres, mientras que un 20,8 % son ingenieros técnicos, diplomados o tienen el título de grado. El porcentaje total de los que no tienen formación universitaria es tan solo de un 6,2 %. Del total de los encuestados, ocupados y no ocupados, el 62,7 % trabaja por cuenta ajena en el sector privado, el 21,3 % en el sector público y un 7,5 % trabajan por cuenta propia. Sobre los puestos de trabajo más comunes, el 20,9 % son programadores, el 20,6 % son jefes de proyecto, mandos intermedios o similares, el 17,7 % analistas, arquitectos o similares, y el 12,4 % son técnicos administradores de sistemas.

El sector TIC da trabajo al 64,9 % de los ocupados, mientras que el resto desarrollan su actividad como profesionales de las TIC en otros sectores productivos (industria, administración pública etc.). Son las consultoras informáticas, el sector público, los proveedores o fabricantes de software y las empresas de servicios informáticos las principales empleadoras, ocupando entre las cuatro al 66,8 % del sector. Respecto a los salarios brutos anuales, el 41,6 % ganan entre 12.000  y 30.000 , el 47,3 % entre 30.000 y 60.000 , un 7,8 % más de 60.000  y tan solo un 3,3 % gana menos de 12.000 . Es interesante hacer notar que la experiencia laboral y la responsabilidad asociada al puesto de trabajo juegan un papel mayor que el nivel de formación en la remuneración, aunque

también ésta influye.

## En Aragón

El informe del OASI [2014] recoge un conjunto de datos sobre el sector TIC en Aragón hasta 2013. Sobre su dimensión, en 2013 hay 1.668 empresas (frente a las 61.902 que hay en toda España), aunque de esas la mayor parte se dedican a actividades de relativamente bajo valor añadido como el comercio al por menor de equipos TIC (un 28,5 %) y la reparación de ordenadores y equipos (un 15,6 %), frente a un 25,6 % de programación y consultoría o un 8,3 % de proceso de datos y alojamiento. Se asume que la apuesta por la logística en Aragón ha favorecido indirectamente el crecimiento de las actividades de comercio TIC en los últimos años.

El número total de empresas en Aragón ha descendido un 7,23 % entre 2008 y 2013, pero el sector TIC ha crecido un 37,17 % en ese periodo, mostrando que este sector ha resistido mejor la crisis que la media. Con respecto al tamaño, en el sector TIC aragonés, como en el español, predomina la microempresa (menos de 10 trabajadores): el 51,7 % de las empresas aragonesas no tienen asalariados, y el 28,5 % tienen uno o dos. En esto también hay diferencias por el tipo de actividad, y es en programación y consultoría donde podemos encontrar más fácilmente empresas medianas y grandes (y en fabricación de equipos y componentes, pero esas representan un porcentaje muy pequeño del total del sector), mientras que en el comercio minorista y en la reparación de ordenadores las empresas son todas pequeñas (la mayor no llega a 100 empleados).

El número de empleados del sector TIC en Aragón ha crecido un 152 % entre 1996 y 2011 (de 2.822 a 7.122), pero solo la Comunidad Autónoma de Madrid ha crecido menos en ese mismo periodo, y por ejemplo Valencia ha crecido un 1.760,48 %.

# Capítulo 3

# Contexto académico

En este capítulo se revisa la estructura y normativa que regula los estudios universitarios de ingeniería informática en España tras la plena adaptación de los mismos al Espacio Europeo de Educación Superior (EEES) (sección 3.1), y se detallan los contenidos del grado en ingeniería informática (sección 3.2) y el máster universitario en ingeniería informática (sección 3.3) en la Universidad de Zaragoza.

## 3.1. Estructura de los estudios universitarios de ingeniería informática

En el año 2005, en plena transición al EEES, la Agencia Nacional de Evaluación de la Calidad y Acreditación (ANECA) publica el Libro Blanco del Título de Grado en Ingeniería Informática [ANECA, 2005]. El libro recoge el trabajo realizado por una red de universidades españolas con el apoyo de la ANECA, para diseñar un título de grado adaptado al EEES. Este trabajo incluye un análisis de estudios similares en Europa, estudios de inserción laboral de titulados y la definición de perfiles y competencias profesionales.

Una de las conclusiones principales de este libro es que se propone una única titulación de grado en Ingeniería Informática, dejando la especialización para los estudios de máster. Los másteres serían de carácter puramente profesional, o de carácter científico y dirigido hacia la investigación y la obtención del grado de doctor. El grado tendría una orientación profesional, que permita a los titulados integrarse en el mercado laboral, y los objetivos formativos integrarían competencias genéricas básicas, otras transversales relacionadas con la formación integral de las personas y otras más específicas que serán las que posibiliten la integración en el mercado de trabajo. Estas competencias comportan conocimientos, procedimientos, actitudes y rasgos que se deben poseer para afrontar situaciones profesionales.

El libro identifica tres perfiles profesionales, desarrollo de software, sistemas, y gestión y explotación de tecnologías de la información, y propone competencias para cada perfil. El objetivo es que los graduados en Ingeniería Informática tengan una formación amplia y sólida, con una base científica y tecnológica, que les permita abordar sistemas, aplicaciones y productos en todas las fases de su ciclo de vida aplicando los métodos y técnicas propios de la ingeniería.

Entre otras muchas cosas, se espera de los graduados en ingeniería informática la comprensión de la dimensión humana, económica, social, legal y ética de la profesión, la capacidad de asumir responsabilidades técnicas y directivas, la habilidad de dirigir proyectos y trabajar en equipos multidisciplinares, poder aprender de manera autónoma a lo largo de la vida, participar en todas las fases de un sistema informático (desde su especificación inicial hasta su mantenimiento y retirada) y tener la base suficiente para continuar con estudios de máster y de doctorado.

En Marzo de 2006, el Consejo de Coordinación Universitaria publica la ficha técnica de propuesta de título universitario de grado en Ingeniería Informática [de Educación y Ciencia, 2006]. Allí se recogen directrices basadas en el libro blanco, así como capacidades y competencias que deben tener los ingenieros informáticos. También se hace una descripción de materias y las competencias que cada una proporciona.

El libro blanco proponía cuatro categorías de contenidos formativos comunes: fundamentos científicos, contenidos generales de la ingeniería, contenidos específicos de la ingeniería informática y el proyecto fin de carrera (PFC). En la ficha técnica, las materias instrumentales corresponden, aproximadamente, con los fundamentos científicos y contenidos generales de la ingeniería, y las materias propias con los contenidos específicos de la ingeniería informática (el PFC se mantiene, con el papel de verificar que el estudiante ha adquirido las competencias requeridas).

Se deja en mano de cada universidad el desarrollo de los contenidos allí reflejados en sus propios planes de estudios, que de acuerdo al Real Decreto 1393/2007, de 29 de Octubre, deberán ser verificados por el Consejo de Universidades y autorizados por las correspondientes Comunidades Autónomas. Además estos títulos deberán ser inscritos en el Registro de Universidades, Centros y Títulos y acreditados.

En la resolución 12977/2009 de 8 de Junio, el BOE publica un acuerdo del Consejo De Universidades en el que se dan recomendaciones para que las universidades propongan la memorias de solicitud de títulos oficiales en los ámbitos de la Ingeniería Informática, Ingeniería Técnica Informática e Ingeniería Química. Este documento lista en su Apartado 3 las competencias que los estudiantes en Ingeniería Informática (que ese documento equipara provisionalmente con el nivel de máster) deben adquirir:

- "Capacidad para proyectar, calcular y diseñar productos, procesos e instalaciones en todos los ámbitos de la ingeniería informática.

- Capacidad para la dirección de obras e instalaciones de sistemas informáticos, cumpliendo la normativa vigente y asegurando la calidad del servicio.

- Capacidad para dirigir, planificar y supervisar equipos multidisciplinares.

- Capacidad para el modelado matemático, cálculo y simulación en centros tecnológicos y de ingeniería de empresa, particularmente en tareas de investigación, desarrollo e innovación en todos los ámbitos relacionados con la Ingeniería en Informática.

- Capacidad para la elaboración, planificación estratégica, dirección, coordinación y gestión técnica y económica de proyectos en todos los ámbitos de la Ingeniería en Informática siguiendo criterios de calidad y medioambientales.

- Capacidad para la dirección general, dirección técnica y dirección de proyectos de investigación, desarrollo e innovación, en empresas y centros tecnológicos, en el ámbito de la Ingeniería Informática.
- Capacidad para la puesta en marcha, dirección y gestión de procesos de fabricación de equipos informáticos, con garantía de la seguridad para las personas y bienes, la calidad final de los productos y su homologación.
- Capacidad para la aplicación de los conocimientos adquiridos y de resolver problemas en entornos nuevos o poco conocidos dentro de contextos más amplios y mulitidisciplinares, siendo capaces de integrar estos conocimientos.
- Capacidad para comprender y aplicar la responsabilidad ética, la legislación y la deontología profesional de la actividad de la profesión de Ingeniero en Informática.
- Capacidad para aplicar los principios de la economía y de la gestión de recursos humanos y proyectos, así como la legislación, regulación y normalización de la informática."

Allí también se describen los módulos que deben incluir, como mínimo, los másteres para ser oficiales, lo que se reproduce en la Tabla 3.1.

Tabla 3.1: Módulos Máster Ingeniería Informática

| **Módulo** | **ECTS** | **Competencias que deben adquirirse** |
|---|---|---|
| Dirección y Gestión. | 12 | Capacidad para la integración de tecnologías, aplicaciones, servicios y sistemas propios de la Ingeniería Informática, con carácter generalista, y en contextos más amplios y multidisciplinares. Capacidad para la planificación estratégica, elaboración, dirección, coordinación, y gestión técnica y económica en los ámbitos de la ingeniería informática relacionados, entre otros, con: sistemas, aplicaciones, servicios, redes, infraestructuras o instalaciones informáticas y centros o factorías de desarrollo de software, respetando el adecuado cumplimiento de los criterios de calidad y medioambientales y en entornos de trabajo multidisciplinares. Capacidad para la dirección de proyectos de investigación, desarrollo e innovación, en empresas y centros tecnológicos, con garantía de la seguridad para las personas y bienes, la calidad final de los productos y su homologación. |
| Tecnologías Informáticas | 48 | Capacidad para modelar, diseñar, definir la arquitectura, implantar, gestionar, operar, administrar y mantener aplicaciones, redes, sistemas, servicios y contenidos informáticos. Capacidad de comprender y saber aplicar el funcionamiento y organización de Internet, las tecnologías y protocolos de redes de nueva generación, los modelos de componentes, software intermediario y servicios. Capacidad para asegurar, gestionar, auditar y certificar la calidad de los desarrollos, procesos, sistemas, servicios, aplicaciones y productos informáticos. Capacidad para diseñar, desarrollar, gestionar y evaluar mecanismos de certificación y garantía de seguridad en el tratamiento y acceso a la información en un sistema de procesamiento local o distribuido. Capacidad para analizar las necesidades de información que se plantean en un entorno y llevar a cabo en todas sus etapas el proceso de construcción de un sistema de información. Capacidad para diseñar y evaluar sistemas operativos y servidores, y aplicaciones y sistemas basados en computación distribuida. Capacidad para comprender y poder aplicar conocimientos avanzados de computación de altas prestaciones y métodos numéricos o computacionales a problemas de ingeniería. Capacidad de diseñar y desarrollar sistemas, aplicaciones y servicios informáticos en sistemas empotrados y ubicuos. Capacidad para aplicar métodos matemáticos, estadísticos y de inteligencia artificial para modelar, diseñar y desarrollar aplicaciones, servicios, sistemas inteligentes y sistemas basados en el conocimiento. Capacidad para utilizar y desarrollar metodologías, métodos, técnicas, programas de uso específico, normasy estándares de computación gráfica. Capacidad para conceptualizar, diseñar, desarrollar y evaluar la interacción persona-ordenador de productos, sistemas, aplicaciones y servicios informáticos. Capacidad para la creación y explotación de entornos virtuales, y para la creación, gestión y distribución de contenidos multimedia. |

Tabla 3.1: Módulos Máster Ingeniería Informática

| **Módulo** | **ECTS** | **Competencias que deben adquirirse** |
|---|---|---|
| Proyecto fin de máster | | Realización, presentación y defensa, una vez obtenidos todos los créditos del plan de estudios, de un ejercicio original realizado individualmente ante un tribunal universitario, consistente en un proyecto integral de Ingeniería en Informática de naturaleza profesional en el que se sinteticen las competencias adquiridas en las enseñanzas. |

Respecto a las competencias de los estudiantes de grado, esta resolución indica las siguientes:

- "Capacidad para concebir, redactar, organizar, planificar, desarrollar y firmar proyectos en el ámbito de la ingeniería en informática que tengan por objeto, de acuerdo con los conocimientos adquiridos según lo establecido en el apartado 5 de este anexo, la concepción, el desarrollo o la explotación de sistemas, servicios y aplicaciones informáticas.
- Capacidad para dirigir las actividades objeto de los proyectos del ámbito de la informática de acuerdo con los conocimientos adquiridos según lo establecido en el apartado 5 de este anexo.
- Capacidad para diseñar, desarrollar, evaluar y asegurar la accesibilidad, ergonomía, usabilidad y seguridad de los sistemas, servicios y aplicaciones informáticas, así como de la información que gestionan.
- Capacidad para definir, evaluar y seleccionar plataformas hardware y software para el desarrollo y la ejecución de sistemas, servicios y aplicaciones informáticas, de acuerdo con los conocimientos adquiridos según lo establecido en el apartado 5 de este anexo.
- Capacidad para concebir, desarrollar y mantener sistemas, servicios y aplicaciones informáticas empleando los métodos de la ingeniería del software como instrumento para el aseguramiento de su calidad, de acuerdo con los conocimientos adquiridos según lo establecido en el apartado 5 de este anexo.
- Capacidad para concebir y desarrollar sistemas o arquitecturas informáticas centralizadas o distribuidas integrando hardware, software y redes de acuerdo con los conocimientos adquiridos según lo establecido en el apartado 5 de este anexo.
- Capacidad para conocer, comprender y aplicar la legislación necesaria durante el desarrollo de la profesión de Ingeniero Técnico en Informática y manejar especificaciones, reglamentos y normas de obligado cumplimiento.
- Conocimiento de las materias básicas y tecnologías, que capaciten para el aprendizaje y desarrollo de nuevos métodos y tecnologías, así como las que les doten de una gran versatilidad para adaptarse a nuevas situaciones.
- Capacidad para resolver problemas con iniciativa, toma de decisiones, autonomía y creatividad. Capacidad para saber comunicar y transmitir los conocimientos, habilidades y destrezas de la profesión de Ingeniero Técnico en Informática.

- Conocimientos para la realización de mediciones, cálculos, valoraciones, tasaciones, peritaciones, estudios, informes, planificación de tareas y otros trabajos análogos de informática, de acuerdo con los conocimientos adquiridos según lo establecido en el apartado 5 de este anexo.
- Capacidad para analizar y valorar el impacto social y medioambiental de las soluciones técnicas, comprendiendo la responsabilidad ética y profesional de la actividad del Ingeniero Técnico en Informática.
- Conocimiento y aplicación de elementos básicos de economía y de gestión de recursos humanos, organización y planificación de proyectos, así como la legislación, regulación y normalización en el ámbito de los proyectos informáticos, de acuerdo con los conocimientos adquiridos según lo establecido en el apartado 5 de este anexo."

Los planes de estudios de grado deben incluir al menos los módulos reflejados en la Tabla 3.2. Estos se dividen en módulos de formación básica (60 ECTS) y común (60 ECTS), y luego módulos de cinco tecnologías específicas de los que hay que cursar 48 ECTS: Ingeniería del Software, Ingeniería de Computadores, Computación, Sistemas de Información y Tecnologías de la Información.

Tabla 3.2: Módulos Grado Ingeniería Informática

| **Módulo** | **ECTS** | **Competencias que deben adquirirse** |
|---|---|---|
| De formación básica | 60 | Capacidad para la resolución de los problemas matemáticos que puedan plantearse en la ingeniería. Aptitud para aplicar los conocimientos sobre: álgebra lineal; cálculo diferencial e integral; métodos numéricos; algorítmica numérica; estadística y optimización. Comprensión y dominio de los conceptos básicos de campos y ondas y electromagnetismo, teoría de circuitos eléctricos, circuitos electrónicos, principio físico de los semiconductores y familias lógicas, dispositivos electrónicos y fotónicos, y su aplicación para la resolución de problemas propios de la ingeniería. Capacidad para comprender y dominar los conceptos básicos de matemática discreta, lógica, algorítmica y complejidad computacional, y su aplicación para la resolución de problemas propios de la ingeniería. Conocimientos básicos sobre el uso y programación de los ordenadores, sistemas operativos, bases de datos y programas informáticos con aplicación en ingeniería. Conocimiento de la estructura, organización, funcionamiento e interconexión de los sistemas informáticos, los fundamentos de su programación, y su aplicación para la resolución de problemas propios de la ingeniería. Conocimiento adecuado del concepto de empresa, marco institucional y jurídico de la empresa. Organización y gestión de empresas. |

Tabla 3.2: Módulos Grado Ingeniería Informática

| **Módulo** | **ECTS** | **Competencias que deben adquirirse** |
|---|---|---|
| Común a la rama de informática | 60 | Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente. Capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social. Capacidad para comprender la importancia de la negociación, los hábitos de trabajo efectivos, el liderazgo y las habilidades de comunicación en todos los entornos de desarrollo de software. Capacidad para elaborar el pliego de condiciones técnicas de una instalación informática que cumpla los estándares y normativas vigentes. Conocimiento, administración y mantenimiento sistemas, servicios y aplicaciones informáticas. Conocimiento y aplicación de los procedimientos algorítmicos básicos de las tecnologías informáticas para diseñar soluciones a problemas, analizando la idoneidad y complejidad de los algoritmos propuestos. Conocimiento, diseño y utilización de forma eficiente los tipos y estructuras de datos más adecuados a la resolución de un problema. Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados. Capacidad de conocer, comprender y evaluar la estructura y arquitectura de los computadores, así como los componentes básicos que los conforman. Conocimiento de las características, funcionalidades y estructura de los Sistemas Operativos y diseñar e implementar aplicaciones basadas en sus servicios.Conocimiento y aplicación de las características, funcionalidades y estructura de los Sistemas Distribuidos, las Redes de Computadores e Internet y diseñar e implementar aplicaciones basadas en ellas. Conocimiento y aplicación de las características, funcionalidades y estructura de las bases de datos, que permitan su adecuado uso, y el diseño y el análisis e implementación de aplicaciones basadas en ellos. Conocimiento y aplicación de las herramientas necesarias para el almacenamiento, procesamiento y acceso a los Sistemas de información, incluidos los basados en web. Conocimiento y aplicación de los principios fundamentales y técnicas básicas de la programación paralela, concurrente, distribuida y de tiempo real. Conocimiento y aplicación de los principios fundamentales y técnicas básicas de los sistemas inteligentes y su aplicación práctica. Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería de software. Capacidad para diseñar y evaluar interfaces persona computador que garanticen la accesibilidad y usabilidad a los sistemas, servicios y aplicaciones informáticas. Conocimiento de la normativa y la regulación de la informática en los ámbitos nacional, europeo e internacional. |
| Ingeniería del Software | 48 | Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y eficiente, sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la Ingeniería del Software. Capacidad para valorar las necesidades del cliente y especificar los requisitos software para satisfacer estas necesidades, reconciliando objetivos en conflicto mediante la búsqueda de compromisos aceptables dentro de las limitaciones derivadas del coste, del tiempo, de la existencia de sistemas ya desarrollados y de las propias organizaciones. Capacidad de dar solución a problemas de integración en función de las estrategias, estándares y tecnologías disponibles. Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software sobre la base de un conocimiento adecuado de las teorías, modelos y técnicas actuales. Capacidad de identificar, evaluar y gestionar los riesgos potenciales asociados que pudieran presentarse. Capacidad para diseñar soluciones apropiadas en uno o más dominios de aplicación utilizando métodos de la ingeniería del software que integren aspectos éticos, sociales, legales y económicos. |
| Ingeniería de Computadores | 48 | Capacidad de diseñar y construir sistemas digitales, incluyendo computadores, sistemas basados en microprocesador y sistemas de comunicaciones. Capacidad de desarrollar procesadores específicos y sistemas empotrados, así como desarrollar y optimizar el software de dichos sistemas. Capacidad de analizar y evaluar arquitecturas de computadores, incluyendo plataformas paralelas y distribuidas, así como desarrollar y optimizar software de para las mismas. Capacidad de diseñar e implementar software de sistema y de comunicaciones. Capacidad de analizar, evaluar y seleccionar las plataformas hardware y software más adecuadas para el soporte de aplicaciones empotradas y de tiempo real.Capacidad para comprender, aplicar y gestionar la garantía y seguridad de los sistemas informáticos. Capacidad para analizar, evaluar, seleccionar y configurar plataformas hardware para el desarrollo y ejecución de aplicaciones y servicios informáticos. Capacidad para diseñar, desplegar, administrar y gestionar redes de computadores. |

Tabla 3.2: Módulos Grado Ingeniería Informática

| **Módulo** | **ECTS** | **Competencias que deben adquirirse** |
|---|---|---|
| Computación | 48 | Capacidad para tener un conocimiento profundo de los principios fundamentales y modelos de la computación y saberlos aplicar para interpretar, seleccionar, valorar, modelar, y crear nuevos conceptos, teorías, usos y desarrollos tecnológicos relacionados con la informática. Capacidad para conocer los fundamentos teóricos de los lenguajes de programación y las técnicas de procesamiento léxico, sintáctico y semántico asociadas, y saber aplicarlas para la creación, diseño y procesamiento de lenguajes. Capacidad para evaluar la complejidad computacional de un problema, conocer estrategias algorítmicas que puedan conducir a su resolución y recomendar, desarrollar e implementar aquella que garantice el mejor rendimiento de acuerdo con los requisitos establecidos. Capacidad para conocer los fundamentos, paradigmas y técnicas propias de los sistemas inteligentes y analizar, diseñar y construir sistemas, servicios y aplicaciones informáticas que utilicen dichas técnicas en cualquier ámbito de aplicación. Capacidad para adquirir, obtener, formalizar y representar el conocimiento humano en una forma computable para la resolución de problemas mediante un sistema informático en cualquier ámbito de aplicación, particularmente los relacionados con aspectos de computación, percepción y actuación en ambientes o entornos inteligentes. Capacidad para desarrollar y evaluar sistemas interactivos y de presentación de información compleja y su aplicación a la resolución de problemas de diseño de interacción persona computadora. Capacidad para conocer y desarrollar técnicas de aprendizaje computacional y diseñar e implementar aplicaciones y sistemas que las utilicen, incluyendo las dedicadas a extracción automática de información y conocimiento a partir de grandes volúmenes de datos. |
| Sistemas de Información | 48 | Capacidad de integrar soluciones de Tecnologías de la Información y las Comunicaciones y procesos empresariales para satisfacer las necesidades de información de las organizaciones, permitiéndoles alcanzar sus objetivos de forma efectiva y eficiente, dándoles así ventajas competitivas. Capacidad para determinar los requisitos de los sistemas de información y comunicación de una organización atendiendo a aspectos de seguridad y cumplimiento de la normativa y la legislación vigente. Capacidad para participar activamente en la especificación, diseño, implementación y mantenimiento de los sistemas de información y comunicación. Capacidad para comprender y aplicar los principios y prácticas de las organizaciones, de forma que puedan ejercer como enlace entre las comunidades técnica y de gestión de una organización y participar activamente en la formación de los usuarios. Capacidad para comprender y aplicar los principios de la evaluación de riesgos y aplicarlos correctamente en la elaboración y ejecución de planes de actuación. Capacidad para comprender y aplicar los principios y las técnicas de gestión de la calidad y de la innovación tecnológica en las organizaciones. |
| Tecnologías de la Información | 48 | Capacidad para comprender el entorno de una organización y sus necesidades en el ámbito de las tecnologías de la información y las comunicaciones. Capacidad para seleccionar, diseñar, desplegar, integrar, evaluar, construir, gestionar, explotar y mantener las tecnologías de hardware, software y redes, dentro de los parámetros de coste y calidad adecuados. Capacidad para emplear metodologías centradas en el usuario y la organización para el desarrollo, evaluación y gestión de aplicaciones y sistemas basados en tecnologías de la información que aseguren la accesibilidad, ergonomía y usabilidad de los sistemas. Capacidad para seleccionar, diseñar, desplegar, integrar y gestionar redes e infraestructuras de comunicaciones en una organización. Capacidad para seleccionar, desplegar, integrar y gestionar sistemas de información que satisfagan las necesidades de la organización, con los criterios de coste y calidad identificados. Capacidad de concebir sistemas, aplicaciones y servicios basados en tecnologías de red, incluyendo Internet, web, comercio electrónico, multimedia, servicios interactivos y computación móvil. Capacidad para comprender, aplicar y gestionar la garantía y seguridad de los sistemas informáticos. |
| Proyecto Fin de Grado | 12 | Ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consistente en un proyecto en el ámbito de las tecnologías específicas de la Ingeniería en Informática de naturaleza profesional en el que se sinteticen e integren las competencias adquiridas en las enseñanzas. |

## 3.2. El grado en ingeniería informática en la Universidad de Zaragoza

La Universidad de Zaragoza oferta esta titulación que se imparte tanto en Zaragoza, en la Escuela de Ingeniería y Arquitectura, como en Teruel, en la Escuela Universitaria

Politécnica. La aprobación del grado se publicó en el BOE de 11 de Noviembre de 2010 y el plan de estudios en el BOE de 29 de Noviembre de 2010.

Este grado requiere que los estudiantes completen 240 créditos, distribuidos en cuatro años cada uno con dos cuatrimestres y con la siguiente estructura:

1. Formación básica: 10 asignaturas de 6 créditos, 9 el primer año y la última el primer cuatrimestre del segundo.

2. Formación obligatoria: 17 asignaturas de 6 créditos de las que 14 se cursan entre los cuatrimestre tercero, cuarto y quinto de los estudios.

3. Formación de especialidad: 8 asignaturas de 6 créditos en la especialidad que elijan los estudiantes que se cursan en los 3 últimos cuatrimestres del grado.

4. Formación optativa: 16 créditos, más 2 créditos de inglés.

5. Y para terminar un trabajo fin de grado de 12 créditos.

Respecto a las especialidades, en Teruel se ofertan las de Sistemas de Información y Tecnologías de la Información, y en Zaragoza se ofertan las de Computación, Ingeniería de Computadores, Ingeniería del Software, Sistemas de Información y Tecnologías de la Información.

Las tablas siguientes muestran la distribución de las asignaturas, tanto las comunes como las de especialidad, en los cuatro cursos del grado en la EINA, en Zaragoza. La tabla 3.3 lista las asignaturas de formación básica, la tabla 3.4 lista las de formación común obligatoria, la tabla 3.6 muestra las asignaturas de cada una de las especialidades y la tabla 3.5 contiene las optativas. A las optativas hay que añadir las de las otras especializaciones, que se pueden cursar como optativas, y otras asignaturas transversales.

Tabla 3.3: Asignaturas de formación básica

| Asignatura | Cuatrimestre | ECTS |
|---|---|---|
| Introducción a los computadores | 1 | 6 |
| Fundamentos de administración de empresas | 1 | 6 |
| Matemáticas I | 1 | 6 |
| Matemáticas II | 1 | 6 |
| Programación I | 1 | 6 |
| Arquitectura y organización de computadores 1 | 2 | 6 |
| Física y electrónica | 2 | 6 |
| Estadística | 2 | 6 |
| Matemática discreta | 2 | 6 |
| Teoría de la computación | 3 | 6 |

De acuerdo a los datos del portal de transparencia de la Universidad de Zaragoza, en general el interés por el grado en ingeniería informática ha evolucionado positivamente desde su implantación en el curso 2010/2011. Con los datos del grado impartido en la EINA, el número de estudiantes preinscritos ese año fue de 293 (128 de ellos en primera preferencia), mientras que en el curso 2015/2016 eran 505 preinscritos, de los

Tabla 3.4: Asignaturas de formación común

| Asignatura | Cuatrimestre | ECTS |
|---|---|---|
| Programación 2 | 2 | 6 |
| Sistemas operativos | 3 | 6 |
| Redes de computadores | 3 | 6 |
| Programación de sistemas concurrentes y distribuidos | 3 | 6 |
| Estructuras de datos y algoritmos | 3 | 6 |
| Arquitectura y organización de computadores 2 | 4 | 6 |
| Administración de sistemas | 4 | 6 |
| Interacción persona ordenador | 4 | 6 |
| Tecnología de programación | 4 | 6 |
| Bases de datos | 4 | 6 |
| Proyecto Hardware | 5 | 6 |
| Sistemas distribuidos | 5 | 6 |
| Ingeniería del Software | 5 | 6 |
| Inteligencia artificial | 5 | 6 |
| Sistemas de información | 5 | 6 |
| Proyecto Software | 6 | 6 |
| Seguridad informática | 7 | 6 |
| Idioma moderno Inglés B1 | 8 | 2 |
| Trabajo fin de Grado | 8 | 12 |

Tabla 3.5: Asignaturas optativas

| Asignatura | Curso | ECTS |
|---|---|---|
| Comercio electrónico | 3 | 6 |
| Metodologías ágiles y calidad | 4 | 6 |
| Bioinformática | 4 | 6 |
| Robótica | 4 | 6 |
| Videojuegos | 4 | 6 |
| Visión por computador | 4 | 6 |
| Laboratorio de sistemas de información | 4 | 6 |
| Sistemas de información distribuidos | 4 | 6 |
| Prevención de riesgos laborales aplicada a la ingeniería | 4 | 6 |
| Inglés técnico | 4 | 6 |

Tabla 3.6: Asignaturas específicas de cada especialidad

| Asignatura | Cuatrimestre | ECTS |
|---|---|---|
| *Computación* | | |
| Algoritmia básica | 6 | 6 |
| Procesadores de lenguajes | 6 | 6 |
| Aprendizaje automático | 6 | 6 |
| Algoritmia para problemas difíciles | 7 | 6 |
| Recuperación de información | 7 | 6 |
| Informática gráfica | 7 | 6 |
| *Sistemas de Información* | | |
| Bases de datos 2 | 6 | 6 |
| Sistemas de información 2 | 6 | 6 |
| Tecnologías de la información en la empresa | 6 | 6 |
| Almacenes y minería de datos | 7 | 6 |
| Sistemas legados | 7 | 6 |
| Sistemas de ayuda a la toma de decisiones | 7 | 6 |
| Sistemas y tecnologías Web | 8 | 6 |
| *Tecnologías de la Información* | | |
| Bases de datos 2 | 6 | 6 |
| Tecnologías de la información en la empresa | 6 | 6 |
| Administración de sistemas 2 | 6 | 6 |
| Centros de datos | 7 | 6 |
| Diseño y administración de redes | 7 | 6 |
| Ingeniería Web | 7 | 6 |
| Sistemas legados | 7 | 6 |
| Sistemas y tecnologías Web | 8 | 6 |
| Diseño centrado en el usuario. Diseño para la multimedia | 8 | 6 |
| *Ingeniería de Computadores* | | |
| Procesadores comerciales | 6 | 6 |
| Sistemas empotrados I | 6 | 6 |
| Multiprocesadores | 6 | 6 |
| Centros de datos | 7 | 6 |
| Diseño y administración de redes | 7 | 6 |
| Sistemas empotrados II | 7 | 6 |
| Laboratorio de sistemas empotrados | 8 | 6 |
| Garantía y seguridad | 8 | 6 |
| *Ingeniería del Software* | | |
| Ingeniería de requisitos | 6 | 6 |
| Verificación y validación | 6 | 6 |
| Arquitectura software | 6 | 6 |
| Ingeniería Web | 7 | 6 |
| Gestión de proyecto software | 7 | 6 |
| Sistemas legados | 7 | 6 |
| Laboratorio de ingeniería del software | 8 | 6 |
| Sistemas y tecnologías Web | 8 | 6 |

cuáles 192 lo habían puesto como primera preferencia. Eso implica que la nota media de admisión al grado también ha aumentado, siendo de un 8,62 el primer curso, y de un 9,701 en el 2015/2016.

Uno de los puntos negativos es la diferencia de porcentajes entre hombres y mujeres matriculados en el grado. En el año 2015/2016 había 325 hombres y 43 mujeres, es decir que solo un 11,7 % de los matriculados son mujeres.

## 3.3. El máster universitario en ingeniería informática en la Universidad de Zaragoza

El máster universitario en Ingeniería Informática de la Universidad de Zaragoza es un máster oficial que tiene como objetivo la formación de profesionales que puedan cubrir las necesidades del mundo profesional industrial y científico y que se imparte en la Escuela de Ingeniería y Arquitectura. Esta dirigido especialmente a graduados en ingeniería informática y a ingenieros en informática e ingenieros técnicos en informática de los planes antiguos, pero se puede acceder con otras titulaciones. Se pretende que los egresados puedan asumir el papel de directores técnicos (CTO) en empresas TIC, directores de tecnologías de la información (CIO) en empresas TIC y no TIC, directores de proyectos de software o hardware, consultores de sistemas de información o investigadores en cualquier ámbito de la informática.

El máster tiene un total de 90 créditos ECTS organizados en cuatro módulos: dirección y gestión (12 créditos), tecnologías informáticas (48 créditos), materias optativas y prácticas en empresa (15 créditos) y un trabajo fin de máster de 15 créditos. Las asignaturas obligatorias están reflejadas en la Tabla 3.7 y las optativas en la Tabla 3.8.

Tabla 3.7: Asignaturas obligatorias del máster

| Asignatura | Curso | ECTS |
|---|---|---|
| Sistemas inteligentes | 1 | 6 |
| Calidad en el desarrollo de software, servicios de infraestructuras TI | 1 | 6 |
| Computación de altas prestaciones | 1 | 6 |
| Redes y sistemas distribuidos | 1 | 6 |
| Administración y dirección estratégica de empresas | 1 | 6 |
| Manipulación y análisis de grandes volúmenes de datos | 1 | 6 |
| Sistemas empotrados ubicuos | 1 | 6 |
| Tecnologías y modelos para el desarrollo de aplicaciones distribuidas | 1 | 6 |
| Computación gráfica-entornos inmersivos-multimedia | 1 | 6 |
| Gestión de la innovación en tecnologías de la información | 1 | 6 |
| Trabajo Fin de Máster | 2 | 15 |

De acuerdo a los datos del portal de transparencia de la Universidad de Zaragoza, el número de matriculados ha pasado de 8 en su primer año (curso 2014/2015) a 11 en el segundo (curso 2015/2016), lejos de la oferta de 60 plazas en el último año.

Tabla 3.8: Asignaturas optativas del máster

| Asignatura | Curso | ECTS |
|---|---|---|
| Prácticas 1 | 2 | 3 |
| Prácticas 2 | 2 | 3 |
| Prácticas 3 | 2 | 3 |
| Machine learning for Big Data | 2 | 3 |
| Sistemas bioinspirados e ingeniería de sistemas complejos | 2 | 3 |
| Análisis avanzado de datos | 2 | 3 |

# Capítulo 4

# La enseñanza-aprendizaje en la educación superior

La educación es una práctica social compleja y como tal no es objeto directo de una reflexión epistemológica, así que son la didáctica y la pedagogía las disciplinas que podemos hacer accesibles a este escrutinio. El objeto principal de estas disciplinas es la enseñanza, objeto que articula a todos los demás en el campo de la educación: el profesor, el alumno, la materia o contenido que se enseña, los métodos de enseñanza y las formas de incitar al aprendizaje [Vásquez, 2012, p. 119-120].

Decía José Ortega y Gasset en su obra "Misión de la Universidad" (publicada por primera vez en 1930, pero editada posteriormente, por ejemplo ver [Ortega y Gasset, 2007]) que el gran paso dado en la historia de la pedagogía consistió en pasar de la enseñanza que parte del saber y del maestro, dejando de lado al discípulo, a una pedagogía que reconoce que es el discípulo y sus condiciones peculiares lo único que puede guiarnos en la enseñanza. Esto, publicado en 1930, todavía conviene recordarlo con cierta frecuencia. La enseñanza para ser efectiva tiene que realizarse a partir de la comprensión sobre cómo aprenden los estudiantes [Fry et al., 2009, p. 3]. Es por eso que este capítulo empieza dando una visión sobre los principios del aprendizaje (sección 4.1), los estudiantes (sección 4.2) y los equipos de estudiantes (sección 4.3), y después de esto es cuando se abordan las distintas metodologías de enseñanza-aprendizaje que se aplican en el contexto de este proyecto (sección 4.4) y la evaluación (sección 4.5).

## 4.1. El aprendizaje

El aprendizaje trata sobre nuestra percepción y comprensión del mundo. Incluye dominar principios abstractos, comprender pruebas, recordar hechos, adquirir métodos y técnicas, razonar, debatir o saber comportarse en distintas situaciones. Existen varias teorías relevantes y fundamentadas en la investigación sobre el aprendizaje y, aunque todavía hay mucho que no sabemos, sí que se pueden usar para decidir los tipos de acciones que pueden ser útiles para facilitar que ocurra el aprendizaje. Las teorías constructivistas son las principales hoy en día.

La base del constructivismo es la idea de construir y corregir las estructuras mentales que recogen el conocimiento. El aprendizaje es un proceso de transformación

individual. Piaget y Bruner son dos de los principales educadores cuyas ideas se pueden enmarcar en el constructivismo. Como profesores, esta visión nos debe recordar que no estamos escribiendo en un tabla rasa, que los estudiantes tienen conocimientos previos, quizás rudimentarios o incluso equivocados, y que sin cambios o adiciones a estos conocimientos previos, poco aprendizaje va a ocurrir. No es solo añadir más conocimientos a los estudiantes, el aprendizaje de alto nivel (comprensión, creatividad) solo ocurre cuando las estructuras mentales existentes son modificadas [Fry et al., 2009, p. 9,10].

Las investigaciones de Ference Marton han llevado a establecer que los estudiantes se aproximan al aprendizaje de dos maneras fundamentales:

- **Aproximación profunda**: caracterizada por la intención de comprender y buscar el significado, conduce a los estudiantes a intentar relacionar conceptos entre sí y con los que ya conocen, a distinguir ideas nuevas y pre-existentes, y a evaluar y determinar conceptos. Los hechos se aprenden en el contexto del aprendizaje.
- **Aproximación superficial**: el objetivo es completar la tarea, memorizar información y tratar la tarea como una impuesta externamente. El objetivo es aparentar que se ha aprendido pero llegando a esto solo a través de un procesamiento cognitivo superficial. Se aprenden hechos sin un marco conceptual.

Otros investigadores han identificado una tercera aproximación, la **estratégica o de consecución**, asociada con la evaluación. El énfasis se pone en organizar el aprendizaje para obtener una nota alta en los procesos de evaluación.

La taxonomía SOLO (*Structure of the Observed Learning Outcomes*, estructura de los resultados de aprendizaje observados), basada en el principio de que conforme los estudiantes aprenden, los resultados de su aprendizaje pasan a través de áreas de complejidad creciente, puede usarse como guía en el desarrollo de currículum y la articulación de resultados de aprendizaje y criterios de evaluación. Es una clasificación jerárquica, en la que cada nivel es la base para el siguiente, y en la que cada nivel representa la cantidad de detalle y calidad del aprendizaje cuando se está en el:

- **Pre-estructural:** comprensión de las palabras. Poca evidencia de aprendizaje relevante. No debería aparecer en la educación superior.
- **Uni-estructural:** los estudiantes se focalizan en un solo aspecto de la tarea y se pierden otros que son fundamentales.
- **Multi-estructural:** hay presentes varios hechos, pero no están estructurados y no se abordan los temas claves.
- **Relacional:** es más que una lista de detalles, aborda el tema principal y relaciona el tema como un todo. Es el primer nivel donde se muestra comprensión en un sentido académicamente relevante.
- **Abstracción extendida:** se conceptualiza un todo coherente en un alto nivel de abstracción y se aplica a nuevos y más amplios contextos. Se ha cambiado la forma de pensar sobre ciertos temas, representa un alto nivel de comprensión.

Otra taxonomía importante es la de Bloom, que cubre seis niveles de habilidades cognitivas:

1. **Conocimiento:** ¿Qué esperamos que aprendan los estudiantes? Registra, examina, reproduce, ordena, define, presenta, describe, identifica, muestra, cita...
2. **Comprensión:** ¿Pueden los estudiantes interpretar e interpolar a partir de lo que saben? Discute, clarifica, clasifica, explica, traduce, extiende, interpreta, revista, selecciona, resume, contrasta...
3. **Aplicación:** ¿Pueden los estudiantes ver la relevancia de una idea un una situación nueva? Resuelve, examina, modifica, aplica, usa, elige, relaciona, calcula, clasifica, demuestra...
4. **Análisis:** ¿Pueden los estudiantes descomponer ideas en sus partes constitutivas y mostrar cómo se relacionan? Diferencia, investiga, categoriza, critica, debate, compara, contrasta, distingue, resuelve, analiza, calcula...
5. **Síntesis:** ¿Pueden los estudiantes combinar elementos en un patrón o estructura que no estaba ahí antes? ¿Derivar relaciones abstractas? ¿Proponer un plan? Ensambla, organiza, compón, propón, construye, diseña, crea, formula, integra, modifica, deriva, desarrolla...
6. **Evaluación:** ¿Pueden los estudiantes hacer juicios basados en evidencias? Juzga, selecciona, evalúa, elige, valora, compara, estima, mide, argumenta, defiende, resume...

Es evidente que la experiencia juega un papel importante en el aprendizaje, y el aprendizaje basado en la experiencia es un fundamento de muchos tipos de enseñanza-aprendizaje. La teoría más habitual sobre el aprendizaje basado en la experiencia es la de David Kolb.

El aprendizaje basado en la experiencia se basa en la idea de que la experiencia contribuye a la formación y a la modificación de la comprensión. Es un proceso continuo en el que llevamos a las situaciones de aprendizaje nuestros conocmientos, ideas, creencias etc., y que estás serán corregidas y reformadas por la experiencia si conseguimos aprender de ella. Esto se concreta en el llamada ciclo de aprendizaje de Kolb, que indica que hacen falta cuatro tipos de habilidades/acciones para que el aprendizaje basado en la experiencia tenga éxito: la experimentación activa, que lleva a tener una experiencia concreta, que lleva a realizar una observación reflexiva que conduce a una conceptualización abstracta que a su vez vuelve a conducir a nuevas experimentaciones activas. Es en la observación reflexiva y la conceptualización abstracta donde se puede ejercer una fuerte influencia en el aprendizaje proporcionando realimentación sobre el mismo y facilitando el paso a la conceptualización abstracta.

Dentro de esta sección es importante hablar algo sobre los "estilos de aprendizaje". A pesar de ser uno de los términos más frecuentes en relación con el aprendizaje en los estudiantes, su misma noción es problemática. Hay distintas categorizaciones de estilos[1], y las pruebas experimentales sobre su existencia son bastante escasas [Fry et

[1]Seriales-Holísticos (Pask), Activos-Reflexivos-Teóricos-Pragmáticos (Honey y Mumford), Convergentes-Divergentes-Asimiladores-Acomodadores (Wolf y Kolb) y otras.

al., 2009, p. 18]. Hasta que la experimentación conduzca a pruebas más solidas, es más seguro afirmar que cada estudiante tiene su propio estilo de aprendizaje y que parte de la tarea del profesor es descubrirlo, que tratar de encajar a todos los estudiantes en unas pocas categorías artificiales.

Finalmente, no se puede olvidar el papel que juega el contexto social y cultural en el aprendizaje, alejando un poco el foco de los individuos. La diferencia entre lo que alguien puede entender por sí mismo, y lo que puede ser entendido con ayuda conducen a conceptos como el de aprendizaje "con andamios", es decir proporcionando soporte y ayuda, el aprendizaje situado, aprender en un contexto, y otras metodologías de enseñanza-aprendizaje actuales.

## 4.2. Los estudiantes

Los estudiantes son una parte fundamental en el proceso de enseñanza-aprendizaje, pero lo cierto es que a veces no se les toma en demasiada consideración: "están ahí porque quieren" o "ya son adultos" es todo lo que algunos necesitan saber sobre su motivación o "algunos no valen, que se dediquen a otra cosa" es cuanto hace falta para justificar sus, y nuestros, fracasos. Pero como ya hemos visto antes, todas las teorías actuales sobre el aprendizaje reconocen que es la comprensión de los estudiantes el primer paso para desarrollar la enseñanza-aprendizaje más efectivos y adecuados.

Sobre la motivación, se suele distinguir primero entre intrínseca y extrínseca. Los estudiantes intrínsecamente motivados disfrutan con los desafíos, quieren dominar la materia, son curiosos y quieren aprender, mientras que los extrínsecamente motivados se preocupan por sus notas, por las recompensas externas y por obtener la aprobación de otros. Otros autores distinguen entre objetivos de cumplimiento y objetivos de aprendizaje. Los objetivos de cumplimiento están ligados con la motivación extrínseca, mientras que los de aprendizaje lo están con la intrínseca. Hay otras distinciones, pero en general se puede reconocer en ellas esas dos clases de motivación [Fry et al., 2009, p. 28].

Otro concepto que surge es de la desmotivación. Algunos estudiantes realmente no saben por qué están en la Universidad, piensan que son incompetentes y creen que no tienen suficiente control sobre los que les pasa. Muestran una ausencia de motivación. Así que podemos concluir que, *grosso modo*, existe un grado de motivación que puede ir desde una gran motivación por obtener logros, intrínseca o extrínseca, a una total desmotivación.

En distintos estudios, los estudiantes manifiestan unos objetivos bastante consistentes que quieren alcanzar cursando una educación superior: principalmente obtener un buen trabajo y desarrollar habilidades útiles, después algunas razones sobre desarrollo personal y, en mucho menor porcentaje, aparecen algunos estudiantes que no saben muy bien por qué están en la Universidad [Fry et al., 2009, p. 31, 32].

¿Cómo podemos mejorar la motivación de los estudiantes? Ningún profesor quiere estudiantes desmotivados y la mayoría sostendría que la motivación intrínseca es mejor que la extrínseca. Para empezar, un estudiante que empieza la Universidad sin objetivos claros, o peor con el único objetivo de retrasar su entrada en la vida laboral o adulta, va a ser bastante difícil, sino imposible, de motivar. Pero hay algunas pruebas

de que lo que hacemos en la Universidad puede desmotivar a nuestros estudiantes. Por ejemplo, si perciben que reciben una realimentación insuficiente o poco adecuada sobre su trabajo, o que trabajar más no se traduce en un aumento claro de sus notas, pierden motivación. Una solución es proporcionar la realimentación adecuada, por ejemplo dándoles un conjunto de ejemplos de lo que se espera de sus trabajos para obtener buenas notas.

Después, tenemos que ver cómo aumentar la motivación intrínseca. Hay bastantes pruebas de que la mayoría de los estudiantes tienden a adoptar aproximaciones superficiales (la motivación extrínseca es parte de esto) en la Universidad. Algunos autores son bastante pesimistas sobre la posibilidad de cambiar esto, y argumentan que la educación universitaria es parte de un sistema que se va a resistir al cambio, y que cambiar una asignatura o grupo de asignaturas no sirve de nada porque todo el entorno va a presionar para que todo siga igual. También hay resultados negativos cuando se ha tratado de enseñar a los estudiantes mejores formas de estudiar, puesto que en general tardan poco en revertir a sus aproximaciones superficiales iniciales.

Un aspecto del sistema educativo que sí que parece crucial para la motivación de los estudiantes es la evaluación [Fry et al., 2009, p. 33-35]. Algunos estudios muestran que los estudiantes de último año empiezan intrínsecamente motivados y tratando de adoptar aproximaciones de aprendizaje profundo, pero que conforme se acercan los exámenes pasan a estar extrínsecamente motivados y a adoptar aproximaciones superficiales. El sistema de evaluación debería alentar la comprensión conceptual. Esto podría conseguirse aumentando el uso de la resolución de problemas, casos de estudio etc., donde el conocimiento debe usarse, y no solo aprenderse. Además estas evaluaciones pueden tener lugar en condiciones formales de examen, para evitar algunos de los problemas asociados con la evaluación continua (como por ejemplo, que algunos estudiantes tratarán de hacer trampas).

La Tabla 4.1, adaptada de [Fry et al., 2009, Table 3.3], resume un conjunto de acciones que pueden llevarse a cabo para aumentar la motivación a los estudiantes, dependiendo del tipo de motivación que necesiten.

Tabla 4.1: Acciones para motivar a los estudiantes

| **Motivación** | **Posibles acciones** |
|---|---|
| Ser eficaz y competente | Proporcionar realimentación clara y adecuada. Diseñar tareas que permiten tener éxito pero también ser un desafío. |
| Tener el control | Proporcionar realimentación que haga énfasis en la naturaleza del aprendizaje como proceso, incluyendo la importancia del esfuerzo y la estrategia. Proporcionar oportunidades para hacer elecciones. Construir relaciones personales de apoyo en la comunidad de aprendizaje. |
| Motivaciones intrínsecas y un alto grado de interés | Proporcionar tareas y materiales estimulantes e interesantes, incluyendo novedad y variación y que tengan algún significado para los estudiantes. Mostrar el interés y la involucración en el contenido y las actividades. |

Tabla 4.1: Acciones para motivar a los estudiantes

| **Motivación** | **Posibles acciones** |
|---|---|
| Hacer algo valioso | Proporcionar tareas y materiales relevantes y útiles, que permitan identificarse personalmente con lo que se aprende. El discurso en clase debería concentrarse en la importancia y utilidad de los contenidos y actividades. |
| Alcanzar objetivos | Usar estructuras organizativas que alienten la responsabilidad y proporcionen entornos predecibles y controlados. Usar grupos cooperativos y colaborativos que permitan alcanzar objetivos sociales y académicos. El discurso en clase debería concentrarse en la maestría, el aprendizaje y la comprensión. Usar estructuras de recompensa y evaluación que promuevan la maestría, el aprendizaje, el esfuerzo, el progreso y la mejora personal, y menos en la comparación social o en estándares normativos. |

## 4.3. Los equipos de estudiantes

Algunas metodologías de enseñanza-aprendizaje, por ejemplo el aprendizaje basado en problemas, son normalmente llevadas a cabo en equipos de estudiantes. La mayor parte de la vida profesional de la mayor parte de los estudiantes va a consistir en trabajar con otros, y esta es una competencia muy demandada por las empresas. Podemos distinguir varios tipos de equipos [Savin-Baden y Major, 2004, p. 71-75]:

- **El equipo guiado por un tutor:** el profesor guía a los estudiantes en las distintas fases del problema y puede proporcionar pistas sobre las técnicas adecuadas para solucionarlos. Adecuado para problemas que están más allá de las habilidades actuales de los estudiantes.

- **El equipo de aprendizaje colaborativo:** el objetivo es desarrollar ciertas habilidades en equipo, lo que requiere ciertas habilidades sociales, y la capacidad de comunicarse claramente con otros miembros del equipo, aceptar otros puntos de vista y resolver conflictos. El profesor proporciona realimentación y al final evalúa al equipo.

- **El equipo reflexivo:** se espera que los estudiantes sean capaces de señalar puntos de incomodidad relacionados con su rol en el equipo, la relación entre logros individuales y colectivos y la naturaleza del apoyo que se encuentra en el equipo. el equipo sirve como un mecanismo para capacitar a los individuos.

- **El equipo cooperativo:** hay evidencias importantes de que los equipos cooperativos son superiores a las estructuras competitivas o individualistas en varios resultados, como en logros académicos, razonamiento de alto nivel, frecuencia

de generación de nuevas ideas y soluciones y una mayor transferencia de conocimientos entre situaciones, además de que crea relaciones más positivas entre los estudiantes y entre los estudiantes y el profesor. El equipo cooperativo es distinto del colaborativo en que el cooperativo busca un pequeño trabajo en grupo para maximizar el aprendizaje del estudiante, mientras que el colaborativo tiene su foco en que el aprendizaje se realiza en un grupo y que ese proceso enriquece a los individuos. En aprendizaje cooperativo se tienden a mantener las líneas tradicionales de conocimiento y autoridad, mientras que el colaborativo se basa más en el constructivismo social (la idea de que el conocimiento humano se construye a través de la interacción con otros).

- **El equipo de aprendizaje en acción:** este equipo es un grupo de personas que se juntan con el objetivo "de hacer las cosas". En la práctica, cada miembro trae un problema y busca ayuda para resolverlo. El aprendizaje ocurre mediante un proceso de reflexión y acción por el individuo sobre su problema, con la ayuda de los otros, el foco está en el individuo y en sus acciones futuras. Esta aproximación genera grupos más individualizados, con poco soporte por parte del profesor, y logra un aprendizaje más personal y reflexivo que el que se vería en un grupo colaborativo.

Existen muchos estudios que confirman que el trabajo en equipo mejora un conjunto importante de resultados de aprendizaje. Los resultados académicos y cognitivos mejoran, los estudiantes tienen que enseñar cosas a otros miembros de su equipo y las aprenden mejor de esta forma, y además el tener que aplicar el conocimiento les obliga a articularlo y dominarlo mejor. Finalmente, los estudiantes se ven más activamente involucrados en el proceso de aprendizaje [Savin-Baden y Major, 2004, p. 76].

Los equipos de aprendizaje exitoso comparten ciertas características. Debe haber interdependencia positiva entre los miembros de un equipo, se necesitan unos a otros para tener éxito, las interacciones entre ellos deben ayudar a mejorar, los individuos deben ser responsables de su trabajo y sus contribuciones, tienen que trabajar en equipo y aplicar habilidades sociales para la toma de decisiones, la comunicación y la gestión de conflictos, y deben ser capaces de reflexionar como equipo al concluir un problema sobre fortalezas y debilidades y como mejorar la próxima vez. Aunque hay quien aboga por grupos pequeños, de dos o tres miembros, y quien preferiría grupos de hasta ocho, el cinco es una buena cifra, suficiente para obligar a que el equipo tenga cohesión[2] y variedad de talentos y experiencia.

Difícilmente van a llegar los estudiantes a la Universidad con habilidades de trabajo en equipo bien desarrolladas. Así que es ahí donde tienen que desarrollar habilidades interpersonales, creación y gestión de equipos, resolución de conflictos etc. Por otra parte, la preponderancia del trabajo por objetivos y resultados en el mundo profesional y la creciente visión del estudiante como "consumidor" empujan hacia el individualismo y hacia la devaluación de las aproximaciones colaborativas y dialogadas al aprendizaje. Los equipos deben ser motivados (trabajo en problemas reales, realimentación adecuada sobre su efectividad y recompensas por sus logros) pero también tienen que

[2]Los grupos pequeños suelen tener más conflictos de personalidad, los grupos de tres tienden a formar una pareja "con carabina", y los grupos de cuatro tienden a formar dos parejas.

trabajar juntos para construir equipo, desarrollando unas reglas que formen la base de un "contrato" entre los miembros del equipo (comprometerse a compartir información, apoyarse mutuamente, criticarse constructivamente, ser puntuales, ser formales cuando haya actividades programadas, mantener un nivel de equidad en la carga de trabajo, reconocer las aportaciones de cada miembro, responsabilizarse como equipo del progreso del trabajo etc.) [Savin-Baden y Major, 2004, p. 79].

## 4.4. Metodología de enseñanza-aprendizaje

### 4.4.1. La lección magistral

La lección, o lección magistral, ha sido el método de enseñanza estándar en educación superior durante siglos. Son un mecanismo eficaz para transmitir mucho material en relativamente poco tiempo, puede impartirse a cientos de estudiantes a la vez, es una forma de que un experto en un campo comparta su conocimiento rápida y sucintamente, y muchos estudiantes pueden aprender de una lección bien preparada [Clement, 2010, ch. 6].

Aunque en los últimos años el constructivismo ha puesto en primer plano las actividades centradas en el estudiante, sigue siendo necesario transmitir conocimientos, y las lecciones y presentaciones son una buena forma de hacerlo, siempre que sean efectivas. Lo primero para ello es evitar algunas cosas: hablar mucho y muy rápido son defectos comunes de los instructores que hay que evitar. Tampoco hay que caer en la trampa de asumir que oído es lo mismo que comprendido. Además se pueden hacer algunas cosas para mejorar las lecciones:

- Ser visual. Proyectar los puntos principales de la lección en pantalla. Mostrar un resumen a grandes rasgos de la presentación. Usar recursos multimedia cuando sean pertinentes.
- Organizar la lección en partes. Deben crearse oportunidades para interactuar con los estudiantes, haciendo que la lección sea más activa para ellos. Hacer pausas cada 12-18 minutos, y dejar que los estudiantes puedan hablar entre ellos sobre el material, responder a preguntas, hacer preguntas o chequear sus notas. Tener una introducción, un cuerpo y una conclusión.
- Adelantar cosas sobre la lección. Captar la atención al principio, puede ser con una cita célebre, una pregunta o un pequeño clip de video.
- Llevar a cabo actividades que permitan reflexionar sobre, y sintetizar, los contenidos transmitidos.

Generar y mantener el interés en una lección incrementa la motivación de los estudiantes por aprender. El profesor tiene que empezar mostrándose entusiasta e interesado, si es posible explicando por qué el tema es interesante para él mismo, organizado y teniendo el control de la situación. Se puede empezar explicando los resultados de aprendizaje que se esperan para la lección y enlazando el tema con algún tema de actualidad. Después hay que usar ejemplos actuales y relevantes, si es posible cercanos

a la experiencia e involucrar a los estudiantes de manera activa conforme se avanza (ver la sección 4.4.2) [Fry et al., 2009, p. 59, 60].

### 4.4.2. Las preguntas

Las preguntas son una herramienta que se puede utilizar en el aula para fomentar un aprendizaje más activo en los estudiantes. Hay distintos tipos y se pueden usar para varios fines [Clement, 2010, p. 76-81]:

- Preguntas para valorar el interés y conocimiento previo de los estudiantes: si queremos que los estudiantes puedan guiarnos sobre lo que saben y necesitan, lo que tenemos que hacer es preguntárselo. No directamente con preguntas de sí o no, puesto que en ese caso tenderán a decir que sí para parecer que saben más, sino con preguntas abiertas e incluso problemas. Puede ser al principio del cuatrimestre, o al principio de una lección individual.

- Preguntas convergentes: aquellas que llevan a los estudiantes a una respuesta única, que pueden usarse para empezar lecciones o para empezar discusiones. Por ejemplo, se les puede preguntar sobre alguna experiencia propia que ya sabemos a dónde conduce.

- Preguntas divergentes: aquellas que tienen varias respuestas posibles. Sirven para que los estudiantes practiquen la resolución de problemas complejos que tienen muchas soluciones posibles.

- Preguntas de orden superior: teniendo en cuenta la taxonomía de Bloom, serían las preguntas que abordan las últimas categorías de la misma (análisis, síntesis y evaluación). Si queremos que nuestros estudiantes sean capaces de analizar, sintetizar y evaluar, tendremos que pedirles que hagan exactamente esas mismas cosas en nuestras clases.

Una vez diseñadas las preguntas, hay que decidir cuándo y cómo hacerlas [Clement, 2010, p. 83, 84]. Para empezar hay que tener claro para qué se hacen. Si es para valorar el conocimiento y comprensión generales de los estudiantes en una clase, hay que hacérselas a un número suficientemente alto. Cuando se hace una pregunta, hay que tener claro si cualquiera puede responderla, o si vamos a preguntar a alguien concreto. Podemos darles un tiempo para que las piensen y luego pedir voluntarios. Si queremos rebajar el estrés, podemos decirles que le den su respuesta a un compañero y luego pedir voluntarios para compartirlas con todos. Hay que hacer del aula un lugar seguro para contestar: no se aprende nada de preguntas que solo asustan o humillan. Hay que ser agradables y profesionales, educados y asertivos.

Cuando nos dan una respuesta, lo que se espera es que proporcionemos realimentación sobre la misma. Si un estudiante se equivoca, es mejor que el profesor dé la respuesta correcta que esperar a que conteste otro compañero, pues eso puede dañar innecesariamente la autoestima del que se equivocó. Se puede felicitar a un estudiante, siempre que tengamos cuidado con felicitarle por la respuesta y por su comportamiento, y no por ser inteligente. Ante la duda, siempre será más seguro felicitar en privado

(o haciendo una anotación en un trabajo o examen); hay que tener cuidado con lo que se dice en clase.

### 4.4.3. Las prácticas de laboratorio

Las clases de laboratorio son una parte integral de los programas de ingeniería, que es una materia con un marcado carácter práctico. Estas clases pueden ser desde pruebas rutinarias a experiencias de primera mano sobre cómo algo funciona o la demostración práctica de conceptos teóricos. Las clases de laboratorio están muy centradas en los estudiantes por su propia naturaleza, y proporcionan un gran número de resultados de aprendizaje: obtener habilidades prácticas, ganar experiencia en el uso de determinados sistemas o equipos, planificar pruebas, relacionar teoría y práctica, obtener y analizar datos, desarrollar habilidades de resolución de problemas y desarrollar habilidades personales. [Fry et al., 2009, p. 269-270].

Los laboratorios son relativamente caros de poner en marcha, mantener y equipar. Además requieren personal que les pueda dedicar tiempo. Es por tanto especialmente importante planificar bien las sesiones de laboratorio e integrarlas programa de la asignatura de manera que se maximicen los beneficios por su uso.

### 4.4.4. La enseñanza con grupos pequeños

Cuando hay menos de 20 estudiantes[3] en un aula (o un seminario, o incluso un despacho si es una tutoría de grupo pequeño), el tipo de enseñanza es distinto a la lección que se ha explicado en la sección 4.4.1. Muchos autores sostienen que esta actividad es más complicada: hay que conocer bien el tema, estar atento a cómo funciona el grupo, abierto a cambiar cosas y ser capaz de dirigir el grupo sin necesidad de imponer autoridad [Fry et al., 2009, ch. 6].

Aunque trabajar con un grupo pequeño parece menos formal que con un grupo grande, un buen profesor planificará deliberadamente los objetivos que quiere conseguir en la sesión. Se pueden realizar muchas actividades en grupos pequeños que son difíciles o imposibles con grupos grandes. Algunos ejemplos:

- Sesión de *brainstorm*: generación de ideas sin criticarlas inicialmente.
- Discusión libre: los estudiantes discuten, el profesor observa.
- Seminario: un estudiante presenta un artículo y el grupo lo discute.
- Simulación o juego: una experiencia estructurada en la que los estudiantes toman ciertos roles. Es necesario que el profesor dé guías y, especialmente, realimentación.
- Tutoría: reunión con un grupo pequeño para dar realimentación sobre un trabajo presentado.

[3]Por ejemplo. Nadie ha establecido definitivamente cuando un grupo de estudiantes es pequeño.

### 4.4.5. El aprendizaje basado en problemas y proyectos

El aprendizaje basado en problemas (PBL, *problem-based learning*) nace como una forma de aprendizaje en el que los estudiantes se ven involucrados en escenarios donde tiene que resolver problemas realistas. En su origen, los estudiantes forman grupos pequeños que exploran una situación problemática y, gracias a eso, descubren las carencias en sus conocimientos y habilidades para decidir que información necesitan para resolver la situación. Desde entonces el concepto ha evolucionado, y podemos distinguir más características que aparecen en muchos cursos que usan PBL: reconocimiento de la experiencia de base de los estudiantes, énfasis en la responsabilidad de los estudiantes sobre su aprendizaje, multidisciplinariedad, teoría y práctica entrelazadas, foco de atención en el proceso más que en la adquisición de conocimiento, un cambio en el papel del tutor de instructor a facilitador, un cambio de foco desde la evaluación por el tutor a la auto-evaluación y evaluación entre pares, énfasis en la comunicación y las habilidades interpersonales [Savin-Baden y Major, 2004, p. 3, 4].

Partiendo de que la resolución de problemas es un importante resultado de aprendizaje, Jonassen [1997] ha analizado los problemas en educación y los ha dividido en tres categorías:

- Problemas puzle: descontextualizados, requieren ciertas estrategias de razonamiento abstracto, tienen una única solución correcta y todos los elementos requeridos para la solución son conocibles y conocidos por adelantado.
- Problemas bien estructurados: más dependientes del dominio pero todavía bien definidos, con un estado inicial, un estado objetivo y un número finito de operadores lógicos para ir de uno al otro.
- Problemas mal estructurados: algunos aspectos de la situación no están bien especificados (la descripción del problema o la información necesaria para resolverlo son incompletas o imprecisas). Las soluciones no son predecibles, puede haber varias y haber varios caminos para llegar a ellas, y pueden requerir la integración de contenidos de varios dominios.

Dabbagh y Dass [2013] han revisado varios trabajos que concluyen que los problemas para ser efectivos deben ser realistas, relevantes, actuales, abordar temas emergentes, hacer que los estudiantes se sientan identificados con, e interesados en, ellos, y deben proporcionar una oportunidad para explorar y para aplicar conocimiento profesional. En el mismo artículo, estos autores señalan que muchos problemas usados en educación son a menudo descripciones parciales de situaciones reales que puede que no proporcionen las habilidades de resolución de problemas necesarias en el mundo profesional.

Los profesionales deben afrontar muchos problemas mal estructurados durante sus actividades por la compleja naturaleza de muchos entornos profesionales [Chen, 2009]. Los problemas mal definidos, o mal estructurados, son tareas prototípicas que los profesionales abordan a causa de la naturaleza mal estructurada de su trabajo [Dabbagh y Dass, 2013]. Jonassen [1997] también señala que los problemas mal estructurados son comunes en la práctica profesional cotidiana, y resalta que estos problemas requieren un conjunto específico de habilidades de resolución de problemas.

Los problemas de diseño (de productos, procesos, sistemas o métodos) son los más comunes en la práctica profesional de la ingeniería [Jonassen et al., 2006], y el diseño es reconocido como una de las actividades clave de esta disciplina [Simon, 1996]. Este tipo de problemas están entre los peor estructurados: requieren conocimiento de dominio y estratégico, la solución siempre es un resultado original, y los requisitos, las restricciones y los criterios de éxito son normalmente vagos, si no desconocidos [Jonassen, 2011].

La ingeniería del software es una disciplina amplia, que requiere la integración de distintos conocimientos y capacidades, y muy focalizada en el diseño de software. El currículum de ACM/IEEE de informática señala que la mejor aproximación para su enseñanza es hacer que los estudiantes participen en proyectos para desarrollar sistemas software, y que esto es incluso más efectivo si usan una aproximación ágil o iterativa [ACM/IEEE-CS Joint Task Force on Computing Curricula, 2013, p. 174]. Pensar sobre diseño requiere, en general, manejar ambigüedad e incertidumbre, pensamiento a nivel de sistema, toma de decisiones y las habilidades para tomar parte en un proceso social y la capacidad para hacer uso de varios lenguajes de diseño; esto se puede aprender mejor en un entorno de aprendizaje basado en proyectos [Dym et al., 2005].

El aprendizaje basado en proyectos tiene muchas cosas en común con el aprendizaje basado en problemas, pero también algunas diferencias: los proyectos están normalmente más cerca a la realidad profesional que los problemas, los proyectos están orientados hacia la aplicación del conocimiento más que hacia su adquisición y la gestión de recursos y diferenciación de roles entre los estudiantes es muy importante en el aprendizaje basado en proyectos [Perrenet et al., 2000]. Otras diferencias que podemos resaltar son que el aprendizaje basado en proyectos es más estructurado, el tutor es un supervisor más que un facilitador y la gestión aparece como una actividad necesaria [Savin-Baden y Major, 2004, Table 1.1].

Con respecto a la autenticidad de la aproximación basada en proyectos, [Strobel et al., 2013, Table 4] han examinado el uso del término "autenticidad" en educación en diseño y educación en ingeniería y han propuesto unos cuantos factores clave de autenticidad. Entre estos factores clave podemos encontrar las situaciones profesionales realistas, los problemas mal estructurados o la toma de decisiones en contextos prácticos. Los problemas mal estructurados también son un estímulo para actividades auténticas [Dabbagh y Dass, 2013]. Una aproximación basada en proyectos puede por tanto proporcionar un entorno de aprendizaje más auténtico en la educación en ingeniería.

### 4.4.6. El *e-learning* y otras tendencias

La expansión del uso de Internet ha cambiado profundamente la sociedad y, por supuesto, también ha traído cambios al mundo educativo. La Web es un recurso enorme de contenidos, de calidad variable, un lugar donde obtener respuestas de un experto[4] es fácil y casi inmediato y el sitio donde cada vez más gente acude para desarrollar una educación informal.

[4]Los sitios web con sistemas de revisiones, votaciones a las mejores respuestas, reconocimientos explícitos a los que responden más y mejor etc. cada vez hacen más difícil pasar por experto en algo sin serlo.

El *e-learning* no es fácil de definir, pero se puede considerar en un sentido amplio que es el uso de la tecnología para crear oportunidades de aprendizaje. La tecnología, especialmente todo lo relacionado con Internet, se puede usar para facilitar algunas tareas de los profesores y estudiantes universitarios, y sería un error no aprovecharla. Por otra parte, la tecnología es un apoyo del profesor, y no un sustituto. Los MOOC (*Massive Open Online Courses*, cursos en línea abiertos y masivos), considerados a veces como el futuro de la educación a distancia (o incluso de la educación), son normalmente cursos de nivel universitario bien diseñados, e impartidos a través de Internet a quien quiera participar en ellos. Son masivos, pueden alcanzar decenas de miles de matriculados, abiertos (usan tecnología con licencias abiertas y están abiertos a que se matricule cualquiera), son totalmente en línea y asíncronos (cada estudiante puede trabajar a su ritmo). Pero siguen siendo cursos, a menudo una versión digital de un curso tradicional, con clases impartidas por profesores experimentados, grabadas en vídeo y materiales publicados en Internet. Hoy en día hay un intenso debate sobre su importancia, su longevidad, y sobre el posible impacto que van a tener en la enseñanza [Simonson et al., 2015, p. 39].

El uso de plataformas para el desarrollo de cursos online, como por ejemplo la plataforma de código abierto Moodle[5], permite fácilmente aprovechar Internet para muchas actividades de enseñanza-aprendizaje, entre las más comunes:

- Compartir materiales con los estudiantes: transparencias, apuntes, documentos de apoyo, enlaces seleccionados a páginas Web, vídeos, podcasts y otros.
- Establecer foros de diálogo en los que el profesor y los estudiantes pueden intercambiar preguntas y respuestas de manera asíncrona, visibles por todos y consultables en cualquier momento.
- Establecer fechas y mecanismos de entrega de resultados: el sistema recuerda a los estudiantes las fechas pendientes, y todas las entregas quedan recogidas en el mismo sitio fácilmente accesible. Las entregas se pueden configurar para que sean por grupos (solo es necesario que entregue un estudiante de cada grupo) o individuales.
- Hacer encuestas o tests en línea: los resultados quedan automáticamente recogidos en formato tabular para facilitar su análisis posterior.
- Hacer anuncios o comunicaciones para los estudiantes en formato "tablón": el sistema también puede enviar automáticamente un e-mail, pero la comunicación queda registrada en la página del curso y los estudiantes no pueden alegar no haberla recibido.

A medio o largo plazo, muchos expertos aprecian dos tendencias claras en la educación superior: el avance de los entornos de aprendizaje flexibles, preparados para facilitar las interacciones requeridas en entornos de aprendizaje activo, y el incremento de la colaboración entre instituciones de educación superior, que cada vez más se

[5] `https://moodle.org/`

están uniendo en consorcios para combinar recursos o avanzar en la innovación educativa. Además de estas dos, el uso de nuevas fuentes de datos para ayudar a personalizar los aprendizajes y medir los resultados del mismo, la expansión de los recursos educativos abiertos, gratuitos y libres en términos de propiedad y derechos de uso, el incremento del aprendizaje mixto, combinando el presencial y el no presencial, la mezcla del aprendizaje formal e informal, siendo el informal el que está dirigido por la curiosidad y contribuye a mejorar la participación e interés de los estudiantes, y las posibilidades que se pueden abrir conforme se implanten tecnologías de consumo como la Internet de las cosas, son desafíos, pero también oportunidades, para mejorar la educación superior y que esta pueda llegar a cada vez más gente [Johnson et al., 2015].

## 4.5. La evaluación

La evaluación en la enseñanza es un tema controvertido, donde hay opiniones muy diversas que se defienden apasionadamente. Aunque no hay muchos datos, los que hay apuntan a que normalmente tanto los profesores como los estudiantes consideran que el principal propósito de la evaluación es poner nota a los logros de los estudiantes. Sin embargo, los profesores piensan que también es un elemento motivador del aprendizaje, algo con lo que los estudiantes tienden a no coincidir en absoluto. Los profesores creen que la evaluación puede proporcionar una realimentación valiosa a los estudiantes[6], pero los estudiantes piensan que tiene muy poco que ver con la mejora de su aprendizaje. La evaluación moldea en buena medida el aprendizaje de los estudiantes, así que si queremos cambiar como aprenden, o lo que aprenden, un mecanismo efectivo es cambiar cómo son evaluados [Fry et al., 2009, p. 133, 134].

Un buen punto de partida sobre cuál es el objetivo de la evaluación lo tenemos en la agencia de aseguramiento de la calidad del Reino Unido (QAA), que determina cuatro propósitos principales de esta (que tienen algunos solapes):

1. **Pedagogía:** proporcionar realimentación a los estudiantes para que mejoren su desempeño, y para determinar qué y cómo aprenden.

2. **Medida:** valorar el conocimiento, comprensión y habilidades de los estudiantes.

3. **Estandarización:** proporcionar una nota que permita establecer el desempeño de los estudiantes. Esta nota puede usarse también para tomar decisiones sobre el progreso.

4. **Certificación:** permitir al público (incluyendo a los empleadores) y a los organismos de educación superior saber que un individuo ha obtenido un nivel adecuado de logros que refleja ciertos estándares académicos establecidos.

Si queremos que la evaluación sirva para hacer pedagogía (los otros objetivos son más fáciles y más típicamente tenidos en cuenta), es decir la *evaluación formativa*, lo primero es que no podemos considerar la evaluación como un punto final, sino como un comienzo o un punto intermedio. Si la consideramos un punto final, la convertimos

[6] Aunque luego sus propias prácticas de evaluación no suelen ser consistentes con esta visión.

casi inevitablemente en *evaluación sumativa*, y esto condicionará nuestra enseñanza (enseñamos para el examen) y el aprendizaje de los estudiantes (estudiarán para el examen).

Una forma de que la evaluación sea más formativa es focalizarla en resultados de aprendizaje, especialmente si son de alta calidad (aquellos que pueden llevarse a, y usarse en, nuevos contextos). Eso contribuye a lograr un aprendizaje en profundidad por parte de los estudiantes. También hay que tratar de diseñar la evaluación antes que los contenidos del módulo que se evalúa. Esto contribuye a tomar un punto de vista focalizado en el estudiante, y no en el profesor, algo que en la literatura cada vez está más claro que contribuye a que los estudiantes adopten una aproximación en profundidad al aprendizaje [Fry et al., 2009, p. 135, 136].

Nicol y MacfarlaneDick [2006] han propuesto siete principios que debería tener una realimentación durante la evaluación del aprendizaje para ser eficaz:

1. Ayuda a clarificar lo que es un buen desempeño (objetivos, criterios, estándares esperados).
2. Facilita el desarrollo de auto-evaluación (reflexión) sobre el aprendizaje.
3. Entrega información de calidad a los estudiantes sobre su aprendizaje.
4. Alienta un diálogo con el profesor y con los otros estudiantes sobre el aprendizaje.
5. Alienta creencias motivacionales positivas y la auto-estima[7].
6. Proporciona oportunidades para cerra el hueco entre el desempeño actual y el deseado.
7. Proporciona información a los profesores que puede ayudarse a dar forma a la enseñanza.

Sobre las pruebas de evaluación, lo más importante es considerar el tipo de resultado de aprendizaje que queremos comprobar y hacerlas consistentes con estos. Por ejemplo, si queremos comprobar si los estudiantes pueden construir un argumento coherente y razonado, hay que pedirles que redacten un texto, mientras que si se quiere evaluar sus habilidades prácticas, observarles llevarlas a cabo en un laboratorio puede ser más adecuado. Y en el momento de poner notas, hay buscar al menos fiabilidad (que dos personas asignaran la misma nota al mismo trabajo), validez (que las notas midan lo que se supone que miden) y transparencia (que los criterios de evaluación sean públicos, las tareas de evaluación se convoquen con tiempo suficiente y haya un proceso justo de apelación) [Fry et al., 2009, p. 143, 144].

[7]Esto es en la práctica muy difícil de conseguir, sobre todo si un estudiante obtiene una calificación baja. En este caso hay que tener mucho cuidado con lo que se le da por escrito y, si es posible, soportarlo con realimentación verbal, que es más fácil de moderar si el estudiante está desmoralizado o muy confuso.

# Capítulo 5

# La materia de gestión de proyectos de software

Este capítulo proporciona una base a las dos asignaturas incluidas en este proyecto docente dando una breve panorámica sobre la materia común de ambas, la gestión de proyectos de software. La sección 5.1 proporciona una breve perspectiva histórica sobre los proyectos, repasa algunas definiciones y conceptos fundamentales y revisa algunas cifras sobre la gestión de proyectos de software en la actualidad. Además de las referencias citadas, se ha consultado material de [Weaver, 2007a] y [Weaver, 2007b]. La sección 5.2 examina el lugar que la gestión de proyectos de software tiene en el Computer Science Curricula de 2013 (CS2013) [ACM/IEEE-CS Joint Task Force on Computing Curricula, 2013]. Finalmente, la sección 5.3 describe el lugar de esta materia en los estudios de ingeniería informática en España, revisando para ello los programas de varias universidades.

## 5.1. Los proyectos

Un proyecto es un esfuerzo temporal que busca crear un producto, servicio o resultado único. Por ejemplo, la fabricación de decenas o cientos de coches iguales (o muy similares entre sí) cada día en un factoría no es un proyecto. Proyectos relacionados con esta actividad serían el diseño de los coches, el diseño y organización de la factoría, el diseño del plan de inspección y aseguramiento de la calidad en esa factoría etc.

Otra definición es que un proyecto es el conjunto de actividades requeridas para entregar resultados (*deliverables*) a algún cliente. Se asume que los proyectos tienen una duración específica, consumen recursos y producen algún tipo de "productos de trabajo".

Actividades que podemos considerar proyectos llevan haciéndose desde hace milenios (por ejemplo, las pirámides de Egipto, hace unos 4500 años) y seguramente se haya reflexionado sobre algunos aspectos de su realización desde hace más o menos el mismo tiempo (por ejemplo, el arte de la guerra de Sun Tzu, escrito hace unos 2500 años, trata sobre planificación y estrategia, entre otros contenidos). Grandes edificios, monumentos o los ferrocarriles transcontinentales en el siglo XIX se considerarían proyectos desde un perspectiva actual. Sin embargo, no es hasta el siglo XX cuando se empieza

a hablar de gestión de proyectos. Hasta entonces se hablaba de adoración/devoción, ingeniería, construcción nacional etc. y los que los dirigían se llamaban a si mismos sacerdotes, ingenieros, arquitectos etc. Normalmente se considera que el proyecto Manhattan (desarrollo de la primera bomba atómica, en los años 40 del siglo XX) es el primer proyecto con gestión moderna, aunque sus directores se veían así mismo como oficiales del ejército y/o científicos.

Es común señalar que hablamos de proyectos en el sentido actual de ese término cuando se siguen ciertos métodos para gestionarlos. Esto nos lleva primero a reflexionar sobre la palabra "gestión". El sentido en el que se usa este término cuando se habla de proyectos, es el que se refiere a la actividad de coordinar los esfuerzos de personas para alcanzar objetivos usando los recursos disponibles eficiente y eficazmente. Esta actividad incluirá la planificación, la organización, la contratación de personal, el liderazgo y la obtención de los recursos financieros, tecnológicos o naturales para llevar a cabo el proyecto, entre otras tareas.

### 5.1.1. El origen de las técnicas de gestión de proyectos: el método del camino crítico y la planificación

La planificación consiste en determinar qué tareas hay que hacer, y en qué orden, para llevar a cabo un proyecto. Ningún proyecto puede llevarse a cabo sin tener al menos una idea a grandes rasgos de esto. Los diagramas de Gantt son una de las herramientas básicas de planificación: muestran el tiempo en el eje X y las actividades en que dividimos el proyecto en el eje Y, y se usan barras horizontales para indicar el inicio y el final de cada actividad. En realidad diagramas similares a estos son anteriores a Henry L. Gantt (que vivió a principios del siglo XX), como mínimo se remontan al siglo XVIII (la figura 5.1 muestra un ejemplo con la línea de vida de distintos personajes de entre el año -600 y el año 0), y tampoco los usaba Henry Gantt, sino que se usaban en las fábricas donde el ayudaba[1]: *"Many shops have a very nice schedule system; they plan their work beautifully—at least, it looks very pretty on paper; but they have no means of finding out whether those schedules are lived up to or not"*.

La planificación de línea de flujo surge en los años 30 (siglo XX) y permite construir el Empire State Building en tiempo récord. Esta técnica sirve para planificar recursos en actividades repetitivas (construcción de carreteras, oleoductos, rascacielos etc.), de manera que se maximice el uso de recursos y se minimicen interrupciones en el trabajo. Después de esto, la línea de equilibro es inventada por Goodyear (años 40) y adoptada por la marina de EE.UU. (años 50) tanto para tareas repetitivas, como no repetitivas. Esta es una técnica para representar gráficamente ciertos hechos con respecto al tiempo. Permite saber qué actividades van acorde a lo planificado, y cuáles no (pero no nos dice por qué hay problemas ni cómo solucionarlos). Los diagramas de hitos son inventados también en los 40, y representan una serie de hitos (eventos de

[1]La principal contribución de Henry Gantt a la gestión de proyectos fue una aproximación innovadora a la gestión de la fuerza de trabajo (hoy se llamaría a esto gestión de equipos), focalizada en el uso eficiente del trabajo y en una división justa de las recompensas debidas a los incrementos en productividad entre los trabajadores y los dueños de las fábricas.

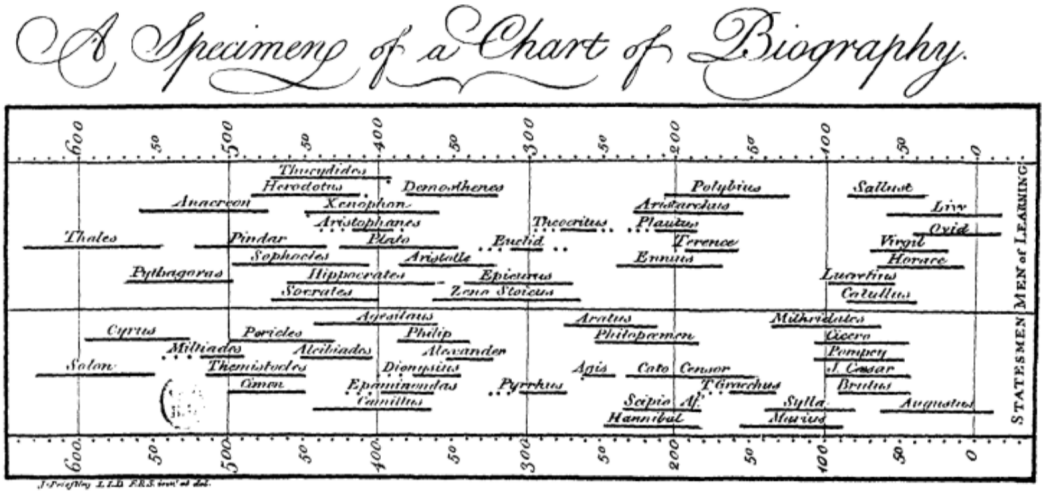



Figura 5.1: Joseph Priestley: Chart of Biography (1765)

los que podemos decir si han sucedido o no) en el tiempo.

En el año 1959 se publica el primer artículo que discute el método del camino crítico (*Critical Path Method*, CPM) en planificación. El camino crítico es el camino más largo de actividades en la planificación de un proyecto, y su longitud es por tanto el tiempo más corto en que podemos terminar el proyecto si todo va de acuerdo con lo planificado. El CPM es una metodología específica para calcular el camino crítico en un proyecto para el que hemos planificado sus actividades y sus dependencias (qué actividades tienen que haber acabado para empezar otras). Poco después se desarrolla el sistema PERT (*Project Evaluation and Review Techniques*), que nos da otra forma de calcular el camino crítico pero aceptando para cada actividad cierta incertidumbre en cuanto a su duración.

El gobierno de EEUU pronto se da cuenta que la planificación es solo parte de la respuesta y pronto su ejército y la NASA desarrollan nuevas herramientas, como la técnica WBS (*Work Breakdown Structure*) para la división de tareas en partes más simples, la técnica PERT/RAMPS que permite considerar la asignación de recursos y la planificación en varios proyectos relacionados y otras. Varias de estas técnicas, y otras que son evolución de estas, se siguen usando en la actualidad. De hecho, y aunque es discutible, se puede afirmar que, salvo la gestión de riesgos, no se ha desarrollado ningún principio fundamental relacionados con el coste, el diseño, o la planificación y su control desde los años 60 (hay técnicas nuevas, pero solo el tiempo dirá si son nuevos principios, o solo refinamientos de los existentes).

### 5.1.2. El alcance de la gestión de proyectos

El qué se incluye, y qué no, dentro de la disciplina de la gestión de proyectos es un tema en continua evolución. Aunque la esencia sigue siendo el "triángulo de hierro", tiempo, coste y resultados, a estos tres elementos A estos se les han ido uniendo con los años la calidad (no solo del producto sino del proceso), los riesgos, la gestión de personas y las comunicaciones.

Conforme la gestión de proyectos iba abarcando más aspectos, se iban desarrollando

metodologías para sistematizar y documentar cómo las organizaciones gestionaban sus proyectos. El núcleo de estas metodologías son las descripciones de sus procesos. En los últimos años parece haber una tendencia que hace más énfasis en los modelos de madurez que en las metodologías.

Solo es con el crecimiento de la disciplina que surge la necesidad de especializarse en ella y de reconocer a las personas encargadas de asegurarse de que se llevan a cabo las acciones necesarias para la gestión de los proyectos. Por ello el papel de director de proyecto no surge hasta el siglo XX. Antes el liderazgo podía ser de un maestro constructor, un arquitecto, o un ingeniero, aunque en los siglos XVIII y XIX ya se separa el diseño arquitectónico de los procesos de gestión y de construcción, que podían ser llevados a cabo por distintos contratistas con responsabilidades contractuales con respecto a un programa de trabajo[2].

### 5.1.3. Fundamentos teóricos de la gestión de proyectos

Encontramos el origen de los fundamentos teóricos de la gestión de proyectos en dos bases. La primera es el liberalismo económico, el capitalismo de Adam Smith y la división del trabajo. La segunda es el origen del método científico, observación, hipótesis, experimento y teoría, y el énfasis que este hace en el esfuerzo para la eliminación de sesgos y valoraciones basadas en opiniones, creencias no comprobadas y prejuicios.

Durante la revolución industrial se inventan algunas técnicas y herramientas aún en uso como la producción estandarizada con partes intercambiables, los controles de calidad de productos, la contabilidad de costes de producción y la planificación de esta producción.

A principios del siglo XX aparece la que hoy se denomina escuela clásica. Se busca la eficiencia e incluye una gestión científica, burocrática y administrativa:

- La gestión científica busca mejorar la productividad mejorando la eficiencia de los procesos de producción y aplicando el reduccionismo, que consiste en coger tareas complejas y aprovechar la división del trabajo dividiéndolas en tareas simples. También se hacen comparaciones entre la producción planificada (o intentada) y la real, buscando identificar las causas de las diferencias.

- La gestión burocrática define y reparte funciones, usa la autoridad legal, jerarquías, reglas y procedimientos escritos, contratación y promoción basada en la competencia y los conocimientos y carreras profesionales bien definidas.

- La gestión administrativa hace énfasis en el gestor y en la gestión: los gestores planifican, organizan, ordenan, coordinan y controlan. Se incluye división del trabajo, autoridad y responsabilidad, disciplina, unidad del mando, unidad de la dirección, subordinación del interés individual al general, remuneración del personal, centralización, cadena de mando, orden, equidad, estabilidad del personal, iniciativa y espíritu de equipo (la unión hace la fuerza).

[2] De hecho, acuerdos contractuales detallados (especificación del trabajo, requisitos de métodos de pago, tiempo de finalización etc.) para grandes obras vienen usándose desde la Grecia y la Roma clásicas, aunque luego se pierden y no se recuperan en Europa hasta el Renacimiento

En los años 20 se empieza a hacer énfasis en los aspectos humanos de las organizaciones como reacción a las limitaciones de la escuela clásica[3]. Se descubre el "efecto Hawthorne", que demuestra en algunos experimentos clásicos[4] que la productividad mejora por el simple hecho de observar a los trabajadores, lo que es la primera prueba empírica de la importancia de los factores humanos en la productividad.

Como resultado, se empieza a considerar que la dirección de una empresa debe establecer y mantener un sistema de comunicación efectivo, contratar y mantener a personal efectivo y motivar a este personal. El foco de la autoridad cambia desde la "autoridad legal" de la escuela clásica a la "aceptación de la autoridad" que sostiene que los gerentes o directores solo tienen la autoridad que sus empleados les permiten tener. Para eso, los empleados deben entender lo que que la gerencia quiere de ellos, deben ser capaces de hacerlo, deben pensar que lo que hacen ayuda a alcanzar los objetivos de la organización y deben creer que lo que hacen no es contrario a sus objetivos personales. A partir de los años 50 la escuela de recursos humanos considera que los trabajadores quieren hacer trabajos significativos, contribuir, participar en la toma de decisiones y en el liderazgo.

En los años 60 se trata de integrar las ideas anteriores alrededor de la teoría de sistemas. Se asume que un sistema es un conjunto de elementos relacionados e interdependientes funcionando como un todo, que tiene entradas desde el entorno, procesos de transformación de las entradas en resultados terminados, y salida de esos resultados de nuevo al entorno. Las distintas partes combinadas y coordinadas consiguen más que individualmente. Los proyectos se ven como sistemas complicados, con múltiples entradas, salidas y procesos relacionados y se asume que puede beneficiarse del análisis de sistemas.

Tras bastantes esfuerzos para encontrar técnicas de gestión universalmente aplicables, surge la visión de contingencias. Bajo esta visión, se debe optimizar la adaptación entre los procesos de una organización y las características de cada situación particular, puesto que distintas situaciones y condiciones requieren distintas técnicas de gestión. Esta visión cuestiona el uso de prácticas de gestión universales, y aboga por el uso de diferentes técnicas para tratar con distintas circunstancias conforme surgen.

Hoy en día se hace énfasis en la gestión de calidad y la reingeniería de procesos, buscando por encima de todo la satisfacción del cliente. Se entienden las organizaciones como sistemas complejos adaptativos que interaccionan y evolucionan con sus entornos, y los proyectos como actividades complejas que tienen que adaptarse a las necesidades cambiantes del entorno (por ejemplo, se asume que el carácter del producto siendo construido va a cambiar durante el desarrollo del proyecto). También se habla de una gestión de programas, que consiste en la gestión coordinada de varias proyectos relacionados y la gestión de portafolios, que busca seleccionar los proyectos y programas

[3]En la escuela clásica el trabajador es una pieza a analizar (gestión científica), a evaluar con respecto a su competencia (burocrática) y parte de una jerarquía (administrativa). Parece natural que pronto surjan teorías "más humanas".

[4]Los estudios de Hawthorne, realizados entre 1924 y 1933. Al principio intentaba comprobar si una mejor iluminación del puesto de trabajo conllevaba un incremento de productividad. Para su sorpresa, tanto el grupo de control como el experimental producían más tanto si las luces estaban encendidas como si no, lo que llevó a descubrir que la productividad incrementada era debida a la atención recibida por el grupo.

que mejor permitan cumplir los objetivos estratégicos de una organización dentro de sus capacidades.

### 5.1.4. El triángulo de hierro

El doctor Martin Barnes es el primero que describe el llamado "triángulo de hierro" (figura 5.2), el formdo por el tiempo, el coste y los resultados, en el año 1969. Estos tres elementos siempre han sido importantes, pero la evolución del control del alcance y coste de manera relativamente precisa ocurre con la revolución industrial (siglo XVIII), y la planificación se desarrolla durante el siglo XX, así que no es hasta los años 60 cuando se conectan las tres cosas bajo el paraguas de la disciplina de la gestión de proyectos.

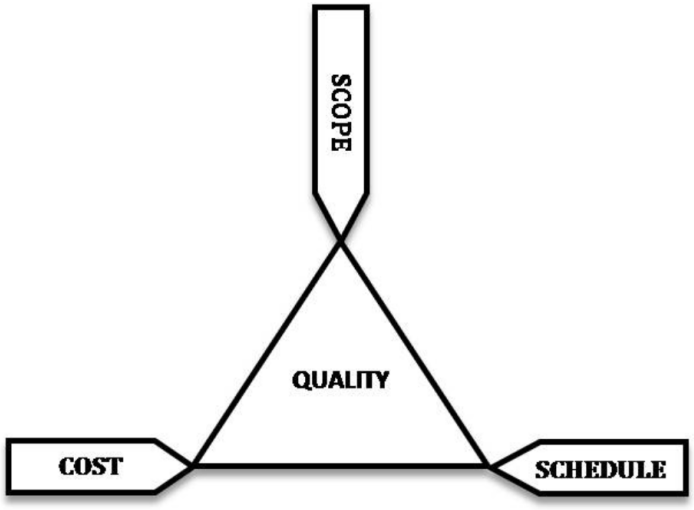



Figura 5.2: El triángulo de hierro (licencia CC BY-SA 3.0 por John Manuel Kennedy T.)

El triángulo de hierro proporciona una metáfora visual, impactante pero simple, que refleja como tiempo, coste y resultados se relacionan y enfrentan entre sí. Por ejemplo, más resultados requieren más tiempo y/o más coste (contratar más personal), menos tiempo implica más coste (contratar más personal o más experimentado) y/o menos resultados o menos presupuesto implica más tiempo (menos personal) y/o menos resultados.

Las relaciones no son lineales ni fácilmente predecibles, por ejemplo duplicar el personal no disminuye el tiempo a la mitad porque se añaden problemas de comunicación, hay que formar al personal nuevo y es casi imposible que pueda subdividir el trabajo de manera óptima para aprovechar al máximo al nuevo personal. También hay que considerar que las personas involucradas en un proyecto tienen prioridades distintas. El profesional primará la calidad, el financiero el coste, la dirección de la empresa los plazos de entrega y los clientes lo querrán todo. Si un proyecto tiene un alcance prefijado, un coste prefijado y un plazo prefijado (proyecto "llave en mano") el equipo que lo desarrolla tiene poco margen de maniobra, y ante cualquier imprevisto o desviación de los planes originales se "romperá" el triángulo: proyecto cancelado,

entrega retrasada, presupuesto inicial superado o resultados por debajo de lo esperado. La alternativa por desgracia suele ser "flexibilizar" la calidad, que generalmente no se especifica con tanta precisión como los otros aspectos del proyecto, lo que se traducirá en fallos en producción, pobre usabilidad, pobre documentación o dificultad de mantenimiento ("calidad flexible").

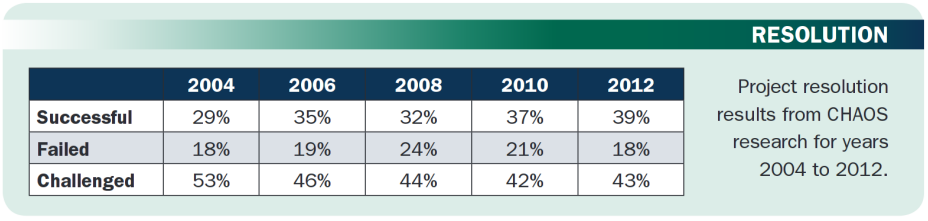
El triángulo de hierro es una característica de los proyectos: en todos los proyectos hay recursos (tiempo y dinero) limitados. En general sería preferible contar con algún grado de flexibilidad (por ejemplo, plazo y presupuesto prefijados con calidad estricta pero alcance variable, o alcance prefijado y calidad estricta, pero plazo y presupuesto variables), y las metodologías ágiles así lo requieren, pero en la práctica el contrato "llave en mano" es seguramente el tipo más extendido en proyectos de software.

El triángulo de hierro puede ser la expresión de los elementos fundamentales que debe abordar la gestión de proyectos, pero no es lo único, y en los últimos años se han ido añadiendo aspectos como la gestión de personal, las comunicaciones, la gestión de riesgos etc.

### 5.1.5. Los proyectos de software en la actualidad

Hoy en día tenemos estándares que describen procesos y objetivos de los tres niveles de gestión (proyectos, programas y portafolios) como los definidos por el *Project Management Institute* (PMI). También se han definido marcos de competencias, certificaciones, másters, doctorados y se aprecia cierta tendencia a pasar de los "directores de proyecto accidentales" que ha habido hasta ahora a los "directores de proyecto aspiracionales", cualificados para eso y con una carrera profesional orientada específicamente.

A pesar de esto, sigue habiendo un importante porcentaje de proyectos de software que fallan en cumplir todas sus expectativas. Algunos datos ampliamente citados al respecto son los incluidos en los denominados *Chaos Reports* compilados y publicados por el Standish Group[5] y basados actualmente en datos de unos 50000 proyectos recopilados desde el año 2002. La figura 5.3 muestra el porcentaje de proyectos exitosos, fallidos y deficientes en los años 2004, 2006, 2008, 2010 y 2012, y aunque se aprecia una leve mejoría, sigue habiendo un número importante de proyectos fallidos y deficientes.



**RESOLUTION**

| | 2004 | 2006 | 2008 | 2010 | 2012 |
|---|---|---|---|---|---|
| **Successful** | 29% | 35% | 32% | 37% | 39% |
| **Failed** | 18% | 19% | 24% | 21% | 18% |
| **Challenged** | 53% | 46% | 44% | 42% | 43% |

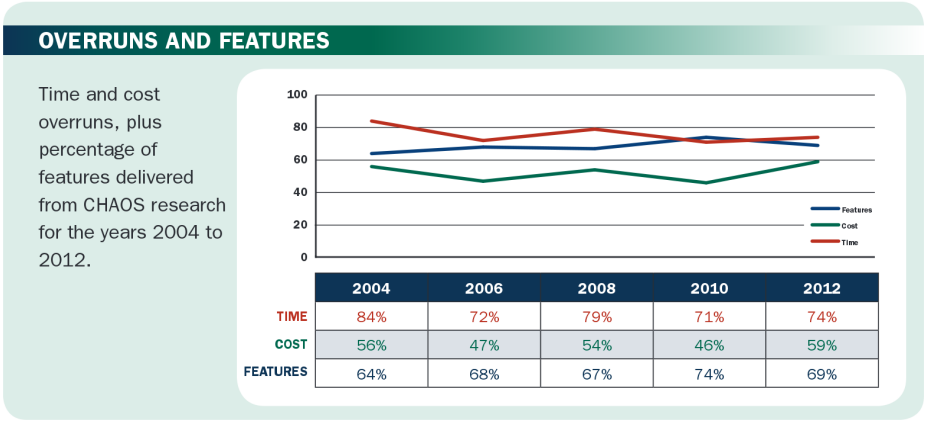
Project resolution results from CHAOS research for years 2004 to 2012.

Figura 5.3: Resolución de proyectos 2004-2012. (c) Standish Group (2013)

La figura 5.4 muestra para los mismos años el porcentaje de tiempo extra necesitado para completar un proyecto, el coste adicional que ha tenido el proyecto y el porcentaje de características que se entregaron. Lo que sí que marca una diferencia es el tamaño

[5] `http://blog.standishgroup.com/`

de los proyectos; la figura 5.5 muestra como los proyectos pequeños, de menos de 1 millón de dólares, tienen éxito muco más a menudo que los grandes, donde todavía hay grandes fracasos.




| | 2004 | 2006 | 2008 | 2010 | 2012 |
|---|---|---|---|---|---|
| TIME | 84% | 72% | 79% | 71% | 74% |
| COST | 56% | 47% | 54% | 46% | 59% |
| FEATURES | 64% | 68% | 67% | 74% | 69% |

Figura 5.4: Sobrecostes, retrasos y características completadas. (c) Standish Group (2013)

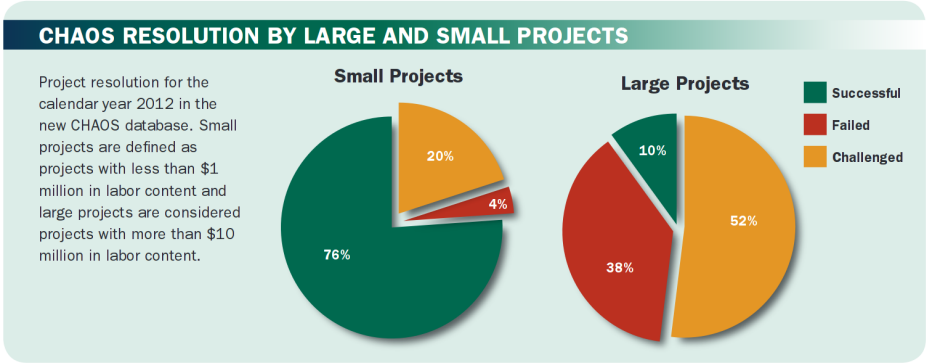



Figura 5.5: Resolución de proyectos: pequeños frente a grandes. (c) Standish Group (2013)

Los datos anteriores se suelen citar como prueba de que existe una "crisis del software" y como argumento de que para superar esta crisis es importante mejorar las técnicas de gestión de proyectos[6]. Sin embargo algunos de sus resultados son discutibles, y discutidos [Eveleens y Verhoef, 2010]:

- Definen exitoso, deficiente y fallido en base a las estimaciones iniciales de coste, tiempo y funcionalidad. Es decir, que miden el éxito o fracaso respecto de la bondad de las estimaciones iniciales, sin tener en cuenta otros factores de éxito, como la utilidad del resultado, los beneficios que proporciona o la satisfacción del usuario, ni la propia naturaleza cambiante del contexto del proyecto.

[6]O las metodologías, técnicas, herramientas, formación...

- Dada esta definición, lo habitual es que para próximos proyectos se sobrestimen costes y plazos y se subestimen las características que pueden completarse. Es decir, este tipo de métrica incentiva estimar peor.

Cuando definimos el éxito en términos de cómo se juzgó realmente el resultado del proyecto, los datos pueden ser otros. A partir de datos recopilados de 173 proyectos en Noviembre y Diciembre de 2013 [Ambler, 2014], preguntando a la organización responsable del proyecto si este fue un éxito o un fracaso, se compilaron los resultados que se ven en las figuras 5.6 y 5.7: solo un porcentaje pequeño de los proyectos se consideró un fracaso, especialmente en aquellos casos en que se seguían metodologías de gestión modernas (*lean* (centrado en el valor para el cliente y la minimización del desperdicio), *agile* (iterativo, ligero, muy colaborativo, auto-organizado y centrado en la calidad), *iterative* (incrementos desarrollados en periodos de tiempo consecutivos), y en general todos los aspectos del proyecto se consideraron exitosos con estas metodologías.

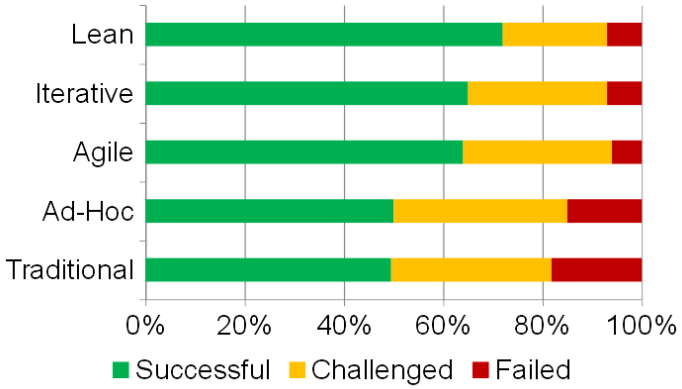



Figura 5.6: Resultado de proyectos según metodologías. (c) 2014 Scott W. Ambler, www.ambysoft.com/surveys/

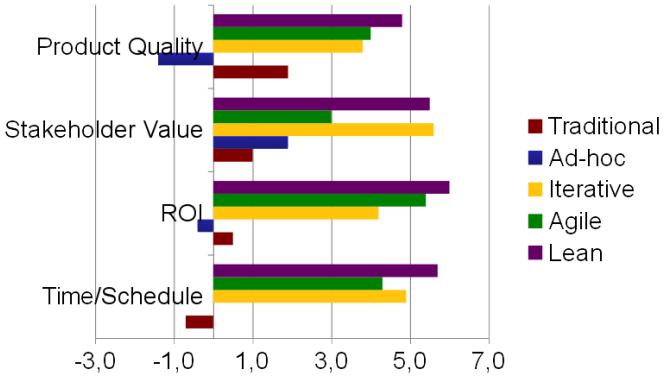



Figura 5.7: Resultado de proyectos desde distintos puntos de vista según metodologías. (c) 2014 Scott W. Ambler, www.ambysoft.com/surveys/

### 5.1.6. Otros tipos de proyectos de ingeniería

Es necesario tomar un poco de perspectiva para ver cómo los proyectos de software se comparan con otro tipo de proyectos de ingeniería para juzgar adecuadamente sus resultados. El hecho es que los sobrecostes y los retrasos son habituales en todo tipo de proyectos de ingeniería, especialmente en los más grandes: un 64 % de proyectos con sobrecostes y un 73 % con retrasos en "megaproyectos" de petróleo y gas [EY, 2014], un 62 % de sobrecoste medio en "megaproyectos" de minería [EY, 2015], estudios que encuentran una clara correlación entre duración y tamaño de los proyectos y los sobrecostes de los mismos en obras públicas en EE.UU. [Shrestha et al., 2013], la construcción del avión Airbus A380 se retrasó de tal manera que en 2009 se habían previsto entregar 120 aviones y se consiguieron entregar solo 10[7], los primeros prototipos del Boeing 787 Dreamliner pesaban 2,3 toneladas más de lo especificado, y 4 años después los prototipos todavía tenían graves problemas (uno de ellos tuvo que hacer un aterrizaje de emergencia durante un vuelo de pruebas por un incendio a bordo)[8], el nuevo submarino S80 de la armada española resultó tener un sobrepeso de entre 75 y 100 toneladas, lo que podía llegar a comprometer su flotabilidad, y se asume que corregirlo supondrá un retraso de entre uno y dos años en la entrega[9]. La consultora Gallup citando datos de diversas fuentes resalta que en un estudio que revisó 10640 proyectos de 200 empresas de varias industrias en 30 países, se concluyó que solo el 2,5 % de las empresas consiguió completar con éxito el 100 % de sus proyectos[10].

A partir de estos datos podemos al menos empezar a intuir que todavía existe un gran margen de mejora en la gestión de proyectos de ingeniería, y que este margen de mejora seguramente tendrá que llegar por distintos frentes (formación, gestión, técnicas, herramientas etc.). También sería aconsejable ser prudentes a la hora de incorporar metodologías, técnicas y herramientas que se usan en otros tipos de proyectos a los de software que, mirados con cierta perspectiva, es posible que no estén tan mal en comparación con otros como algunas veces se da por sentado.

### 5.1.7. El futuro

El primer punto a tener en consideración es la definición de proyecto. Las definiciones actuales, p.ej. "un esfuerzo temporal que busca crear un producto, servicio o resultado único", son en general compatibles con actividades que seguramente no querremos considerar proyectos (por ejemplo hornear un pastel en casa) y siguen muy ligadas a proyectos industriales con resultados "tangibles".

Una propuesta interesante es considerar los proyectos como "organizaciones temporales de conocimiento". El instrumento básico de la gestión de proyectos es el equipo que lo lleva a cabo, y la completa previsibilidad no es una realidad en la gestión de proyectos. También interesantes son las aproximaciones desde la complejidad, que ven

[7] `https://en.wikipedia.org/wiki/Airbus_A380#Production_and_delivery_delays`

[8] `https://en.wikipedia.org/wiki/Boeing_787_Dreamliner#Production_and_delivery_delays`

[9] `https://es.wikipedia.org/wiki/Submarinos_Clase_S-80#Problemas_durante_la_construcci.C3.B3n`

[10] `http://www.gallup.com/businessjournal/152429/cost-bad-project-management.aspx`

al equipo de proyecto como un sistema complejo adaptativo que se adapta a su entorno (competencia, clientes, etc.), y el énfasis en las interacciones entre personas y en la naturaleza participativa y responsiva de los procesos humanos, con las organizaciones siendo, al menos en parte, propiedades emergentes de la interacción entre sus integrantes.

Aceptar estas propuestas más recientes desplaza el foco de la gestión de proyectos desde los propios proyectos hacia la gente involucrada en los mismos. Estas personas son las que crean el proyecto, trabajan en el y lo completan. Los documentos (calendarios, planes etc.) pasan de ser herramientas para tratar de controlar el futuro a mecanismos para la comunicación entre los participantes (clientes, inversores, equipo de proyecto etc.). Estas propuestas también requieren la aceptación de la incertidumbre. Entender que haber escrito un plan detallado no significa que podamos controlar el futuro, sino que simplemente hemos descrito un futuro posible, sujeto a una incertidumbre que tendremos que aprender a gestionar como una realidad inevitable, una característica, y no a tratar de eliminarla como si fuera un defecto solamente atribuible a nuestros fallos.

## 5.2. La gestión de proyectos software en el Computer Science Curricula de ACM/IEEE

El Computer Science Curricula de 2013 (CS2013) [ACM/IEEE-CS Joint Task Force on Computing Curricula, 2013] es un esfuerzo conjunto de la Association for Computing Machinery (ACM) y la IEEE-Computer Society para recopilar un cuerpo de conocimiento sobre lo esencial que debe contener un currículum de informática.

El CS2013 está organizado en 18 áreas de conocimiento, de las que la de Ingeniería del Software sería la que engloba la gestión de proyectos software, aunque este documento avisa de que las áreas están interrelacionadas y tienen dependencias unas de otras.

Según el CS2013, la ingeniería del software es la disciplina que se ocupa de la aplicación de teoría, conocimiento y práctica a la construcción eficiente y efectiva de sistemas software que satisfagan los requisitos de clientes y usuarios. Esta disciplina engloba todas las fases del ciclo de vida de un sistema software, desde la captura de requisitos hasta la operación y mantenimiento, y se preocupa de la mejor forma de construir buenos sistemas software tanto con procesos de desarrollo tradicionales (dirigidos por planes) como ágiles u otros que puedan surgir.

El CS2013 señala que la mejor forma de aprender gran parte del material de ingeniería del software es hacer que los estudiantes participen en proyectos en equipo para desarrollar sistemas software. Una gran parte la ingeniería del software trata sobre la comunicación efectiva entre miembros de un equipo y otros interesados. También indica que cada vez hay más pruebas de que los principios de la ingeniería del software se aprenden de manera más efectiva usando aproximaciones iterativas (metodologías ágiles o ciclos de vida iterativos), donde los estudiantes tienen la oportunidad de valorar su trabajo en un ciclo y aplicar lo aprendido al siguiente.

Dentro del área de conocimiento de ingeniería del software, los componentes fun-

damentales relacionados con la gestión de proyectos como se aborda en este proyecto docente son:

- Procesos Software: se incluyen aquí la interacción de un sistema con su entorno, los distintos ciclos de vida, y aspectos de calidad, mejora y medida de procesos y modelos de madurez.

- Gestión de Proyectos Software: aquí va el grueso de la disciplina, incluyendo participación en equipo (organización, reparto de responsabilidades, toma de decisiones, estructura de una reunión, calendarios de trabajo, roles y responsabilidades, resolución de conflictos, valoración de los individuos y potenciales riesgos en equipos virtuales (comunicación, percepción y estructura)) estimación de esfuerzos, gestión de riesgos (identificación, análisis y evaluación, tolerancia, planificación), planificación y seguimiento, análisis coste/beneficio, métricas y estimaciones y aseguramiento de la calidad.

- Herramientas y entornos: gestión de configuraciones y control de versiones, gestión de lanzamientos e integración continua están recogidos, con otras cosas, aquí.

## 5.3. La gestión y dirección de proyectos de software en ingeniería informática en la universidad española

Como se ha visto en detalle en la Sección 3.1, los estudios de ingeniería informática (grados y másteres oficiales) están regulados en España mediante la resolución 12977/2009 de 8 de Junio en la que el BOE publica un acuerdo del Consejo De Universidades con las competencias y módulos que grados y másteres deben abordar, aunque cada universidad tiene libertad para desarrollar sus planes de estudios bajo esas guías.

Respecto de la gestión de proyectos de software, podemos señalar las siguientes competencias que se espera que adquieran los graduados como relacionadas con esta disciplina (se han resaltado algunos de los términos por considerarse especialmente relevantes):

- "Capacidad para concebir, redactar, **organizar**, **planificar**, desarrollar y firmar **proyectos** en el ámbito de la ingeniería en informática que tengan por objeto, de acuerdo con los conocimientos adquiridos según lo establecido en el apartado 5 de este anexo, la concepción, el desarrollo o la explotación de sistemas, servicios y aplicaciones informáticas.

- Capacidad para **dirigir** las actividades objeto de los proyectos del ámbito de la informática de acuerdo con los conocimientos adquiridos según lo establecido en el apartado 5 de este anexo.

- Capacidad para concebir, desarrollar y mantener sistemas, servicios y aplicaciones informáticas empleando los métodos de la ingeniería del software como instrumento para el **aseguramiento de su calidad**, de acuerdo con los conocimientos adquiridos según lo establecido en el apartado 5 de este anexo.
- Conocimiento y aplicación de elementos básicos de economía y de **gestión de recursos humanos, organización y planificación de proyectos**, así como la **legislación, regulación y normalización en el ámbito de los proyectos informáticos**, de acuerdo con los conocimientos adquiridos según lo establecido en el apartado 5 de este anexo."

De alguien con un título de máster además se espera que tenga las siguientes competencias relacionadas con la disciplina (de nuevo se han resaltado algunos de los términos por considerarse especialmente relevantes). Es de destacar que de las diez competencias que recoge el BOE, seis incluyen aspectos que están dentro del ámbito de la dirección y gestión de proyectos.

- "Capacidad para **proyectar**, calcular y diseñar productos, procesos e instalaciones en todos los ámbitos de la ingeniería informática.
- Capacidad para la **dirección de obras e instalaciones** de sistemas informáticos, cumpliendo la normativa vigente y asegurando la calidad del servicio.
- Capacidad para **dirigir, planificar y supervisar equipos multidisciplinares**.
- Capacidad para la **elaboración, planificación estratégica, dirección, coordinación y gestión técnica y económica de proyectos** en todos los ámbitos de la Ingeniería en Informática siguiendo criterios de calidad y medioambientales.
- Capacidad para la **dirección general, dirección técnica y dirección de proyectos** de investigación, desarrollo e innovación, en empresas y centros tecnológicos, en el ámbito de la Ingeniería Informática.
- Capacidad para aplicar los principios de la **economía y de la gestión de recursos humanos y proyectos, así como la legislación, regulación y normalización** de la informática."

### 5.3.1. En la Universidad de Zaragoza

En el quinto cuatrimestre del grado, se imparte la asignatura obligatoria "Ingeniería del Software" donde se introduce a los estudiantes el concepto de ciclo de vida de un de proyecto de software, pero el principal contenido común de la materia llega en el sexto cuatrimestre con la asignatura "Proyecto Software", descrita en detalle en el capítulo 6. Después, dentro de la especialidad en Ingeniería del Software, en el séptimo cuatrimestre se ofrece la asignatura "Gestión de Proyectos Software", obligatoria de esa especialidad y descrita en detalle en el capítulo 7.

En el máster universitario en ingeniería informática no hay asignaturas sobre gestión de proyectos de software.

### 5.3.2. En otras universidades españolas

El contenido de esta sección se ha obtenido accediendo a las webs de las titulaciones de ingeniería informática (grados y másteres) en algunas de las principales universidades españolas y consultando la información relativa al último curso completado en el momento de redactar esta sección, que es el 2015/2016. Se ha buscado en los programas de las titulaciones asignaturas cercanas a las presentadas en este proyecto docente utilizando como palabras claves de búsqueda "proyectos", "gestión", "dirección" e "ingeniería del software". La inclusión del último término se debe a que en algunas universidades algunos aspectos de la gestión de proyectos se imparten dentro de una "ingeniería del software II" o similar. Se han quedado fuera algunas asignaturas, especialmente de másteres, que podrían considerarse desde una perspectiva amplia como parte de la disciplina de gestión de proyectos, pero que por su carácter muy especializado no ayudaban a proporcionar la visión general que se quiere dar en esta sección.

#### Universitat Politècnica de Catalunya

El grado se da en dos campus, el principal (el que tiene más menciones) en la Facultad de Informática de Barcelona.

En el quinto cuatrimestre, dentro de la mención en Ingeniería del Software se ofrece la asignatura "Gestión de Proyectos de Software", de 6 créditos ECTS, en la que se trata a partes iguales sobre la gestión clásica de Proyectos de Software (predictiva, planificada) ejemplificada con el *Rational Unified Process*, y de la gestión ágil ejemplificada con Scrum, XP y Kanban. Se realizan dos proyectos en grupo, uno con cada tipo de metodología.

En el sexto cuatrimestre, también dentro de la mención en Ingeniería del Software, se imparte "Proyecto de Ingeniería del Software", de 6 créditos ECTS, que es un asignatura de carácter muy práctico que a partir de la asignatura anterior, y otras, propone a los estudiantes la realización de un proyecto en equipo.

En el máster universitario en ingeniería informática no hay asignaturas sobre la gestión de proyectos de software.

#### Universidad Politécnica de Madrid

El grado en ingeniería informática se imparte en la Escuela Técnica Superior de Ingenieros Informáticos (antes Facultad de Informática) y consta de una única especialidad.

La asignatura "Ingeniería del Software II", obligatorio del séptimo cuatrimestre, tiene un temario que puede considerarse de gestión de procesos y proyectos de software: ciclo de vida, trabajo en equipo, estimación y planificación, gestión de configuraciones y gestión de la calidad.[11]

El máster universitario en ingeniería informática tiene una asignatura "Dirección de proyectos" que profundiza en la dirección de proyectos, el papel y las responsabili-

[11]La "Ingeniería del Software I", también obligatoria, está focalizada en el análisis y diseño de sistemas software.

dades de la directora del proyecto, el ciclo de vida, la definición del el alcance (work breakdown structure), la gestión de tiempo y costes, los recursos humanos, el control y seguimiento del proyecto y la gestión de riesgos.

En el grado, se puede mencionar también la asignatura "Gestión de Procesos de Tecnologías de la Información", obligatoria del séptimo cuatrimestre, tiene un temario organizado desde una perspectiva de procesos, con un fuerte énfasis en estándares de calidad (ISO 9000-2008), mejora continua (ISO 15504) y procesos de TI (ISO 20000, CMMMi 1.3), así como en la gestión cuantitativa de los mismos (Six-Sigma, GQM etc.).

**Universitat Politècnica de València**

El grado en ingeniería informática en esta universidad se imparte en dos facultades: en la Escuela Politécnica Superior de Alcoy (con las especialidades de Ingeniería de Computadores, Sistemas de Información y Tecnologías de la Información) y en la Escuela Técnica Superior de Ingeniería Informática que incluye las especialidades de Ingeniería del Software, Ingeniería de Computadores, Computación, Sistemas de Información y Tecnologías de la Información.

La asignatura "Gestión de Proyectos" obligatoria del sexto cuatrimestre incluye temas sobre el papel de director de proyecto, planificación y seguimiento, de alcance, plazos y económicos, gestión de riesgos, comunicación y trabajo en equipo, pliegos de condicioens, calidad de proyectos, y herramientas para la planificación y seguimiento de proyectos. La evaluación incluye la realización de un proyecto en equipo, dividido en dos entregables. Ya dentro de la especialidad de ingeniería del software se puede cursar la asignatura "Proyecto de Ingeniería del Software", donde los estudiantes llevan a cabo un proyecto siguiendo pautas basadas en metodologías ágiles y en tres sprints, la de "Calidad del Software" que incluye los temas de aseguramiento y control de la calidad y métricas y estimaciones de proyectos de software, que en este proyecto docente se consideran dentro de la gestión de proyectos, y "Proceso de Software", donde se describen metodologías de desarrollo de software tanto tradicionales (RUP y Métrica 3) como ágiles, algunos de cuyos componentes son de gestión.

En el máster universitario en ingeniería informática incluye la asignatura obligatoria de "Planificación y Gestión de Proyectos de TI", que abarca la dirección de proyectos (organización del trabajo, plan de proyecto, equipo de trabajo y habilidades para la dirección de proyectos) y de portafolios (contenido que queda más allá del ámbito de lo que se plantea en este proyecto docente, que trata sobre la gestión de proyectos). La asignatura de "Auditoría, Calidad y Gestión de Sistemas de Información" incluye aseguramiento y evaluación de la calidad y auditoría de productos y procesos, elementos que entran dentro de la disciplina de gestión de proyectos.

**Universidad de Granada**

El grado en ingeniería informática en la Universidad de Granada se imparte en dos centros: la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación en Granada (con las especialidades de Computación y Sistemas Inteligentes, Ingeniería de Computadores, Ingeniería del Software, Sistemas de Información y Tecnologías de la

Información), y la Facultad de Educación y Humanidades en Ceuta (con la especialidad en Sistemas de Información).

Algunos, pocos, aspectos de planificación y gestión de proyectos se incluyen en la "Fundamentos de Ingeniería del Software", obligatoria del cuarto cuatrimestre, pero el grueso de la materia se imparte en "Dirección y Gestión de Proyectos", en el sexto cuatrimestre y dentro de la especialidad de Ingeniería del Software, incluyendo los coneptos básicos sobre proyectos, gestores y equipos de desarrollo, planificación, control y cierre de proyectos, y la gestión de costes, recursos humanos, comunicación, calidad, riesgos, adquisición, integración y alcance. Esta asignatura se evalúa en su mayor parte a partir de un proyecto en grupo. También hay una "Metodologías de Desarrollo Ágil" en el séptimo cuatrimestre, y dentro de la especialidad de Ingeniería del Software, pero esta trata las metodologías ágiles desde un punto de vista amplio y por tanto la parte de gestión ágil de proyectos no tiene más que una pequeña parte, en la que se habla de Scrum.

Dentro del máster universitario en ingeniería informática, la asignatura obligatoria "Planificación y Gestión de Proyectos Informáticos", engloba los contenidos fundamentales de la materia de gestión de proyectos, y las técnicas de planificación, estimación, gestión de riesgos, estándares, gestión de la configuración, gestión de personal, visibilidad y control y ciclos de vida de los proyectos informáticos.

# Capítulo 6

# La asignatura de Proyecto Software

Un proyecto de software es una acción emprendida para crear un producto o servicio de software único en un periodo de tiempo delimitado. El fin de un proyecto se alcanza cuando se alcanzan los objetivos del mismo, o cuando se decide que no van a poder cumplirse, o bien que el proyecto ya no es necesario.

Todos los proyectos, de software o no, se realizan con recursos y tiempo limitados. La asignatura destaca la importancia de reconocer este hecho inevitable para poder gestionarlo, es decir para controlarlo, proporciona una visión de las técnicas básicas de gestión de proyectos que se usan para ello, y trata cómo compaginarlas con las tareas específicas del desarrollo de software.

Al finalizar la asignatura, cada estudiante habrá participado en un proyecto de software completo, realizado en equipo, donde habrá tenido que aplicar muchas de las competencias adquiridas hasta ese momento y donde tendrá que aplicar las competencias de gestión que proporciona esta asignatura.

Los resultados de aprendizaje esperados, tomados de la ficha de la asignatura en la EINA, están en la Tabla 6.1, mientras que las competencias que se adquieren o amplían tras cursarla, tomadas de la misma fuente, están en la Tabla 6.2.

Tabla 6.1: Resultados de aprendizaje de Proyecto Software en la EINA (tomados de la ficha de la asignatura)

| **Resultado de aprendizaje** |
|---|
| Conoce cómo diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente. |
| Es capaz de planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social. |
| Comprende la importancia de la negociación, los hábitos de trabajo efectivos, el liderazgo y las habilidades de comunicación en todos los entornos de desarrollo de software. |
| Conoce cómo elaborar el pliego de condiciones técnicas de una instalación informática que cumpla los estándares y normativas vigentes. |
| Conoce cómo llevar a cabo el mantenimiento de sistemas, servicios y aplicaciones informáticas. |
| Conoce los fundamentos básicos de la normativa y la regulación de la informática en los ámbitos nacional, europeo e internacional. |
| Aprecia la necesidad del diálogo permanente y colaborativo. |

Tabla 6.2: Competencias adquiridas en Proyecto Software en la EINA (tomados de la ficha de la asignatura)

| **Competencia** |
|---|
| **Transversales** |
| Concebir, diseñar y desarrollar proyectos de Ingeniería. |
| Planificar, presupuestar, organizar, dirigir y controlar tareas, personas y recursos. |
| Combinar los conocimientos generalistas y los especializados de Ingeniería para generar propuestas innovadoras y competitivas en la actividad profesional. |
| Resolver problemas y tomar decisiones con iniciativa, creatividad y razonamiento crítico. |
| Comunicar y transmitir conocimientos, habilidades y destrezas en castellano y en inglés. |
| Usar las técnicas, habilidades y herramientas de la Ingeniería necesarias para la práctica de la misma. |
| Analizar y valorar el impacto social y medioambiental de las soluciones técnicas actuando con ética, responsabilidad profesional y compromiso social. |
| Trabajar en un grupo multidisciplinar y en un entorno multilingüe. |
| Aprender de forma continuada y desarrollar estrategias de aprendizaje autónomo. |
| **Específicas de Ingeniería del Software** |
| Planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social. |
| Comprender la importancia de la negociación, los hábitos de trabajo efectivos, el liderazgo y las habilidades de comunicación en todos los entornos de desarrollo de software. |
| Conocer la normativa y la regulación de la informática en los ámbitos nacional, europeo e internacional. |

## 6.1. La asignatura en la titulación

En los dos primeros cursos del grado, se adquieren las competencias necesarias para el desarrollo de aplicaciones software pequeñas, fundamentalmente desde el punto de vista técnico y sin dar explícitamente una visión integradora de las mismas. En el quinto cuatrimestre, la asignatura de Ingeniería del Software da a los alumnos las competencias necesarias para llevar lo que han aprendido al desarrollo de aplicaciones de software de tamaño mediano.

La asignatura de Proyecto Software parte de ahí y aporta una visión integradora, basada en los procesos de gestión básicos necesarios para llevar a cabo con éxito proyectos de software medianos o grandes.

## 6.2. El programa de la asignatura

El programa de la asignatura aborda cuatro grandes bloques temáticos que dan una visión general de la disciplina: gestión de proyectos, gestión de configuraciones, gestión de la calidad y normativa y regulación del desarrollo de software. Esto se traduce en los ocho temas que se describen a continuación.

### 6.2.1. Introducción a la gestión de proyectos

En este tema se presenta una breve introducción a la disciplina de la gestión de proyectos, desde su origen a las tendencias que marcan su rumbo actual. También se

presenta una visión general de los proyectos de software hoy en día, las tareas que se incluyen en la gestión (planificar, organizar, liderar y controlar), las dos aproximaciones principales (ad hoc y dirigida por procesos), y los procesos más comunes en proyectos de software, separados en las fases de adquisición, inicio, ejecución y cierre. Finalmente se dan algunas cifras sobre tasas de éxito en proyectos de software para ilustrar el estado actual de la cuestión. Parte de el contenido de este tema está incluído y desarrollado en el Capítulo 5.

**Contenidos**

- Proyectos.
- El origen de las técnicas de gestión de proyectos.
- Evolución de las teorías generales sobre gestión.
- El alcance de la gestión de proyectos.
- La gestión de proyectos hoy.
- Tendencias.
- Proyectos de software: aproximaciones y gestión dirigida por procesos.
- Estado actual de los proyectos de software.

**Bibliografía**

Dado que la bibliografía sobre gestión de proyectos normalmente no le da demasiada importancia a los orígenes de la disciplina, la mayor parte del contenido está sacado de dos artículos, Weaver [2007a] y Weaver [2007b], con algunos datos tomados de Group [2013] y Eveleens y Verhoef [2010]. La visión general de los procesos que se abordan como parte de la gestión de proyectos de software está tomada fundamentalmente de [Chemuturi y Cagley, 2010, ch. 1 y 2, Apéndice A].

### 6.2.2. Gestión de configuraciones

Esta unidad aborda uno de los grandes bloques de la gestión de proyectos de software, que es la gestión de configuraciones o gestión del cambio. El cambio en un proyecto es la norma, y es necesario adoptar una aproximación que permita conocer en cualquier momento el estado de cualquier aspecto del proyecto, y poder consultar toda la historia de los cambios por los que ha ido pasando. La gestión de configuraciones del software es una disciplina, técnica y administrativa, para identificar los elementos de configuración (todo lo que se trata como una entidad atómica desde el punto de vista de la gestión de configuraciones), controlar, registrar e informar sobre los cambios a estos elementos y verificar su completitud y consistencia.

Los elementos de configuración en un proyecto de software pueden ser artefactos de software (ficheros fuente, ejecutables, bibliotecas...), elementos de soporte al desarrollo

(un compilador), documentos (de captura de requisitos, de diseño, manuales de usuario...), un sistema completo desplegado e instalado e incluso elementos de hardware (un computador usado para pruebas, que puede ser virtual). Los elementos de configuración se cambian cuando se realiza una petición de cambio (más o menos formal dependiendo del proceso) y esta es aceptada (de nuevo más o menos formalmente). Los cambios implican nuevas versiones de los elementos de configuración, que son identificadores asignados al estado de un elemento de configuración en un momento concreto (versiones sucesivas difieren en uno o más cambios). Una línea base es una versión de un elemento de configuración revisada formalmente y aprobada, que solo puede cambiarse mediante una petición de cambio formal y es base para futuros desarrollos (puede ser desde un documento de requisitos hasta una versión hecha pública de un producto de software).

Además de identificar elementos de configuración, controlar sus cambios y registrar y analizar el estado de los mismos, la gestión de configuraciones del software suele incluir las auditorías y revisiones (validar las versiones que van a hacerse públicas) y facilitar la construcción del software (desde la compilación hasta el despliegue. Las herramientas básicas para la gestión de configuraciones de software son los sistemas de control de versiones, que mantienen un registro de los cambios en los ficheros controlados, y los gestores de incidencias (*issue trackers*) que mantienen un registro de peticiones de cambios y del estado de las mismas (no asignada, asignada, descartada, completada...), y que muchas veces están integrados con algún sistema de control de versiones.

**Contenidos**

- El cambio en los proyectos de software.
- La gestión de configuraciones del software.
- Conceptos de la gestión de configuraciones del software.
- Actividades de la gestión de configuraciones del software.
- Roles de la gestión de configuraciones del software.
- El plan de la gestión de configuraciones del software.
- Herramientas de la gestión de configuraciones del software.

**Bibliografía**

Material del [Sommerville, 2011, capítulo 25] complementado con algo del [Pressman, 2010, capítulo 22].

### 6.2.3. Gestión de configuraciones. Control de versiones con Git

Dada la importancia de los sistemas de control de versiones en el desarrollo profesional de proyectos de software desde que empiezan a popularizarse a finales de los

años 90 y el impacto que han tenido los sistemas de control distribuidos como Git desde su aparición, es necesario incorporar material sobre esto en cualquier asignatura de gestión de proyectos de software. Este tema hace una revisión de toda la funcionalidad fundamental del sistema de control de versiones distribuido Git, que es uno de los más populares hoy en día, además de ser uno de los más potentes (aunque la curva de aprendizaje es empinada, como ilustra bien la figura 6.1).


Figura 6.1: (c) http://xkcd.com/1597/, under a Creative Commons Attribution Non Commercial 2.5 License

Un proyecto Git tiene tres secciones principales: el repositorio (normalmente en un directorio llamado .git) donde está toda la historia del proyecto desde el inicio, el directorio o árbol de trabajo donde hay una copia de una versión de los ficheros del proyecto que se puede manipular libremente y el índice (*stage*) donde se van añadiendo los cambios que irán en el próximo commit que se introducirá en el repositorio. El trabajo básico consiste en modificar (crear, borrar, cambiar) ficheros en el directorio de trabajo, añadir los cambios al índice y hacer commit de los mismos cuando se quiera (un commit es una instantánea de los ficheros en el stage que se ha añadido al repositorio y es la forma de "grabar" los cambios que queremos tener controlados[1]). Luego se pueden compartir commits con otros repositorios remotos para sincronizar el trabajo con el de otras personas.

Cuando haces un commit, Git hace una instantánea del índice y almacena un objeto que tiene un puntero a esta instantánea y al commit o commits justo anteriores en la historia del repositorio (padre o padres). Una rama es un puntero a un commit y en Git podemos tener múltiples ramas. La rama master apunta por defecto al último commit realizado en el repositorio. Se pueden crear ramas múltiples para aislar unos cambios de otros temporalmente (por ejemplo, para trabajar en añadir una nueva funcionalidad que va a tardar varios días y que hasta que no esté terminada no queremos que quede registrada en la rama principal) y luego se pueden mezclar (en el ejemplo,

[1]Aunque cada commit se debe ver, funcionalmente, como una copia de todos los contenidos del repositorio en un momento determinado, Git los almacena internamente de forma muy eficiente aprovechando que cada commit tiene mucho contenido en común con su padre.

para combinar la nueva funcionalidad con la rama principal). La Figura 6.2 muestra un repositorio Git con una historia de commits, que a partir del commit C2 diverge: se ha creado una nueva rama (llamada funcX) para trabajar en cierta funcionalidad nueva (commits C3 y C5) y mientras tanto se ha corregido un error en la rama principal (commit C4). El commit C6 es un commit de mezcla (*merge commit*), donde se han integrado ambas ramas. Si mezclamos dos ramas que tienen cambios incompatibles (se ha cambiado el mismo fichero en las dos ramas y los cambios no se pueden combinar automáticamente) Git informará de un conflicto de mezcla, que tendremos que resolver.

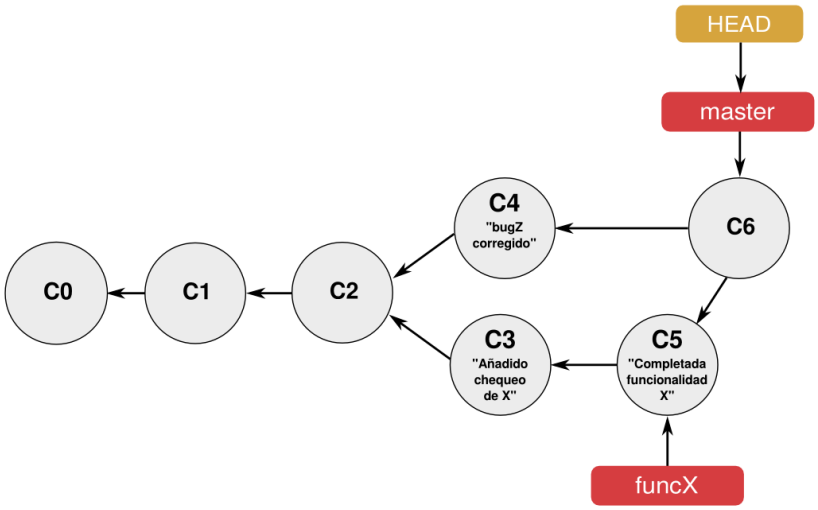



Figura 6.2: Un ejemplo de repositorio Git con dos ramas mezcladas

La característica distintiva de los sistemas distribuidos es que todos los desarrolladores tienen una copia completa del repositorio (con todo su historial de cambios) en su computador y pueden intercambiarse los cambios entre ellos como quieran[2]. En Git un repositorio remoto es cualquiera accesible por la red y al que se tiene acceso (de lectura, o de lectura y escritura). Los repositorios remotos permiten intercambiar cambios con otros desarrolladores subiendo los commits de una rama local a una rama del repositorio (*push*) o bajando los cambios de una rama del remoto a una local (*fetch*) y luego mezclándola con alguna rama local (*merge*), o haciendo ambas cosas a la vez para ramas predefinidas (*pull*).

Para facilitar la organización y el trabajo con repositorios remotos en distintos tipos de proyectos, existen diversos flujos de trabajo que se han ido estableciendo con el tiempo. El básico es el flujo de trabajo centralizado, en el que hay un repositorio central accesible por todos, y cada desarrollador tiene uno local donde trabaja normalmente, y que sincroniza con el centralizado con cierta frecuencia. Este flujo de trabajo es similar al que se sigue con un sistema de control de versiones centralizado, y es adecuado para proyectos en los que trabaja un número no muy grande de personas, muchas veces de la misma organización y con una dedicación similar al proyecto.

[2]En los centralizados típicamente solo hay una copia completa del repositorio en un servidor, y los desarrolladores solo tienen una versión concreta (o parte de una) de lo almacenado allí en sus computadores.

**Contenidos**

- Fundamentos de Git.
- Comandos básicos.
- Trabajo con ramas. Un ejemplo de uso de ramas.
- Trabajo con remotos.
- Ramas remotas.
- *Rebasing*.
- Flujo de trabajo centralizado.
- Para saber más.

Esta unidad se completa con una práctica de laboratorio donde los grupos del proyecto de la asignatura tienen la ocasión de poner en marcha un repositorio Git en GitHub, y probar los comandos básicos que necesitarán a lo largo del curso. GitHub es una herramienta profesional de gestión de proyectos de software, que integra control de versiones, seguimiento de incidencias, gestión de documentación y otras funcionalidades, que ofrece muchas posibilidades como herramienta docente en informática [Lopez-Pellicer et al., 2015].

**Bibliografía**

La mayor parte del material está adaptado de [Chacon y Straub, 2014, capítulos 1, 2, 3 y 5], disponible libremente en `https://www.git-scm.com/book/en/v2`.

### 6.2.4. Proceso de desarrollo y equipo humano

Un proyecto de software comienza con el acuerdo del proyecto, e incluye las actividades técnicas necesarias (cuáles, el orden, la duración y las fronteras entre estas actividades depende del tipo de proceso que se siga). Además de estas actividades, hay un conjunto de tareas de gestión que se realizan de manera continuada en paralelo con ellas, fundamentalmente la gestión del proyecto, la gestión de configuraciones y la gestión de la calidad. Se suele considerar como proceso de desarrollo al que incluye las actividades de obtención del contrato, especificación de requisitos, análisis, diseño, implementación, pruebas, transferencia, mantenimiento y retirada.

Las personas son las encargadas de llevar a cabo, y/o responsables de, todas las actividades de un proyecto. En un proyecto de software son el activo más importante y donde se invierte la mayor parte de los recursos económicos. Normalmente se estructuran en equipos (no muy grandes, entre 5 y 9 personas es lo habitual, para facilitar la comunicación interna y la coordinación). Cada equipo suele tener un líder, que debe tener buenas habilidades de comunicación, organización y resolución de conflictos, además de conocer la tecnología que se usa. Los líderes son los responsables últimos de los resultados de los equipos (positivos y negativos). En algunas metodologías se

prefieren los equipos auto-organizados, donde los liderazgos son cambiantes, decididos internamente y basados en la capacidad para cada tipo de problema. Dentro de un equipo debe haber un reparto más o menos explícito de roles, y lo ideal es que no haya ninguna tarea que dependa de una única persona del equipo.

**Contenidos**

- Procesos de desarrollo: obtención, realización, transferencia, mantenimiento y retirada.
- Equipos humanos: organización y liderazgo.

Esta unidad se complementa con dos ejercicios prácticos:

- Creación de presupuestos: tras dar una pautas muy básicas, se les dan a los estudiantes datos de los módulos y requisitos de un proyecto, los esfuerzos que se han estimado para ellos y algunos requisitos de hardware y viajes, y se les pide elaborar una oferta económica (compromiso con el cliente, no demasiado detallado, basado en funcionalidad/módulos, lo que se va a cobrar) y un presupuesto de costes (detallado, basado en horas de trabajo, estimación de coste real, lo que va a costar).
- Toma de decisiones en equipo: es un juego inspirado por uno similar llevado a cabo en entrenamientos de la NASA, y que consiste en ordenar por prioridad 15 piezas de equipo necesarias para sobrevivir en un escenario de alunizaje forzoso en la Luna, siendo la única opción de salvación alcanzar la base lunar, a 300 kilómetros de distancia. Cada integrante del equipo hace su propia priorización individual, y después en equipo se discute para integrarlas en una única respuesta. Después se compara esa respuesta con la respuesta "óptima" (no es indiscutible que sea la solución óptima puesto que es muy difícil medir la bondad de las distintas opciones, pero sin duda es una de las mejores opciones). En términos generales, las soluciones consensuadas suelen ser mejores que las individuales, incluso sin que los equipos usen ninguna técnica específica, que es lo que se busca transmitir.

**Bibliografía**

Basado fundamentalmente en temas de [Project Management Institute, 2013, capítulos 2 y 9].

### 6.2.5. Métricas y estimaciones

En este tema se presenta una panorámica de las técnicas más frecuentes para la medición y la estimación en proyectos de software, de tamaños, tiempos y costes, tratando de hacer siempre una reflexión sobre la aplicabilidad y la fiabilidad de estas técnicas según los datos disponibles y la fase del proceso en la que se apliquen.

Los proyectos de software siempre tienen incertidumbres, un proyecto siempre crea algo único, y en ellos hay que tomar muchas decisiones basadas en datos (costes,

plazos, tamaños, la ocurrencia o no de ciertos sucesos etc.) que no podemos saber con exactitud. Por tanto, tendremos que hacer estimaciones, predicciones a partir de la información disponible, o bien intentar ser muy flexibles sobre esas decisiones (p.ej., esperar a tomarlas hasta que tengamos información suficiente). En general las estimaciones deberíamos transmitirlas siempre con intervalos de confianza para resaltar la incertidumbre que tenemos sobre ellas, aunque a veces esta incertidumbre será tan difícil de cuantificar como la propia estimación.

Las métricas de software relacionan medidas (indicaciones cuantitativas sobre la extensión, cantidad, dimensión, capacidad o tamaño, que se expresan en algunas unidades y posiblemente tienen algún margen de error) que hacemos sobre el mismo, que buscan proporcionarnos información útil sobre el proceso de desarrollo del software, el proyecto, o el propio producto software. Aunque en los años 90 varias compañías grandes de software introdujeron las métricas de software en sus procesos, hoy en día sigue sin estar claro si se usan habitualmente o si las decisiones tomadas en base a ellas son significativamente mejores. Las métricas más habituales están basadas en los requisitos, y permiten estimar el tamaño[3] del software a construir en las fases tempranas, o bien en el producto que estamos construyendo, que se calculan a partir del código fuente o la documentación de diseño y dan estimaciones más precisas durante el desarrollo. La métrica más habitual basada en requisitos es la de los puntos de función, mientras que las basadas en producto más comunes son las líneas de código, la complejidad ciclomática, y algunas específicas para código orientado a objetos como la suite CK.

Si lo que queremos es hacer una estimación de costes de un proyecto de software, problema muy frecuente, normalmente necesitamos conocer los esfuerzos de desarrollo puesto que el personal es la partida más costosa con diferencia. Los esfuerzos de desarrollo se miden en meses-persona, la cantidad de trabajo hecha por una persona en un mes de trabajo a tiempo completo. Los esfuerzos se pueden estimar en base a modelos empíricos basados en datos históricos propios o públicos, y que se basan en el tamaño estimado del software a construir (que podemos estimar por ejemplo usando puntos de función). El modelo empírico más importante es COCOMO II, aunque si no se parametriza con datos propios suficientes en una organización[4] no será fiable.

Cualquier estimación que queramos realizar se puede hacer en base al juicio de expertos, algo que puede funcionar mejor que los métodos empíricos cuando no se tienen datos suficientes como para parametrizarlos adecuadamente. El juicio de expertos funciona mejor si se tienen en cuenta unas cuantas condiciones y se evitan algunos sesgos cognitivos: que los expertos tengan experiencia relevante, que tengan acceso a la información disponible, y mejor si es en forma gráfica, que apliquen distintos métodos de estimación, que la selección del método de estimación y los datos históricos más útiles se base en un registro de lo que ha funcionado mejor en el pasado y no se haga sobre la marcha, y que se dividan los problemas a estimar en partes más pequeñas cuyas estimaciones luego se puedan agregar fácilmente. El método Delphi es una forma de realizar estimaciones con grupos de expertos evitando algunos sesgos cognitivos que

[3] E indirectamente la complejidad, el esfuerzo de desarrollo y el coste.

[4] Muy pocas organizaciones hacen el número suficiente de proyectos lo bastante parecidos entre sí como para tener datos suficientes para parametrizar adecuadamente los modelos empíricos. Esta es la principal debilidad de los mismos.

se sabe que afectan negativamente a las estimaciones.

**Contenidos**

- Introducción, estimaciones e incertidumbres.
- Métricas de software basadas en requisitos.
- Métricas de software basadas en el producto.
- Estimación de costes con modelos empíricos.
- Estimaciones basadas en el juicio de expertos.
- La técnica Delphi.

Esta unidad se completa con un ejercicio en el que se combinan las técnicas de los puntos de función y la técnica Delphi para estimar el tamaño del software que desarrollan en el proyecto. Dado que la métrica de puntos de función incluye algunos factores subjetivos, se combina con Delphi para que cada equipo alcance un consenso.

**Bibliografía**

La preparación de este tema ha incluido material de Armstrong [2001], de [Pressman, 2010, capítulos 23 y 26], de [Sommerville, 2011, capítulos 23 y 24] y algunos datos de Abran [2010].

### 6.2.6. Planificación, gestión de riesgos, y el plan de gestión del proyecto software

Los planes se diseñan en general para tratar de controlar el futuro a partir de la información que tenemos en el presente: qué querríamos que ocurriera, y en qué orden, teniendo en cuenta lo que queremos hacer, las restricciones de tiempo, coste y recursos disponibles, y los riesgos que pueden surgir.

En el caso de los proyectos de software, lo primero es determinar qué queremos hacer con una granularidad suficiente como para poder hacer un plan inicial. Para ello se diseña una estructura de descomposición del trabajo o EDT (*work breakdown structure*), que es una descomposición jerárquica orientada a la construcción de los resultados del proyecto y estructurada en paquetes de trabajo. Después se estima el tiempo que costará cada paquete de trabajo (dependerá de los recursos y personal que podamos destinar a ellos), se establecen las relaciones de dependencia entre ellos y a partir de esa información se diseña el calendario del proyecto. Este calendario se suele reflejar gráficamente en un diagrama de Gantt. La actividad continua de monitorización y control del tiempo se encarga de chequear periódicamente el estado de los paquetes de trabajo, de manera más formal en los hitos especificados en el calendario del proyecto, y de tomar las medidas necesarias si se está incumpliendo el plan inicial, incluyendo la modificación del mismo.

Los riesgos son eventos que pueden surgir en un proyecto afectando a sus objetivos. Pueden ser positivos (oportunidades) o negativos (amenazas), siendo necesario al menos considerar estos últimos. Los riesgos negativos posibles se identifican a partir de la experiencia y los datos históricos (hay algunas fuentes públicas de tipos de riesgos si no tenemos datos propios). Después hay que asignarles una gravedad en base a la probabilidad de que ocurran y al tamaño del perjuicio que causen, y asegurarse de establecer estrategias para mitigar al menos los más graves (o bien reducir la probabilidad de que surjan, o bien minimizar las consecuencias negativas de los mismos).

El plan de gestión del proyecto de software es la herramienta que permite controlar el proyecto, y es el sitio donde recoger o enlazar todo lo que haya que tener en cuenta en el mismo desde el punto de vista de su gestión. Es responsabilidad de la directora de proyecto el crearlo y mantenerlo actualizado (es un documento vivo), aunque tomará muchos elementos de las prácticas de gestión de la organización en que se encuentre. Típicamente contiene una descripción del proyecto, los elementos a entregar y las fechas debidas, la organización del proyecto (ciclo de vida, roles, identificación de los interesados), el proceso de gestión (prioridades y objetivos, la gestión de riesgos, los mecanismos de monitorización y control, el plan de personal), el proceso técnico (métodos, herramientas y técnicas de software, documentación, enlaces a los documentos de gestión de configuraciones y aseguramiento de la calidad, guía de estándares técnicos utilizados), los paquetes de trabajo, el calendario, el presupuesto del proyecto y, al terminar, un anexo de cierre con datos de interés del proyecto (tamaño del producto realizado, esfuerzo empleado, estadísticas de defectos, riesgos que han ocurrido, lecciones aprendidas etc.).

**Contenidos**

- Planificación.
- Productos y actividades: estructuras de descomposición del trabajo, paquetes e hitos.
- Calendarios y diagramas de Gantt.
- Gestión de riesgos.
- El plan de gestión del proyecto de software.

Esta unidad se complementa con dos ejercicios. El primero es de trabajo en equipo y se denomina "el desafío del marshmallow"[5], en el que los equipos deben construir la estructura más alta posible que se sostenga sola utilizando 20 espagueti, 1 metro de cordel y 1 metro de cinta adhesiva en 18 minutos y con el requisito de que debe ponerse un marshmallow en la cima de la estructura. Este ejercicio, que se ha llevado a cabo miles de veces en todo tipo de entornos desde que lo inventó Tom Wujec, ingeniero de Autodesk, simula algunas condiciones típicas de los proyectos: los recursos son limitados, el tiempo es limitado, hay muchas soluciones posibles, requiere colaboración

[5] `http://marshmallowchallenge.com`

en equipo y, muy interesante, modeliza las asunciones ocultas que hay en todos los proyectos usando el marshmallow[6].

El segundo es un problema de planificación en el que los estudiantes, en grupos de dos o tres, deben diseñar un plan para un proyecto del que ya se han estimado unos esfuerzos iniciales para las fases de análisis, diseño, implementación y pruebas, y que debe ser realizado por un equipo de siete personas con distintos perfiles. Además, existen varios conjuntos de restricciones relacionados con dependencias entre tareas, y vacaciones previstas de los distintos miembros del equipo. Para cada conjunto de restricciones hay que diseñar un plan en forma de diagrama de Gantt y entregarlo.

**Bibliografía**

Basado principalmente en [Sommerville, 2011, capítulo 23], [Pressman, 2010, capítulos 27 y 28], y [Project Management Institute, 2013, capítulos 4.2, 5.4, 6 y 11].

### 6.2.7. Aseguramiento de la calidad. El modelo de madurez CMMI y la norma ISO 9001

Un producto o servicio de calidad es el que hace lo que se ha acordado con el cliente, y/o que satisface sus necesidades. Si además podríamos volver a hacerlo, eso es un indicio de la calidad de nuestro proceso de desarrollo. Para los productos fabricados, la calidad del proceso es el principal determinante de la calidad de los productos. En el caso de los productos diseñados, p.ej. los productos de software, hay otros factores involucrados en su calidad: las restricciones impuestas (coste, tiempo, plazos) y el nivel de conocimiento y experiencia de nuestro personal principalmente. La garantía de calidad es un modelo sistemático y planificado de todo lo necesario para asegurar que un producto responde a los requisitos establecidos.

La gestión de la calidad comprende su planificación, identificar los requisitos y/o estándares de calidad para el proyecto y sus entregables y adaptarlos si es necesario, su aseguramiento, auditar los requisitos de calidad y los resultados de las medidas de control de calidad para asegurar que se usan los estándares de calidad adecuados, y su control, que consiste en monitorizar y registrar los resultados de ejecutar las actividades de calidad para valorar las prestaciones y recomendar los cambios necesarios. El objetivo del aseguramiento de la calidad es adquirir confianza en que los resultados del proyecto se completarán de manera que cumplan los requisitos esperados, mientras que el control de calidad permite identificar las causas de una baja calidad, recomendar acciones para eliminarlas y validar que los entregables y el trabajo de un proyecto cumplen los requisitos. El aseguramiento de la calidad debería usarse durante la planificación y la ejecución del proyecto, mientras que el control se ejecuta durante la ejecución y el cierre del mismo para mostrar que los criterios de calidad establecidos se han alcanzado. Por ejemplo, si el control de la calidad exige realizar ciertas pruebas con cada componente que se completa, el aseguramiento de la calidad consiste en veri-

[6]Es bastante típico que el marshmallow no se añada a la estructura hasta el último momento pensando que no va a dar problemas (asunción oculta), y que la estructura se venga abajo sin remedio en cuanto se coloque

ficar que estas pruebas se hacen siempre que se tienen que hacer, llevar un seguimiento de sus resultados y analizarlos por si hay que cambiar el método de prueba.

Las revisiones y auditorías son el método principal para validar la calidad de un proceso o producto. Consisten en un examen del proceso o producto y su documentación asociada para descubrir problemas potenciales. Hay que planificarlas puesto que consumen recursos (tiempo, personal) del proyecto. Los estándares son normas o requisitos, normalmente especificados en documentos formales, que establecen criterios técnicos uniformes, o bien definen métodos, procesos o prácticas. Son el fundamento de una gestión de calidad efectiva, puesto que determinan con qué tenemos que comparar los resultados del proyecto, intermedios o finales, para determinar muchos aspectos de su calidad. En un proyecto de software podemos definir estándares de documentación, de diseño, de código fuente, de comentarios, de tests y otros. Si podemos revisarlos automáticamente evitaremos que dejen de hacerse por "excesivamente burocráticos" o muy trabajosos. También hay estándares relacionados con los distintos aspectos de la gestión del proyecto.

El CMMI (*Capability Maturity Model Integration*) es un marco para la evaluación y mejora de procesos de desarrollo y mantenimiento de sistemas y proyectos de software desarrollado por el *Software Engineering Institute* de la universidad de Carnegie Mellon. Tiene una versión escalonada, donde a cada organización le corresponde un nivel de madurez del 1 al 5 y una versión continua donde 22 áreas de proceso (de gestión de proceso, de gestión de proyecto, de ingeniería y de soporte) son valoradas del 0 al 5. El CMMI establece unos objetivos abstractos sobre el estado deseable por una organización en cada área de proceso y propone un conjunto de buenas prácticas para alcanzar cada objetivo.

La norma ISO 9001:2008 es un modelo genérico de proceso de calidad para desarrollo de software, que cada organización debe instanciar. La norma establece unos principios generales de calidad, describe procesos de calidad de forma genérica y lista los estándares y procedimientos que deberían estar definidos y documentados en cada organización. Que una organización esté certificada como que cumple esta norma, no significa que el software que produce tenga cierta calidad, o que el software que produce sea mejor que el de una organización no certificada. Lo que certifica es que la organización tiene ciertos procedimientos de gestión de la calidad y que los sigue[7].

**Contenidos**

- Calidad de procesos y productos.
- Gestión de la calidad: planificación, control y aseguramiento.
- Revisiones y auditorías.
- Estándares.
- El modelo CMMI: origen, estructura, niveles de capacidad y de madurez.
- La norma ISO 9001: objetivos, principios y estructura.

[7]Por ejemplo, es posible que estos procedimientos no sean buenos, pero mientras estén definidos y se sigan se cumple la norma.

**Bibliografía**

Material tomado principalmente de [Pressman, 2010, capítulo 30.3], [Sommerville, 2011, capítulos 24 y 26.5] y [Project Management Institute, 2013, capítulo 8].

### 6.2.8. El marco legislativo del software

Esta unidad aborda un tema complicado, pero del que que todo ingeniero que aspire a dirigir proyectos de software debería tener al menos unas ideas básicas porque afecta a muchos aspectos de la gestión de los proyectos. Algunas de las leyes que afectan al trabajo en proyectos de software son:

- La ley 34/2002, de 11 de Julio, de servicios de la sociedad de la información y de comercio electrónico, que regula el régimen jurídico de los servicios de la sociedad de la información y la contratación por vía electrónica, en lo referente a prestadores de servicios e intermediarios en la transmisión de contenidos por las redes de telecomunicaciones.

- La ley 11/2007, de 22 de Junio, de acceso electrónico de los ciudadanos a los servicios públicos, que establece el derecho de los ciudadanos a relacionarse con las administraciones públicas por medios electrónicos y regula el uso básico de las tecnologías de la información en la administración pública.

- El real decreto 4/2010, de 8 de Enero, por el que se regula el esquema nacional de interoperabilidad en el ámbito de la administración electrónica, y que establece criterios, recomendaciones y principios para favorecer el desarrollo de la interoperabilidad en las administraciones públicas.

- La ley orgánica 15/1999, de 13 de Diciembre, de protección de datos de carácter personal. Se define dato de carácter personal como cualquier información relativa a personas físicas identificadas o identificables. La ley es aplicable cuando estos datos están registrados en un soporte físico que los haga susceptibles de tratamiento, y solo están excluidos los ficheros mantenidos por personas físicas para actividades personales o domésticas, los que están sometidos a la normativa sobre materias clasificadas y los establecidos para la investigación del terrorismo y otras formas de delicuencia organizada. Los datos deben recogerse con una finalidad, los ciudadanos deben ser informados y dar un consentimiento explícito, tienen derecho a impugnar sus valores (ya rectificarlos y cancelarlos), existe un deber de secreto y la obligación de registrar los ficheros con datos personales en el registro general de protección de datos. Este registro puede consultarse para saber qué empresas tienen nuestros datos personales, y para qué. También establece limitaciones para la cesión y el acceso por terceros a los datos, y los niveles de seguridad que hay que tener con estos datos.

La propiedad de un software que se ha desarrollado depende de las condiciones contractuales en las que se ha hecho. La propiedad del software es intelectual, los derechos que corresponden a autores y otros titulares sobre sus obras y las prestaciones fruto de su creación, y los hay morales (irrenunciables) y patrimoniales (explotación

económica). Los programas de software pueden registrarse en el registro de la propiedad intelectual. La propiedad industrial, que no se aplica al software en España, es la que concede derechos en exclusiva sobre creaciones inmateriales como diseños industriales, marcas y nombres comerciales, patentes y modelos de utilidad y topografías de semiconductores. Los derechos patrimoniales asociados a la propiedad intelectual se pueden ceder en acuerdos contractuales y licencias. Las licencias son contratos mediante los que se reciben derechos (uso, copia, distribución, estudio, modificación) sobre bienes normalmente intelectuales, y que pueden requerir una contraprestación económica. Las hay privativas, que acotan todas las condiciones de uso, libres, que pueden ser permisivas (sin condiciones) o recíprocas (condiciones a la redistribución) o intermedias. Las licencias de una parte del software pueden afectar al conjunto (por ejemplo la GNU GPL), así que hay que determinar si ese es nuestro caso.

El trabajo para la administración pública, el mayor contratista de software en España, se atiene a unas normas específicas que hay que conocer. En general, los contratos se conceden a través de concursos públicos resueltos en base a unas propuestas que deben adecuarse a unos pliegos de condiciones técnicas y administrativas y que se publican junto a los concursos y los plazos establecidos en los boletines oficiales. Si los contratos son pequeños (menos de 18.000 euros + IVA) se pueden adjudicar directamente, mientras que si están entre 18.001 y 60.000 euros + IVA hay procedimientos algo menos complicados que los concursos, aunque en general tiene que haber varias ofertas de distintos proveedores.

**Contenidos**

- Algunas leyes que reglamentan nuestro trabajo.
- La ley orgánica de protección de datos de carácter personal.
- Propiedad del software.
- Licencias de software.
- Contratación con la administración pública.

**Bibliografía**

Los fundamentos están sacados de Menéndez Mato y Gayo Santa Cecilia [2014], aunque los temas se han complementado consultando las diversas normas, leyes y reglamentos mencionados.

## 6.3. El proyecto en equipo

En esta asignatura tiene un gran peso la realización de un proyecto software en equipo que, desde el punto de vista docente, se lleva a cabo siguiendo la metodología de aprendizaje basado en proyectos (ver Sección 4.4.5). Los estudiantes forman grupos de 8 o 9 personas, que tendrán que desarrollar un pequeño proyecto software a lo

largo del curso. Estos grupos forman equipos colaborativos, como se han descrito en la Sección 4.3.

Los equipos se emparejan formando relaciones cliente-proveedor, de manera que todos los equipos tienen ambos roles. Estos emparejamientos se forman de manera que si un equipo A es proveedor de un equipo B (es decir, el B es cliente de A), el equipo B no sea a su vez el proveedor del equipo A. En versiones anteriores de esta asignatura eran los profesores los que realizaban el papel de clientes de todos, pero esta aproximación da la oportunidad a los estudiantes de asumir este papel, que muchos de ellos también tendrán que desempeñar en su vida profesional (subcontratar, supervisar a proveedores etc.).

En el papel de clientes tendrán que hacer una solicitud de propuestas a partir de una idea inicial, estudiar la propuesta técnica y económica de sus proveedores, negociar un contrato y supervisar el trabajo de los proveedores. En su papel de proveedores, tendrán que hacer una propuesta técnica y económica, negociar el contrato, y llevar a cabo el trabajo necesario para entregar todo lo comprometido a tiempo.

### 6.3.1. Ideas

Una vez formados los equipos, pero antes de establecer las relaciones cliente-proveedor, esto es lo primero que hay que decidir. En su papel de clientes, cada equipo presentará dos ideas en clase sobre proyectos de software que quieren contratar. Las ideas se presentan en formato *elevator pitch*[8]. Todas las ideas son votadas por todos los estudiantes, y la idea ganadora de cada equipo será la que tendrán que encargar a sus proveedores.

### 6.3.2. Solicitud de trabajo

Una vez decidida la idea, la siguiente tarea como clientes es redactar una solicitud de trabajo (*request for proposal*) que la desarolle y concrete un poco. La solicitud de trabajo consiste en un documento donde se establece, normalmente sin mucho detalle, el software que se quiere tener, descrito desde el punto de vista de clientes y usuarios (por ejemplo sin muchos detalles sobre tecnologías concretas). Esta solicitud se hace llegar al equipo proveedor, así como a los profesores, y debe constar de las siguientes secciones:

- Objetivos del sistema.
- Restricciones técnicas.
- Glosario de términos y otros anexos.

### 6.3.3. Propuesta técnica y económica

La primera tarea como proveedores es redactar una propuesta técnica y un presupuesto como respuesta a la solicitud de los clientes. La propuesta debe demostrar que

[8]Breve presentación oral, típicamente con un máximo de dos minutos de duración y orientada a potenciales clientes, usuarios e inversores

los proveedores han entendido la solicitud de trabajo y desarrollar un plan de trabajo que permita completarla a tiempo, con detalle suficiente como para que el cliente pueda evaluar si el prespuesto es adecuado. Esta solicitud se hace llegar al equipo cliente, así como a los profesores, y debe constar de las siguientes secciones:

- Resumen ejecutivo.
- Objetivos del sistema.
- Descripción técnica.
- Plan de trabajo.
- Equipo técnico encargado del proyecto.
- Presupuesto.
- Glosario de términos y otros anexos.

Previamente a la redacción de la propuesta inicial, se les da una clase con pautas para realizarla, especialmente relacionadas con la parte económica donde normalmente tienen las cosas menos claras. Esta clase incluye un ejercicio práctico con una plantilla de hoja de cálculo para que aprendan a hacer una estimación de costes inicial y un presupuesto. La propuesta inicial se refina tras una reunión con los clientes para redactar una definitiva, que será un anexo al contrato.

### 6.3.4. Contrato

Una vez negociada la propuesta técnica y económica entre clientes y proveedores, hay que redactar un contrato, que incluya la propuesta, el presupuesto, la forma de pago y cómo y cuándo se realiza la entrega de resultados. A partir de que el contrato sea firmado por ambas partes, el trabajo está establecido en firme, y esto será lo que se utilice para determinar si los proveedores han alcanzado el éxito en la realización de su proyecto de software.

### 6.3.5. El plan de gestión del proyecto

Antes de empezar el proyecto, los proveedores tienen que redactar el plan de gestión del proyecto, que es el documento que describe cómo se gestionará éste. A partir de una plantilla, cada equipo tiene que redactar el suyo. Algunas de las secciones de la plantilla serán complicadas de rellenar porque son cosas nuevas para los equipos, pero eso no es un problema. Se pide que hagan un esfuerzo inicial, y que luego refinen el documento conforme se avance en la materia del curso, con el objetivo de que la entrega final incluya una versión refinada, corregida y aumentada de este plan. La plantilla que se las propone es razonablemente completa para un proyecto pequeño o mediano y está formada por estas secciones:

- Resumen del proyecto: propósito, alcance, objetivos, entregables e hitos principales.

- Organización del proyecto: el equipo del proyecto, es necesario saber quién está asignado al proyecto (e ir cambiándolo conforme entren o salgan personas al proyecto, aunque en el curso eso no debería pasar), las interfaces con el cliente (cómo se contacta con el cliente, y con qué restricciones), y los roles y responsabilidades (qué hace dentro del proyecto cada miembro del equipo).
- Procesos:
  - De inicio: indicar cómo se lleva a cabo la revisión de estimaciones iniciales, la identificación y asignación de recursos (personal, servidores en cloud, móviles para pruebas...), formación inicial de los miembros del equipo y lanzamiento (asegurarse de que el proyecto puede comenzar porque todo lo necesario para ello está establecido).
  - De ejecución y control: explicar cómo se hace el reparto del trabajo diario entre los miembros del equipo, la ejecución de algunas de las actividades del plan de aseguramiento de la calidad, cómo se aborda la gestión del equipo (moral, resolución de disputas...), la gestión de los clientes (qué tipo de contacto se mantiene con ellos y con qué objetivos), medidas de progreso y monitorización del estado del proyecto (qué se mira/mide, cada cuánto tiempo, qué se hace si se detectan problemas de rendimiento o avance insuficiente o desviaciones respecto al plan inicial...), cómo se hará la entrega de resultados al cliente, y finalmente cómo se hace la documentación (qué documentos se mantienen, cada cuánto y quién los actualiza...).
  - De cierre: cómo se hace el resumen de los resultados del proyecto (horas empleadas realmente, coste que ha tenido el proyecto, grado de satisfacción del cliente, diferencia entre las estimaciones iniciales y los resultados finales), la documentación de buenas y malas prácticas y las lecciones aprendidas, identificación y control de componentes reusables e identificación de nuevos riesgos a tener en cuenta en el futuro.
  - Técnicos: el ciclo de vida del software que se sigue en el proyecto y describir los métodos, herramientas y técnicas necesarios tanto para construir el software (p.ej. herramientas de desarrollo) como dar soporte a los planes descritos en la sección de Planes.
- Planes:
  - Gestión de configuraciones: convenciones de nombres (documentos, código, tablas de BD) y numerado de versiones, responsables de las distintas actividades, recursos (repositorios de código), procedimientos, responsables y medios para llevar a cabo cambios y solicitudes de cambio.
  - Construcción y despliegue: cómo se construye e integra el software, si hay scripts de construcción automatizada o no (y en ese caso cómo se garantiza que todos los participantes compilan igual y con las mismas dependencias), qué se incluye en la construcción (descarga y actualización de dependencias, compilación, ejecución de tests automáticos...) y cada cuánto se construye (compila, integra, prueba) el sistema completo, cómo se despliega el software

más allá de las máquinas de desarrollo (contenedores, máquinas virtuales, servidores en cloud) y cómo se configuran esos entornos (rutas, usuarios y contraseñas, puertos y otros elementos).

- Aseguramiento de la calidad: estándares de código y otros (p.ej. para la documentación técnica), actividades de control de calidad del código que se realizarán (revisiones de código por pares, revisiones de requisitos o diagramas UML por pares, tipos de tests automáticos o manuales que se llevarán a cabo, análisis de causa raíz de problemas recurrentes...).
- Análisis de riesgos: identificación de riesgos, cuantificación/priorización, y estrategias de mitigación de los mismos.
- Calendario del proyecto y división del trabajo: cronograma / diagrama de Gantt al menos a alto nivel, división del trabajo en partes (módulos) y reparto de los mismos entre el equipo de desarrollo, al menos a alto nivel (el reparto de labores concretas en el día a día no se detalla aquí, pero hay que explicar bajo qué criterios y quién/cómo se hace antes). Debe haber una correspondencia con las tareas que aparecen en el diagrama de Gantt (que no necesariamente tiene que ser una relación 1 a 1).

- Glosario de términos y otros Anexos.

Para ayudarles en la redacción de este plan, hay una sesión práctica de dos horas durante las cuales los equipos avanzan en la comprensión de las distintas secciones de este plan, toman algunas decisiones y reparten el trabajo para completarlas. Esta sesión es supervisada por los profesores para que puedan resolver todas las dudas que les puedan surgir.

### 6.3.6. Reuniones con los equipos

A lo largo del curso se tienen ocho reuniones con cada equipo, cada dos semanas aproximadamente. Algunas son específicas y en otras se supervisan los avances de su trabajo y se resuelven sus dudas. Después de cada reunión se les piden ciertos resultados intermedios (como mínimo la entrega de una acta con lo tratado en la reunión) para lo que se les da un plazo de una semana desde la reunión. Cada equipo tiene un profesor asignado para estas reuniones, que realiza una labor de asesoramiento y de evaluación del trabajo de los equipos. Estas reuniones se atienen a las pautas dadas para el trabajo en grupos pequeños con formato de tutoría en la sección 4.4.4.

**Primera: conocer al equipo.** Reunión con equipos en su papel como proveedores. En esta reunión se trata de que el profesor conozca al equipo, y de darles pautas para las primeras tareas que tienen que llevar a cabo, concretamente para la propuesta técnica y económica.

Antecedentes:

- Cada equipo tiene asignado su cliente y sistema a desarrollar.
- Cada equipo ha recibido la solicitud de trabajo de su cliente.

- Se ha hecho sesión en clase de presentación de lo que es la propuesta técnica y económica, así como un ejercicio de presupuesto.

Desarrollo:

- Presentación del equipo.
- Revisar la plantilla de propuesta técnica y económica para ver si están claras todas las secciones.
- Revisar la plantilla del presupuesto para ver si está claro cómo hacerlo.
- Si han avanzado ya en la propuesta técnica y económica revisar los avances. Si no, empezar a trabajar en ella hasta final de la reunión.

**Segunda: reunión de negociación del contrato.** Reunión con representantes (hasta cuatro) de cada equipo cliente con cada equipo proveedor. El objetivo es juntarles para que negocien el contrato.

Antecedentes:

- Los clientes han recibido la propuesta técnica y económica inicial.

Desarrollo:

- El equipo proveedor explica cómo va a construir la solución al problema presentado.
- El equipo cliente y el profesor preguntan dudas sobre cómo se va a construir la solución. El equipo proveedor debe aclararlas.
- El equipo proveedor debe tomar nota de todo lo que se acuerde cambiar sobre la propuesta inicial para modificar el documento de referencia y generar la nueva versión.
- El equipo proveedor explica el presupuesto presentado al cliente justificando su dimensión. El equipo cliente y el profesor preguntan dudas. Si es el caso, se modifica el presupuesto de cara a la entrega del documento final de refrencia.
- El profesor hace hincapié en que el documento final es el contrato que debe cumplir el proveedor.
- Ambos equipos deben elaborar acta de la reunión.

**Tercera: primera reunión de control.** Reunión con equipos en su papel como proveedores. Reunión de seguimiento del trabajo en marcha.

Antecedentes:

- Ya se ha negociado el contrato y entregado la propuesa técnica y económica definitiva.

- Se ha realizado la actividad de creación del plan del proyecto y se ha entregado este.

Desarrollo:

- Verificar que los equipos tienen un mecanismo para controlar y reportar los esfuerzos realizados.
- Presentarles el formulario de seguimiento, y acordar cada cuánto se va a completar (se recomienda cada semana). Deben completarlo entre todos especificando que han hecho cada uno de ellos desde la última vez que lo actualizaron. El objetivo es que quede claro para todos qué está haciendo cada integrante del grupo.
- Revisar la especificación de requisitos del sistema. Si no lo han hecho, dedicarle 15 minutos a que empiecen a trabajarlo.
- Repasar reparto de tareas entre los integrantes del equipo. Si no la han hecho, que empiecen a hacerla en la reunión hasta final de la misma.
- Recordarles que hay una entrega de resultados de la primera iteración.

**Cuarta: segunda reunión de control.** Reunión con equipos en su papel como proveedores. Reunión de seguimiento del trabajo en marcha.

Antecedentes:

- Los requisitos están cerrados.
- La planificación detallada de tareas para la primera iteración está cerrada.
- Están a mitad, más o menos, de la primera iteración.

Desarrollo:

- Verificar que realizan el control de esfuerzos.
- Verificar que completan el seguimiento en los formularios establecidos.
- Revisar el proyecto hasta la fecha, contrastar lo previsto frente a lo real. Calcular horas invertidas y que estimen el estado de completitud del mismo.
- Que informen sobre problemas encontrados y soluciones alcanzadas.
- Que expresen su grado de confianza en cumplir la fecha de la primera iteración y, si no es alto, proponer medidas para corregir esto.

**Quinta: presentación de resultados de la primera iteración.** Reunión con representantes (hasta cuatro) de cada equipo cliente con cada equipo proveedor. El objetivo es juntarles para que los proveedores expliquen el trabajo realizado a los clientes.

Antecedentes:

- Los proveedores han terminado la primera iteración.

Desarrollo:

- El equipo proveedor explica lo que ha hecho en la primera iteración y muestran los resultados del trabajo realizado.
- Los clientes, y el profesor, preguntan dudas sobre lo que hay hecho y lo que está pendiente, que el equipo proveedor debe aclarar.
- Ambos equipos entregan acta de la reunión.

**Sexta: tercera reunión de control.** Reunión con equipos en su papel como proveedores. Reunión de seguimiento del trabajo en marcha.

Antecedentes:

- Se ha completado y entregado la primera iteración.
- Se ha lanzado la segunda iteración.

Desarrollo:

- Verificar que realizan el control de esfuerzos.
- Verificar que completan el seguimiento en los formularios establecidos.
- Revisar el resultado de la primera iteración y posibles desfases. Analizar el por qué de estos desfases.
- Revisar la planificación de la segunda iteración.
- Establecer la fecha de finalización del proyecto (límite a partir del que no se puede cargar ningún esfuerzo al proyecto). Esta fecha debe ser anterior o igual a la de entrega (fijada en el calendario de la asignatura).
- Que informen sobre problemas encontrados y cómo los han solucionado.

**Séptima: cuarta reunión de control.** Reunión con equipos en su papel como proveedores. Reunión de seguimiento del trabajo en marcha.

Antecedentes:

- La planificación de tareas para la segunda iteración está cerrada.
- La segunda iteración está a mitad (más o menos).

Desarrollo:

- Verificar que realizan el control de esfuerzos.
- Verificar que completan el seguimiento en los formularios establecidos.
- Revisar resultados hasta la fecha. Contrastar lo previsto con lo real. Calcular horas empleadas, estimar estado actual y determinar si se van a poder cumplir los objetivos (especialmente las fechas de entrega). Proponer medidas de mitigación si no fuera así.
- Que informen sobre problemas encontrados y cómo los han solucionado.

**Octava: presentación de resultados finales.** Reunión con representantes (hasta cuatro) de cada equipo cliente con cada equipo proveedor. El objetivo es juntarles para que los proveedores expliquen el trabajo realizado a los clientes.

Antecedentes:

- Los proveedores han terminado el desarrollo del sistema (que no el proyecto, todavía no han hecho el cierre del mismo).

Desarrollo:

- El equipo proveedor explica el trabajo realizado, detallando cómo se han cumplido todos los requisitos.
- Los clientes, y el profesor, preguntan dudas sobre lo que hay hecho, que el equipo proveedor debe aclarar.
- Ambos equipos entregan acta de la reunión. El equipo cliente debe emitir una valoración razonada sobre el cumplimiento del contrato por parte del equipo proveedor.

### 6.3.7. Presentación comercial

Cada equipo, en su papel de cliente, debe presentar en el aula ante sus compañeros el resultado del proyecto que han encargado. La presentación a clientes (entre 10 y 15 minutos, según cuantos equipos haya) debe estar orientada a un público no técnico e interesado solo en el valor que el resultado del proyecto les aporta: problemas que soluciona, funcionalidad, parámetros de calidad que pueda percibir el usuario etc. La presentación puede tener un cierto componente comercial: imagen "de marca", comparación con otras soluciones en el mercado, u otros elementos que el equipo considere de interés.

### 6.3.8. Presentación técnica

Cada equipo, en su papel de proveedores, debe presentar en el aula ante sus compañeros el resultado del proyecto que han realizado. La presentación técnica asume que el público son los técnicos que reciben el proyecto (y que tendrán que operarlo, configurarlo en el día a día, resolver dudas a usuarios, hacer copias de seguridad, o incluso mantenerlo y actualizarlo caso de un proyecto en el que se entregue el código fuente). Por tanto esta presentación debe recoger los aspectos técnicos principales del proyecto: arquitectura, modelos de datos, tecnologías empleadas etc.

## 6.4. Metodologías de enseñanza-aprendizaje

Las actividades de enseñanza-aprendizaje de esta asignatura se basan fundamentalmente en:

1. Lecciones magistrales (sección 4.4.1): se usan para transmitir de forma eficiente los contenidos teórico-prácticos de la asignatura. Se trata también de provocar cierto diálogo con los estudiantes, pero dado que es una asignatura bligatoria y los grupos son grandes, esto es más complicado que con grupos pequeños.

2. Resolución de problemas y casos: se trata de aplicar los conceptos teóricos a situaciones pequeñas, que puedan resolverse drante una sesión de problemas por parte de los estudiantes. Algunas veces se usa el formato de simulación o juego (Sección 4.4.4).

3. Desarrollo de un trabajo en equipo siguiendo la metodología de aprendizaje basado en proyectos (sección 4.4.5). Este trabajo, descrito en detalle en la sección 6.3, es apoyado por los profesores en sesiones de tutorías (Sección 4.4.4).

4. Prácticas de laboratorio: para que los estudiantes tengan la oportunidad de aprender a usar sobre el computador algunas herramientas de manera controlada y con la ayuda de un profesor para resolver inmediatamente las dudas que les puedan surgir (Sección 4.4.3). Las prácticas se realizan siguiendo un guion, y en general se trata de que los resultados finales de las mismas se puedan aplicar directamente al proyecto de la asignatura.

5. Todos los años se intenta traer a dos o tres profesionales externos para dar seminarios y charlas de interés dentro de los objetivos de la asignatura.

## 6.5. Planificación temporal de la asignatura

La asignatura está diseñada para ser cursada con una dedicación de 150 horas por parte de los estudiantes (6 ECTS), de las que unas 36 horas son para actividades presenciales (teoría, problemas y prácticas), 15 para estudio y evaluación, y 99 para el trabajo en grupo.

Algunas actividades del proyecto se realizan en formato de prácticas, en seminarios o laboratorios. Los proyectos propuestos serán entregados al finalizar el cuatrimestre.

El programa de la asignatura por semanas (asumiendo 3 horas de actividades en aula por semana) está detallado en la Tabla 6.3. Aunque en la tabla figura una distribución por semanas de las actividades por claridad, algunas actividades ocupan algo más de una semana, y otras pueden moverse algo por festividades u otros hechos variables.

| Semana | Teoría y Problemas | Actividades del Proyecto |
|---|---|---|
| 1 | Presentación + grupos + pautas para elevator pitch | |
| 2 | Gestión de proyectos: introducción | Elevator pitch + votaciones |
| | Preparación de propuesta técnica y económica | Entrega de la solicitud de propuesta (cliente) |
| 3 | Gestión de con guraciones | Reunión 1: conocer al equipo |
| | | Entrega de la propuesta técnica y económica (proveedor) |
| 4 | Gestión de con guraciones. Git | Reunión 2: negociación del contrato |
| | | Entrega del contrato y la propuesta técnica y económica nales |
| 5 | Entorno, proceso de desarrollo y equipo humano | Creación infraestructura de proyectos y uso de GitHub |
| | Ejercicio de realización de reuniones | Ayuda a la preparación del plan de gestión del proyecto |
| | Ejercicio de toma de decisiones en equipo | |
| 6 | Métricas y estimaciones | Reunión 3: primera de seguimiento |
| | Ejercicio de estimación de tamaño de software utilizando Delphi y Puntos de Función | Entrega del plan de gestión del proyecto |
| 7 | Plani cación, gestión de riesgos y el plan de gestión del proyecto software | |
| | El desafío del Marshmallow | |
| | Ejercicio de plani cación | |
| 8 | Aseguramiento de la calidad. CMMI e ISO 9001 | Reunión 4: segunda de seguimiento |
| 9 | El marco legislativo del software | Fin de la primera iteración |
| | | Reunión 5: entrega de resultados de la primera iteración a clientes |
| 10 | | Reunión 6: tercera de seguimiento |
| 11 | | |
| 12 | | Reunión 7: cuarta de seguimiento |
| 13 | Presentación comercial – funcionalidad / demostración | Reunión 8: entrega de resultados del proyecto a clientes |
| | | Fin de la segunda iteración |
| 14 | Presentación técnica | |
| | | |
| | | Entrega del informe nal del proyecto |

Tabla 6.3: Calendario por semanas

## 6.6. La evaluación

La evaluarán de la asignatura tendrá dos partes:

1. Realización y defensa del trabajo en grupo, que valdrá un 80 % de la nota final. El trabajo se evaluará en base a los entregables requeridos (software, documentación técnica y de gestión y presentaciones). La nota individual de cada estudiante parte de la del trabajo, pero se tendrá en consideración la evaluación entre pares dentro del grupo y los esfuerzos individuales dedicados al proyecto (sus informes periódicos, pero también el registro de sus repositorios en GitHub).

    - 55 % como proveedores: un 10 % la propuesta técnica y financiera, el plan de gestión del proyecto, el plan de gestión de configuraciones, la infraestructura técnica de construcción y despliegue y el cumplimiento de los planes y reparto de trabajo en el equipo y un 5 % por la presentación técnica.
    - 25 % como clientes: un 5 % por la solicitud de propuesta, uno 15 % por el contrato y seguimiento del producto contratado y un 5 % por la presentación comercial.

2. Un breve examen teórico escrito sobre los conceptos impartidos en la teoría que valdrá un 20 % de la nota final.

Para superar la asignatura será necesario superar las dos partes por separado. En caso de no alcanzar en alguna de las dos partes una nota de 5 puntos sobre 10, la calificación global en la asignatura será la mínima entre 4.0 y el resultado de ponderar con los porcentajes de cada parte.

## 6.7. Bibliografía comentada

- **Pressman RS (2010). *Software Engineering: A Practitioner's Approach*. McGrawHill Higher Education. McGrawHill, 7th edition:** Uno de los textos básicos para la enseñanza de ingeniería del software, de sus cuatro bloques uno es de gestión de proyectos software y otro sobre gestión de la calidad, así que el libro tiene bastantes contenidos de esta disciplina (métricas, estimaciones, planificación, riesgos, aseguramiento de la calidad etc.). Al libro se le nota un poco que va por la séptima edición en el hecho de que ha ido incorporando temas que no estaban en las primeras ediciones (por ejemplo, metodologías ágiles) que a veces chocan un poco con algunos de los planteamientos más tradicionales que también están ahí.

- **Sommerville I (2011). *Software Engineering*. Addison-Wesley - Pearson, 9th edition:** El otro gran texto básico de enseñanza de ingeniería del software. De sus cuatro bloques, uno es el de gestión de software, donde trata sobre gestión en general y temas de planificación, gestión de la calidad, gestión de configuraciones y mejora de procesos. Aunque tiene muchos temas que lógicamente también están recogidos en el de Pressman [2010], creo que son dos textos

con enfoques que se complementan bien y que entre ambos cubren bastante bien el material básico de ingeniería del software y de gestión de proyectos de software.

- **Chemuturi M, Cagley, Jr. TM (2010). *Mastering Software Project Management. Best Practices, Tools and Techniques.* J. Ross Publishing:** Un libro razonablemente actual con un buen vistazo a los aspectos principales de la gestión de proyectos software desde el punto de vista de la gestión dirigida por procesos, dentro de un marco compuesto por cuatro grandes bloques: adquisición, inicio, ejecución y cierre. El libro no detalla técnicas de estimación o planificación temporal y algunos otros aspectos considerados por los autores lo bastante grandes como para no poder hacerles justicia, pero es una introducción bastante completa y genérica a la materia. Lo peor del libro son las breves incursiones que los autores hacen de algunas partes más técnicas, las más cercanas a la ingeniería del software, por ejemplo el uso de sistemas de control de versiones, que sí que están bastante desactualizadas considerando las herramientas y prácticas más modernas.

- **Brooks FP (1995). *The Mythical Man-month: Essays on Software Engineering.* Essays on software engineering. Addison-Wesley Publishing Company:** Este libro es sobre todo una reflexión sobre las experiencias del autor gestionando el desarrollo del sistema operativo IBM OS/360 en los años 60. El autor propone que los grandes proyectos de programación tienen problemas de gestión distintos de los pequeños, y que estos se deben fundamentalmente a problemas relacionados con la división del trabajo, la comunicación y la problemática de preservar la integridad conceptual del diseño en esta situación. La re-edición de 1995 añade dos capítulos que revisan cómo las ideas originales se han mantenido en general bastante bien en el tiempo. Algunas citas de este libro son casi lengendarias como la que dice que *"the bearing of a child takes nine months, no matter how many women are assigned"*, que es una forma de decir que si una tarea no se puede dividir porque hay restricciones secuenciales, añadir esfuerzo (p.ej. desarrolladores) puede no tener ningún efecto en los plazos de finalización, y otras han quedado con un estatus de ley, como la denominada ley de Brooks, que dice que *"adding manpower to a late software project makes it later"*. El libro está bien escrito y aunque es más un catálogo de problemas que de soluciones, es uno de esos libros que cualquiera que tenga que dirigir un proyecto de software de cierto tamaño debería haber leído.

- **Project Management Institute (2013). *A Guide to the Project Management Body of Knowledge (PMBOK Guide).* Project Management Institute, Inc., 5th edition:** este libro proporciona mucha información sobre la gestión de proyectos, y contiene un estándar y una guía reconocidos para la profesión. No es especifico de proyectos de software aunque, como era de esperar, casi todos los contenidos se pueden aplicar a este tipo de proyectos. El libro establece un marco concreto sobre lo que engloba la gestión de proyectos y proporciona definiciones cuidadosas de los términos que emplea, lo que lo convierte en un recurso valioso para la docencia. Por otra parte es un poco duro de leer; por ejemplo, muchos de los términos son largos, relativamente parecidos entre sí y

resultan pesados al aparecer frecuentemente (p.ej. "project management process groups", grupos de procesos de gestión de proyectos). Aunque el libro no entra en mucha profundidad en las técnicas, y desde luego no es un libro orientado a la docencia (no tiene ejercicios, y no muchos ejemplos), es un buen recurso que organiza casi toda la materia de gestión de proyectos y, como mínimo, da muchos punteros sobre las técnicas empleadas en las distintas áreas de la misma.

- **Chacon S, Straub B (2014). *Pro Git.* Apress, second edition:** Un buen libro y bastante actualizado sobre Git, que además está disponible libremente. Libros y documentación sobre Git hay muchos y, dado que es una herramienta compleja, dominarlo normalmente requerirá tener y consultar varios de ellos. Este es una buena elección como punto de partida, está bastante bien organizado y escrito y es lo bastante completo como para cubrir las necesidades de muchos de los usuarios de este sistema de control de versiones.

# Capítulo 7

# La asignatura de Gestión de Proyecto Software

Un proyecto de software es una acción emprendida para crear un producto o servicio de software único en un periodo de tiempo delimitado. El fin de un proyecto se alcanza cuando se alcanzan los objetivos del mismo, o cuando se decide que no van a poder cumplirse, o bien que el proyecto ya no es necesario. El proyecto comprende actividades técnicas y de gestión.

Cada proyecto de software crea un resultado único, aunque es posible que haya algunas tareas repetitivas en algunos de los entregables y actividades del mismo. Esto implica diferencias con todos los proyectos realizados anteriormente, lo que conlleva incertidumbre.

El objetivo de esta asignatura es que los estudiantes tengan la posibilidad de llevar a cabo un proyecto software aplicando algunas técnicas diseñadas explícitamente para trabajar bajo esta incertidumbre. También se profundizará en algunas técnicas de gestión que se han introducido en la asignatura de Proyecto Software. Los resultados de aprendizaje esperados, tomados de la ficha de la asignatura en la EINA, están en la Tabla 7.1, mientras que las competencias que se adquieren o amplían tras cursarla, tomadas de la misma fuente, están en la Tabla 7.2.

Tabla 7.1: Resultados de aprendizaje de Gestión de Proyecto Software en la EINA (tomados de la ficha de la asignatura)

| **Resultado de aprendizaje** |
|---|
| Conoce estrategias y aproximaciones para desarrollar y gestionar los procesos vinculados a la obtención de un contrato de un proyecto software. Esto incluye aproximaciones para la definición de objetivos y entregables de un proyecto, estimación del coste del proyecto y la elaboración de un presupuesto para el mismo. |
| Conoce las bases para abordar la gestión y optimización del equipo humano que integra el proyecto. |
| Esto incluye estrategias para la formación del equipo, herramientas para optimizar su funcionamiento (basadas principalmente en dinámicas de grupo), y aproximaciones a la identificación, caracterización y asignación de roles dentro de un proyecto. |
| Conoce el concepto de riesgo dentro de un proyecto software, así como mecanismos para la planificación de su gestión. Estos mecanismos comprenden, entre otros elementos, la identificación, valoración, selección y definición de estrategias de mitigación. |
| Conoce las bases conceptuales y diversas técnicas para el seguimiento, revisión y evaluación de un proyecto software. |
| Conoce procedimientos para llevar a cabo el cierre de un proyecto software, las implicaciones que esto tiene, la medición y evaluación de un proyecto, así como el aprovechamiento de la información generada estos procesos. |

Tabla 7.1: Resultados de aprendizaje de Gestión de Proyecto Software en la EINA (tomados de la ficha de la asignatura)

| **Resultado de aprendizaje** |
|---|
| Conoce las problemáticas asociadas al mantenimiento del software. |
| Sabe gestionar y organizar las actividades involucradas en el mantenimiento del software. |
| Conoce los aspectos éticos, sociales, legales y económicos intrínsecos al desarrollo de un proyecto software de empresa, generales y específicos al ámbito de uno o varios dominios de aplicación. |
| Conoce una infraestructura de procesos y herramientas necesarios para desarrollar un proyecto software, basado en las buenas prácticas de ingeniería de software disponible en un entorno empresarial de factoría de software. |
| Pone en práctica los conocimientos adquiridos en las asignaturas de la intensificación de Ingeniería de Software en un proyecto concreto desarrollado en equipo: requisitos, análisis, diseño, pruebas (verificación y validación), gestión de proyectos. |

Tabla 7.2: Competencias adquiridas en Gestión de Proyecto Software en la EINA (tomados de la ficha de la asignatura)

| **Competencia** |
|---|
| **Transversales** |
| CT1. Capacidad para concebir, diseñar y desarrollar proyectos de Ingeniería. |
| CT2. Capacidad para planificar, presupuestar, organizar, dirigir y controlar tareas, personas y recursos. |
| CT4. Capacidad para resolver problemas y tomar decisiones con iniciativa, creatividad y razonamiento crítico. |
| CT7. Capacidad para analizar y valorar el impacto social y medioambiental de las soluciones técnicas actuando con ética, responsabilidad profesional y compromiso social. |
| CT8. Capacidad para trabajar en un grupo multidisciplinar y en un entorno multilingüe. |
| **Específicas de Ingeniería Informática y de Ingeniería del Software** |
| CGC2. Capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social. |
| CGC3. Capacidad para comprender la importancia de la negociación, los hábitos de trabajo efectivos, el liderazgo y las habilidades de comunicación en todos los entornos de desarrollo de software. |
| CGC4. Capacidad para elaborar el pliego de condiciones técnicas de una instalación informática que cumpla los estándares y normativas vigentes. |
| CGC8. Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados. |
| CEIS2. Capacidad para valorar las necesidades del cliente y especificar los requisitos software para satisfacer estas necesidades, reconciliando objetivos en conflicto mediante la búsqueda de compromisos aceptables dentro de las limitaciones derivadas del coste, del tiempo, de la existencia de sistemas ya desarrollados y de las propias organizaciones. |
| CEIS3. Capacidad de dar solución a problemas de integración en función de las estrategias, estándares y tecnologías disponibles. |
| CEIS5. Capacidad de identificar, evaluar y gestionar los riesgos potenciales asociados que pudieran presentarse. |
| CEIS6. Capacidad para diseñar soluciones apropiadas en uno o más dominios de aplicación utilizando métodos de la ingeniería del software que integren aspectos éticos, sociales, legales y económicos. |

## 7.1. La asignatura en la titulación

En los dos primeros cursos del grado, se adquieren las competencias necesarias para el desarrollo de aplicaciones software pequeñas, fundamentalmente desde el punto de vista técnico y sin dar explícitamente una visión integradora de las mismas. En el quinto cuatrimestre, la asignatura de Ingeniería del Software da a los alumnos las

competencias necesarias para llevar lo que han aprendido al desarrollo de aplicaciones de software de tamaño mediano. En el sexto cuatrimestre, la asignatura Proyecto Software proporciona una visión integradora de lo aprendido apoyada en las actividades de gestión necesarias para llevar a cabo con éxito un proyecto de software, teniendo en cuenta el coste, los plazos, el alcance, la calidad, los riesgos y los recursos disponibles.

Esta asignatura por una parte profundiza en algunas de las técnicas vistas en Proyecto Software, aplicables a muchos tipos de proyectos, y por otra parte proporciona técnicas adecuadas para proyectos en los que existe un grado de incertidumbre mayor de la habitual, aunque cada vez más se usan para todo tipo de proyectos.

## 7.2. El programa de la asignatura

El programa de la asignatura se compone de dos grandes bloques. Uno sobre la metodología ágil Scrum, compuesto a su vez por doce unidades didácticas, y otro sobre técnicas generales de gestión de proyectos compuesto por cuatro unidades didácticas.

El material de soporte a las clases (transparencias) de casi todo el bloque de Scrum está publicado bajo una licencia Creative Commons y disponible para su descarga en `https://github.com/UNIZAR-30248-GeProSoft/scrum-slides`.

### 7.2.1. El planteamiento de producto

El objetivo de este breve tema es proporcionar a los estudiantes unas pautas básicas para la presentación informal de una propuesta de producto software, con el objetivo de que cada uno de ellos pueda preparar su propuesta para el trabajo que realizarán en la asignatura.

#### Contenidos

- Elegir un producto.
- Elegir un nombre.
- Declaración de objetivos y requisitos iniciales.
- Licencia.
- El plan de producto.

#### Bibliografía

Se toman ideas de los capítulos 2 y 10 de Fogel [2005] y de los capítulos 2 y 4 de 37Signals [2006].

### 7.2.2. Scrum 1 - Introducción

Scrum es una aproximación ágil para desarrollar productos y servicios innovadores, no necesariamente software. Se parte de una lista priorizada de requisitos (la pila del producto), y se va trabajando primero en los más importantes para el cliente. El trabajo se realiza en iteraciones cortas de tamaño fijo (sprints). El objetivo es que al final de cada sprint haya un incremento de producto potencialmente entregable, es decir, un avance que se pudiera poner en manos de los usuarios inmediatamente. Se trabaja en equipos de, normalmente, entre 5 y 9 personas. Los equipos están formados por un dueño de producto (representa a clientes, usuarios u otros interesados), un Scrum master (hace lo posible por eliminar obstáculos al desarrollo y asesora al equipo sobre el proceso Scrum) y el equipo de desarrollo, que son las personas encargadas de llevar a cabo el proyecto.

El origen está en algunas formas de organización del trabajo de empresas japonesas en los años 80, que se llevan al desarrollo de software en los años 90. Scrum está especialmente adaptado a problemas complejos: cosas más impredecibles que predecibles y es necesario explorar el problema para comprenderlo y proponer soluciones creativas e innovadoras. Aunque se puede usar para otro tipo de problemas, seguramente habrá mejores técnicas para ellos.

#### Contenidos

- ¿Qué es Scrum?
- Origen y razón de ser.
- ¿Cuándo elegir Scrum?
- Componentes de Scrum: roles, actividades y artefactos.

#### Bibliografía

Basado principalmente en los capítulos 1 y 2 de Rubin [2012].

### 7.2.3. Scrum 2 - Principios ágiles

La visión tradicional de los proyectos, también llamada dirigida por planes (*plan-driven*), secuencial, prescriptiva o predictiva, se basa en la idea de un gran plan, que abarque todo el proyecto y tenga en cuenta todo lo necesario para llegar de unos requisitos a un producto final dentro de unas restricciones (coste, tiempo, recursos etc.). Aunque ninguna metodología de gestión actual considera el plan inicial como algo inmutable, se acepta que la re-planificación es una tarea de carácter continuo, las metodologías ágiles toman la incertidumbre y la variabilidad como algo que ayuda a desarrollar el mejor producto posible en un entorno complejo y que cambia rápido, y no algo contra lo que hay que ir luchando.

Para aprovechar la variabilidad, las metodologías ágiles trabajan en iteraciones cortas que crean muchas oportunidades de planificación detallada (pero a muy corto plazo) y realimentación, para poder adaptarse a los cambios. También se favorece la

exploración de los problemas, el no trabajar con asunciones sin validar, la integración frecuente del producto, medir el progreso solo por lo que se ha entregado y ha sido aceptado por el cliente y trabajar con elevados criterios de calidad de producto desde el primer momento. Finalmente, las metodologías ágiles buscan siempre formas de eliminar el desperdicio, que es el trabajo que no aporta valor al cliente (directa o indirectamente).

**Contenidos**

- Desarrollo dirigido por planes.
- Variabilidad e incertidumbre.
- Predicción y adaptación.
- Trabajo en marcha.
- Progreso.
- Rendimiento.

Este tema se completa con un ejercicio en clase, denominado "passing pennies" o "the penny game", en el que los estudiantes recuerdan la importancia de una buena paralelización del trabajo para mejorar el rendimiento, algo que las metodologías ágiles tienen en cuenta de diferentes maneras (visibilidad del trabajo en marcha, gestión activa del flujo de trabajo (p.ej. medidas del tiempo total en que cada tarea pasa del estado "propuesta" al estado "completada"), límites explícitos del trabajo en marcha etc.).

**Bibliografía**

Basado principalmente en el capítulo 3 de Rubin [2012]. El ejercicio "the penny game" está adaptado del descrito en `http://tastycupcakes.org/2013/05/the-penny-game/`.

### 7.2.4. Scrum 3 - Sprints

Los sprints en Scrum son iteraciones cortas y de tamaño prefijado (típicamente entre una semana y un mes), con un objetivo que no se cambia hasta que no termina el sprint y donde el equipo Scrum realiza el trabajo del proyecto. Se dividen en cuatro actividades: planificación, ejecución, revisión y retrospectiva. Cada sprint se planifica seleccionando las entradas de la pila que se llevarán a cabo y dividiéndolas en tareas (que puedan ser realizadas por una o dos personas en unas horas), se lleva a cabo, y se completa presentadno los avances en el producto a los clientes (revisión) y los problemas en el proceso con el equipo Scrum (retrospectiva).

Los sprints son de duración limitada para obligar a priorizar y no tener nunca empezadas muchas cosas (es decir, para limitar el WIP (*work in process*)), cortos para tener oportunidades frecuentes para la adaptación a cambios y de duración consistente para facilitar la planificación. En un sprint hay un compromiso con el cliente de que éste

no cambie los objetivos del sprint (cambiar a mitad siempre implica desperdiciar parte del trabajo realizado), y a cambio se le ofrecen múltiples oportunidades de realizar cambios en el proyecto en la planificación de cada sprint (el cambio puede ser frecuente y a la vez estar controlado).

Al finalizar cada sprint debe haber un incremento de producto potencialmente entregable. Esto significa que el equipo considera que es lo bastante bueno como para ponerlo en producción. Para asegurarse de ello, cada equipo tiene que definir lo que significa para ellos "hecho": completado, código revisado, pasa los tests, se ha desplegado con éxito en un entorno como el de producción, aceptado por el dueño de producto (un representante del cliente) etc. Algo que no pasa la definición de hecho, no se considera realmente terminado y vuelve a la pila del producto para completarlo en otro sprint.

**Contenidos**

- Sprints.
- Duración limitada.
- Duración corta.
- Duración consistente.
- No se cambian los objetivos.
- Definición de hecho.

**Bibliografía**

Basado principalmente en el capítulo 4 de Rubin [2012].

### 7.2.5. Scrum 4 - Requisitos e historias de usuario

En los proyectos dirigidos por planes (tradicionales), se trata de tener unos requisitos inmutables y completos lo antes posible. En Scrum se negocian continuamente, y solo se terminan de detallar cuando van a llevarse a cabo. Lo que en las metodologías tradicionales es un inconveniente que se trata de evitar (cambios en los requisitos), en Scrum es un grado de libertad que se puede manipular para alcanzar los objetivos de negocios. Cuanto más innovador sea tu producto, más beneficio se le puede sacar a esta aproximación.

Scrum no exige un formato específico para los requisitos, pero uno habitual en proyectos ágiles es el de las historias de usuario. Su formato típico es "Como [rol o usuario] quiero [objetivo] para [beneficio]" y suelen limitarse a lo que puede escribirse en una tarjeta pequeña. Este formato se complementa con documentos y criterios de aceptación, pero esencialmente es un recordatorio de que hay que tener una conversación con el cliente cuando se vaya a a implementar el requisito que expresa (o cuando se vaya a estimar o a priorizar...), y que habrá que negociar una forma de confirmar que el requisito se ha implementado correctamente (p.ej., unos criterios de aceptación).

**Contenidos**

- Requisitos.
- Historias de usuario: las tres "C" (Carta (*Card*, tarjeta), Conversación, Confirmación), nivel de detalle y los criterios "INVEST" (Independiente, Negociable, Valiosa, Estimable, pequeña (*Small*) y Testeable).
- Requisitos no funcionales y adquisición de conocimiento.
- Conseguir historias.

Este tema se completa con un ejercicio en clase, denominado "story spines", en el que los estudiantes tienen que completar una historia (sobre cualquier tema) de manera colaborativa a partir de un esqueleto genérico de la misma. Este ejercicio refuerza la idea de la creación colaborativa de las historias de usuario, en las que el equipo Scrum y los clientes trabajan conjuntamente para definirlas, y para completarlas y aclararlas cuando es necesario.

**Bibliografía**

Basado principalmente en el capítulo 5 de Rubin [2012]. El ejercicio "story spines" es una adaptación del descrito en `http://tastycupcakes.org/2012/10/story-spines/`.

### 7.2.6. Scrum 5 - La pila del producto

Los requisitos, y en general toda la funcionalidad que falta por implementar, así como defectos conocidos que hay que corregir, mejoras o tareas de exploración pendientes (p.ej. probar dos componentes que no se han usado antes y de los que uno se quiere incorporar), se expresan en Scrum en las entradas de la pila del producto (*product backlog items*, PBI).

La pila del producto es una lista priorizada de todas estas labores a realizar, y aunque no funciona estrictamente como una estructura de datos tipo pila, las entradas más cercanas a la cima son más prioritarias, estarán más detalladas y mejor estimadas, y se llevarán a cabo antes, mientras que las de más abajo serán lo opuesto. La tarea de refinar, estimar, completar y re-priorizar entradas de la pila (el *grooming*) se hace de manera continua y oportunista, puesto que la pila es una estructura que va cambiando continuamente conforme avanza un proyecto.

La pila del producto es esencial tanto para la planificación a largo plazo (que Scrum no requiere pero que se puede hacer de forma compatible con los principios ágiles) como para la planificación de los sprints, pues son las entradas más cercanas a la cima entre las que típicamente se elegirán las que van al siguiente sprint.

En proyectos de tamaño mediano que involucren a varios equipos Scrum es posible tener más de una pila. La regla básica es "un producto, una pila", pero no hay una definición de producto estricta, y dependiendo de si organizamos a los equipos por subproductos, por áreas o bien si conseguimos que sean totalmente intercambiables, podemos decidir tener pilas de subproducto o de área que sean "vistas parciales" de la pila del producto principal.

**Contenidos**

- La pila del producto.
- Los criterios "DEEP" para entradas de la pila (Detalladas apropiadamente, Emergentes, Estimadas y Priorizadas).
- Preparación (*grooming*).
- Gestión del flujo de trabajo.
- ¿Cuántas pilas?

**Bibliografía**

Basado principalmente en el capítulo 6 de Rubin [2012].

### 7.2.7. Scrum 6 - Estimación y velocidad

En Scrum, generalmente se estima el tamaño o esfuerzo requerido de lo que se va a construir, se calcula la velocidad a la que se trabaja y, a partir de ahí, se estima el tiempo que va a costar terminar un proyecto. Estimar es una tarea de todo el equipo Scrum.

Se parte de la estimación del tamaño/esfuerzo requerido de las entradas de la pila del producto. Las entradas más prioritarias deben estar mejor estimadas, y las menos prioritarias pueden estar estimadas de manera muy aproximada hasta que vayan escalando posiciones en la pila. Las entradas más altas normalmente se estiman en puntos de historia (no es un requisito, pero es común), que es una medida relativa. En general es más fácil estimar con medidas relativas (esta entrada costará más que esta, pero menos que esta otra) que absolutas (esta entrada costará 8 semanas).

Se suele usar una escala numérica para los puntos de historia en la que no están todos los números para favorecer la toma de decisiones por consenso frente a los compromisos forzados (si Ana cree que son 3 puntos de historia y Blas cree que son 5, lo más fácil es forzar un compromiso y decir que son 4, pero si la escala no lo permite, tendrán que discutir hasta tomar una decisión consensuada). Muchas veces se usa la técnica del póquer de planificación, basada en el método Delphi, para hacer las estimaciones en equipo, pero no es un requisito de Scrum.

La velocidad de un equipo en un sprint en Scrum es la suma de las estimaciones de las PBI completadas en ese sprint por ese equipo. La velocidad de un equipo es la media de la velocidad que han obtenido en los sprints del proyecto que han llevado a cabo. Esa velocidad se puede usar para ver el progreso de un equipo (como herramienta de diagnóstico) en el tiempo o hacer estimaciones (quedan 120 puntos de historia por implementar, el equipo implementa 40 por sprint, nos quedan 3 sprints) pero, por la naturaleza relativa de los puntos de historia, no se debe usar para comparar equipos (no es una medida de productividad). Si se usa para planificar, es mejor dar la velocidad como un rango (entre 35 y 45 puntos de historia por sprint), lo que transmite mejor la incertidumbre que tenemos.

**Contenidos**

- Velocidad.
- Estimaciones.
- Estimar entradas de la pila (*product backlog items*).
- Póquer de planificación.
- Velocidad.

Este tema se completa con un ejercicio en clase en el que los estudiantes aplican la técnica del póquer de planificación a algunas de las entradas de la pila del producto del proyecto que están creando.

**Bibliografía**

Basado principalmente en el capítulo 7 de Rubin [2012].

### 7.2.8. Scrum 7 - Planificación

Aunque a veces se dice que en Scrum se planifica poco o nada, lo cierto es que puede que se planifique más que con las metodologías tradicionales. La diferencia crucial es que la planificación se hace lo más tarde posible (*just-in-time*) en lugar de tratar de hacerla casi toda al principio (que es cuando menos información se tiene), y que se hace más planificación a corto plazo y menos a largo plazo.

Respecto a la planificación de medio-largo plazo (la de corto plazo se ve en la siguiente unidad), en Scrum no se requiere hacer nada, pero lo normal es que las empresas donde se llevan a cabo los proyectos (o los clientes) la requieran, y se puede hacer de manera compatible con los principios ágiles. Hay que considerar que se planifica a distintos niveles (horizontes temporales), y que la planificación a largo plazo necesariamente será más imprecisa por falta de información (y que por tanto, es un desperdicio hacer un gran esfuerzo en planificar con detalle a largo plazo).

La planificación de producto captura las características esenciales del producto a construir (en forma de visión y pila del producto de alto nivel), y establece unos plazos aproximados de lanzamiento de las primeras versiones y sus características (hoja de ruta). La planificación de lanzamientos busca equilibrar alcance, fecha y presupuesto para entregas de producto incrementales. Para ello se divide la pila del producto en bloques, para establecer que PBI se llevarán a cabo en las próximas versiones, y se utilizan las estimaciones de las PBI y las velocidades de los equipos para dar fechas aproximadas de las versiones. La planificación de lanzamientos se hace una vez tras la de producto, pero luego puede volverse a realizar en cualquier momento considerando los cambios que se hayan producido en la pila.

Para la planificación de lanzamientos podemos encontrarnos con que el alcance (características a implementar), los plazos y el presupuesto están prefijados (proyecto "llave en mano") o con que alguna de estas restricciones es flexible. Scrum es perfecto

cuando tenemos una fecha de entrega prefijada, pero tenemos flexibilidad con el alcance, y adecuado cuando tenemos un alcance prefijado pero flexibilidad con la fecha de entrega (pero hay que ser conscientes de que si extendemos la fecha de entrega, estamos extendiendo también el presupuesto). A efectos prácticos debemos considerar que Scrum es incompatible con los proyectos en los que todo está prefijado, aunque siempre podemos aplicar algunas de las técnicas de Scrum en ellos. De hecho, en estos casos en los que no hay flexibilidad ni en alcance, ni en fecha, ni en presupuesto, los equipos de desarrollo acaban sacando flexibilidad de donde menos deberían hacerlo que es en la calidad del producto (el terrible concepto de "calidad flexible").

**Contenidos**

- Principios de la planificación.
- Planificación multinivel.
- Planificación de producto.
- Planificación de lanzamientos: restricciones, alcance prefijado, fecha prefijada y cálculo de costes.

**Bibliografía**

Basado principalmente en los capítulos 14, 15, 17 y 18 de Rubin [2012].

### 7.2.9. Scrum 8 - Los sprints en detalle

Los sprints en Scrum comienzan con su planificación. Para ello, el equipo Scrum al completo selecciona las entradas de la pila que se completarán durante el sprint (el dueño de producto decide las que más interesan, pero es el equipo de desarrollo el que determina de esas las que tienen capacidad suficiente (horas disponibles) para terminar). Cada entrada se divide en tareas y se estima cuántas horas costará cada una. Las entradas de la pila a implementar y las tareas en las que se han dividido forman la denominada pila del sprint, que a lo largo del sprint típicamente se transforma en el tablero del sprint, que es un mecanismo de comunicación para reflejar el estado de avance en las tareas (al menos "por hacer", "haciendo", "hechas", y quién está con cada cosa).

Una vez planificado, el sprint se ejecuta. Los miembros del equipo de desarrollo eligen tareas y las llevan a cabo. El ideal es el equipo auto-organizado, en el que no es necesario que alguien asigne tareas, y multidisciplinar, el equipo tiene todas las habilidades requeridas para completar las entradas de la pila. Una vez al día, normalmente a primera hora, se realiza una breve reunión (el Scrum diario) que permite que todo el mundo sepa en lo todos están trabajando y en la que se lleva a cabo cierta coordinación general (la coordinación más específica se resuelve después). El dueño de producto está continuamente disponible para ser consultado, y el Scrum master hace lo posible para que el equipo de desarrollo trabaje sin obstáculos.

Al terminar el sprint se hace una reunión de revisión, en la que se presentan los avances terminados a los clientes, usuarios etc. con el objetivo de recabar sus impresiones, ideas etc. Después se lleva a cabo la retrospectiva, que es una reunión del equipo Scrum, sin clientes, donde se examina el trabajo realizado en el último sprint desde el punto de vista del proceso, se determinan problemas y se proponen mejoras para implementar durante el siguiente sprint.

**Contenidos**

- Planificación de un sprint.
- Ejecución de un sprint.
- Revisión de un sprint.
- Retrospectiva de un sprint.

Como complemento a este tema se realiza un ejercicio de análisis de la causa raíz utilizando diagramas de causa-efecto, que es una técnica para analizar un problema hasta distinguir sus causas raíz, sus síntomas y sus consecuencias y poder abordar las primeras. Esta técnica no es específica de Scrum, pero es muy compatible con metodologías ágiles y su carácter general permite aplicarla a muchos problemas que pueden surgir durante un sprint de Scrum. Para demostrar el carácter general de la técnica, el ejercicio propone a los estudiantes que analicen el problema de la corrupción en España frente al objetivo[1] de tener unas administraciones públicas que gasten el dinero de manera eficiente y que los ciudadanos paguen impuestos justos.

**Bibliografía**

Basado principalmente en los capítulos 19, 20, 21 y 22 de Rubin [2012] y el capítulo 20 de Kniberg [2011].

### 7.2.10. Scrum 9 - Los roles en detalle

Un equipo Scrum está formado por el dueño del producto, el Scrum master y el equipo de desarrollo, y normalmente estará formado por entre 5 y 9 personas. Se buscan equipos de vida larga (equipo, no grupo), que trabajen a un ritmo sostenible (el que se puede mantener indefinidamente) y, cuando es posible, focalizados en un único proyecto cada vez.

La persona dueña del producto representa los intereses de clientes, usuarios o inversores, se comunica con ellos cuando es necesario, y es la autoridad con respecto a esto de cara al equipo Scrum. Tiene que realizar cierta gestión económica, participar en el *grooming* de la pila, definir criterios de aceptación y validarlos y estar disponible siempre que el equipo necesita resolver alguna duda. Para un desarollo para terceros, debe ser un representante de ellos, mientras que en un desarrollo interno puede ser alguien de la organización que pueda llevar a cabo las tareas anteriores.

[1] Algo solo es un problema si dificulta o impide alcanzar algún objetivo.

Scrum master es el rol de la persona que asesora al equipo sobre el proceso Scrum, actúa como "lider sirviente" (¿qué puedo hacer para ayudaros a ser más efectivos?), protege al equipo de injerencias externas y trata de eliminar los obstáculos que impiden al equipo rendir al máximo. Este es un papel que normalmente se compagina con otras tareas.

El equipo de desarrollo son los encargados de llevar a cabo el producto o servicio. Debe ser multidisciplinar, y tener todas las habilidades necesarias para completar el trabajo[2]. Los equipos especializados no encajan bien con Scrum, puesto que en un momento dado suele haber alguna entrada de la pila prioritaria que solo puede hacer un equipo especializado y ya está ocupado con otras, con lo que uno de los puntales de Scrum, que el dueño de producto puede decidir casi totalmente el orden en el que se implementan las características del mismo, se tambalea. El equipo debe auto-organizarse (ser capaz de repartir tareas y tomar decisiones sin que se les impongan un líder), tener "actitud de mosqueteros" (ser colaborativos, estar dispuestos a trabajar o ayudar en lo que sea más importante o urgente cuando toque) y una buena capacidad de comunicación.

Para productos grandes, se hará necesario coordinar a varios equipos Scrum puesto que cada uno suele tener como máximo 9 personas. Un producto grande se suele dividir en componentes. Luego se puede asignar uno o varios equipos Scrum a cada componente (equipos de componentes), o se pueden tener equipos intercambiables que vayan tomando entradas de la pila indistintamente y tocando los componentes necesarios para completarlas (equipos de características). Aunque los de características son más flexibles, los de componentes suelen ser preferidos por las organizaciones y pueden funcionar bien, sobre todo si hay un único producto grande en desarrollo.

Scrum no requiere más roles, pero tampoco los prohíbe. Muchas organizaciones van a exigir que cada proyecto tenga al menos alguien en el papel de "director de proyecto". Si una empresa quiere trabajar con Scrum, tendrá que adaptarse al tipo de directores más compatibles con esta metodología: apoyar más que dirigir, poner objetivos de medio/largo plazo (pero dejar que ellos decidan los de corto), coordinar con otros equipos Scrum, grupos u organizaciones, eliminar obstáculos etc.

**Contenidos**

- El dueño del producto.
- El ScrumMaster.
- El equipo de desarrollo.
- Estructuración de equipos Scrum.
- Managers.
- Directores de proyecto.

[2]El equipo como un todo debe tener todas las habilidades, no se espera que todos los integrantes las tengan.

**Bibliografía**

Basado principalmente en los capítulos 9, 10, 11, 12 y 13 de Rubin [2012] y en alguna idea del capítulo 12 de Pham y Pham [2011].

### 7.2.11. Scrum 10 - Arquitectura y deuda técnica

Uno de los mitos persistentes sobre Scrum para proyectos software es que no se hace diseño arquitectural a alto nivel, o que nadie tiene una visión global del proyecto. De hecho es habitual hacerlo, y se puede hacer de manera que no se colisiones con los principios ágiles.

Se puede diseñar y documentar la arquitectura del software que vayamos a construir siempre que recordemos que puede mejorarse, debe concretarse en el código y que se debe mantener la capacidad de responder a los cambios. Lo que es incompatible con las metodologías ágiles es la arquitectura prescriptiva, que tiene los mismos problemas que la planificación prescriptiva: en un producto complejo y con incertidumbres, el que mejor encaja con una metodología ágil, no tienes suficiente información al principio como para tomar todas las decisiones técnicas sobre el proyecto, especialmente las más críticas y luego dejar que otros las lleven a cabo sin casi intervenir y además dificultando que puedan hacer los cambios necesarios. Se pueden tomar ciertas decisiones al principio, pero conforme avanza el proyecto se deben ir completando, concretando o corrigiendo, y para ello es necesario que los diseñadores más experimentados estén ahí en el día a día del proyecto. El modelo ideal en este tipo de proyectos es el de "arquitecto y maestro programador", no el de "arquitecto de torre de marfil".

Un formato adecuado es el de los talleres de diseño, donde todos los equipos participan y donde se discute y se toman decisiones sobre la arquitectura del sistema a implementar (antes de empezar un proyecto nuevo), o se documenta el sistema en su estado actual (para reflejar los cambios y correcciones que se han ido llevando a cabo). Lo habitual es dedicar uno o dos días, trabajar sobre pizarras y dividir a la gente en pequeños grupos facilitando en lo posible el intercambio de ideas. Lo típico es que la voz de los diseñadores más experimentados tenga más peso en estos talleres y que los más nuevos se dediquen sobre todo a aprender, pero se fomenta que todo el mundo participe y se busca que todo el mundo adquiera al menos cierta visión sobre el sistema en su conjunto.

Uno de los problemas de descuidar los aspectos de diseño en cualquier proyecto de software (tomar atajos para entregar algo a tiempo, reducir tests, no automatizar tareas etc.) es la acumulación de problemas que tarde o temprano acaban por causar más retrasos que los "ahorros" de tiempo iniciales. Esto se puede ver como la adquisición de una deuda técnica que, tarde o temprano, hay que pagar y que además acumula intereses: por ejemplo, un componente mal diseñado y difícil de comprender, causa un retraso cada vez que se integra con otro porque el desarrollador que lo hace tiene que perder más tiempo del necesario entendiendo cómo funciona. El objetivo de denominar a esto deuda es poder explicar a la gerencia de la empresa en términos a los que estén habituados los problemas que va a causar no dedicar el tiempo suficiente al diseño del software porque, al menos al principio, un mal diseño no tiene por qué reflejarse en un mal producto y los usuarios no lo van a percibir, así que la gerencia puede tener

problemas para comprender que los problemas llegarán tarde o temprano.

**Contenidos**

- Arquitectura y diseño de software.
- Una perspectiva ágil de la arquitectura de software.
- Talleres de diseño ágil.
- Modelizado *just-in-time* y cercano al código.
- Talleres de documentación de la arquitectura del sistema.
- La deuda técnica.

**Bibliografía**

Basado principalmente en el capítulo 8 de Larman y Vodde [2010] y el capítulo 8 de Rubin [2012].

### 7.2.12. Scrum 11 - Tests en proyectos ágiles

Dada la importancia de los tests en las metodologías ágiles, es importante considerarlos en un tema sobre Scrum para proyectos de software. Desde esta perspectiva, hay que empezar cuestionando (no necesariamente descartando totalmente) algunas de las asunciones "tradicionales", como que el testeo debe ser independiente del desarrollo, que no puede empezar hasta terminar la implementación, que debe haber un responsable de los tests, debe hacerse al final, debe estar planificado, debe haber una estrategia y un plan maestro de tests, que cubrir el 100 % de los casos es demasiado caro o que el testeo debe ser realizado por especialistas en eso.

El desarrollo dirigido por tests (TDD, *test driven development*) es una técnica de diseño de software donde los tests se construyen antes del sistema y se usan por tanto como especificación del mismo antes de ser usados como pruebas automáticas. El desarrollo dirigido por tests de aceptación (A-TDD, *acceptance-test driven development*) es una técnica de captura y análisis de requisitos donde los tests se usan para especificar el comportamiento del sistema y verificar después que es correcto. Ambas técnicas, TDD y A-TDD, que se ven en otras asignaturas, son puntales de las metodologías ágiles y tienen como característica que la captura de requisitos y la especificación del diseño del software se basan en la creación de código que luego puede usarse durante el desarrollo para validar los requisitos y verificar las especificaciones de forma automática.

Dentro de Scrum, hay que considerar como hacer el seguimiento de los defectos y cuando se hacen los tests. Tener pilas solo para defectos se hace algunas veces, pero no es lo más recomendable. Es más flexible tener los defectos en la pila del producto, y que así se pueda priorizar corregir un defecto o crear una característica nueva. En los sprints, los tests se consideran cuando se hace el *grooming* de la pila, por ejemplo clarificando los requisitos que aparecen en ella mediante tests de aceptación, en la planificación del sprint, donde hay que pensar las tareas necesarias para corregir los

defectos que se van a abordar, en la ejecución del sprint, donde hay que ejecutar los tests y diseñar nuevos y en la revisión donde se pueden usar los tests de aceptación como forma de mostrar que la funcionalidad es la esperada. Los errores que se encuentran a mitad de un sprint o se arreglan inmediatamente, o se tratan de abordar dentro del mismo sprint. Los defectos que se encuentran fuera de un sprint (p.ej. en producción), se reflejan en el sistema de seguimiento de errores y/o en la pila del producto para abordarlos cuando se decida.

**Contenidos**

- Tipos de tests
- Desarrollo dirigido por tests
- Desarrollo dirigido por tests de aceptación
- Estrategias y situaciones que surgen en los proyectos
- Los tests en Scrum

**Bibliografía**

Basado principalmente en el capítulo 3 de Larman y Vodde [2010] y el capítulo 4 de Humble y Farley [2010].

### 7.2.13. Scrum 12 - Gestión de configuraciones en proyectos ágiles: introducción a la entrega continua

Scrum es un marco de gestión de proyectos ágil, con pocas actividades requeridas pero que difícilmente puede funcionar bien si no se acompaña de algunas prácticas técnicas, especialmente de aquellas que permiten automatizar procesos. La integración y el despliegue o lanzamiento de software han sido siempre actividades problemáticas, y era habitual que, en proyectos medianamente complejos, pasaran semanas o meses entre el momento en que en teoría se había completado el desarrollo y el momento en que los usuarios finales podían usarlo. Siguiendo la receta ágil de "si duele, hazlo más a menudo", se han desarrollado en los últimos años herramientas y técnicas para automatizar estos procesos casi por completo, de manera que se puedan hacer de manera continua para que cuando se quiere hacer un despliegue hacia los usuarios finales, este proceso sea esencialmente automático y muy rápido (minutos en lugar de meses) porque se ha hecho casi igual decenas o cientos de veces durante el desarrollo.

La técnica de la entrega continua (*continuous delivery*), parte de la gestión de configuraciones de un proyecto de software, integrar en un pipeline automático las distintas fases de entrega de un software. A partir de la subida a un sistema de control de versiones de un conjunto de cambios en cualquier parte del código fuente, se dispara un procedimiento automático que ejecuta el pipeline y culmina en el despliegue en producción (o en un sistema de pruebas equivalente) si tiene éxito: compilación, enlazado con bibliotecas, análisis estático del código, tests unitarios, configuración del

entorno de pruebas, despliegue en el entorno de pruebas, tests de integración, tests de aceptación automáticos, tests de capacidad automáticos, tests manuales, configuración del entorno de despliegue, y despliegue. Si hay problemas en cualquier fase del pipeline, se informa a los desarrolladores para que puedan tomar las medidas necesarias.

Para que la entrega continua se pueda implementar, es necesario trabajar regularmente con un sistema de control de versiones y realizar integración continua, que consiste en la compilación y prueba automática del sistema software completo, todo el tiempo, cada vez que se sube a control de versiones un conjunto de cambios en el código fuente de cualquiera de sus componentes.

**Contenidos**

- Problemas comunes en el despliegue y lanzamiento del software.
- Solución: la entrega continua.
- El pipeline de despliegue.
- Beneficios de la entrega continua.
- Fundamentos para la entrega continua: control de versiones y dependencias.
- Fundamentos para la entrega continua: la integración continua.

**Bibliografía**

Basado en los capítulos 1, 2 y 3 de Humble y Farley [2010].

### 7.2.14. Técnicas de gestión 1 - Gestión de riesgos

Un riesgo para un proyecto es un evento o condición inciertos, que si ocurre tendrá un impacto (positivo/oportunidad, o negativo/amenaza) en alguno de los objetivos de este proyecto (en alcance, plazos, coste o calidad). Todos los proyectos tienen incertidumbres, y los riesgos se originan en estas. Gestionar los riesgos consiste en identificar riesgos potenciales y estimar su probabilidad e impacto. Luego se priorizan por su importancia y se diseñan medidas para el caso de que ocurran.

Para identificar los riesgos potenciales se puede partir de clasificaciones y listas existentes, de la información histórica que la organización tenga sobre proyectos similares o se puede contar con expertos. Una vez identificados, el análisis establece la exposición a cada riesgo a partir de las probabilidades y los impactos estimados, permitiendo clasificarlos y seleccionar los más prioritarios que hay que considerar (p.ej. aquellos cuya exposición sea mayor).

Una vez analizados, hay que planificar las respuestas que se tomarán si los riesgos finalmente se manifiestan. El objetivo es mitigar las amenazas y aprovechar las oportunidades. Cuando es posible, hay que evitar las amenazas (p.ej., no llevar a cabo acciones que puedan conducir a ellos sino otras alternativas). Si no, se pueden transferir a terceros (p.ej. contratar un seguro). Si no se pueden evitar ni transferir, habrá que mitigarlas reduciendo su probabilidad y/o su impacto (p.ej. hacer más pruebas o

valorar calidad antes que precio en la adquisición de componentes). En el caso de las oportunidades, se tratará de explotarlas (p.ej. si aparece la oportunidad de incluir en el proyecto a alguien muy experimentado, tratar de reducir plazos) o compartirlas con un tercero que esté en mejores condiciones de aprovecharla (p.ej. se puede pasar un contacto de alto nivel inesperado al departamento de marketing).

Los riesgos, igual que todo en el proyecto, hay que monitorizarlos para descubrirlos lo antes posible y poder ejecutar la respuestas planificadas, lo que puede requerir reportarlos al nivel de decisión adecuado. Una herramienta básica son las auditorías de riesgos, que típicamente se realizan por personal externo al proyecto y aseguran que la gestión de riesgos es la adecuada. Las revisiones de riesgos son tareas internas periódicas que buscan descubrir nuevos riesgos potenciales o identificar aquellos que han surgido.

Todas las metodologías actuales consideran la gestión de riesgos como una parte integrada de la gestión de proyectos y proponen actividades similares, las que se ven en el tema, aunque cada organización tendrá que adaptarlas a sus propias prácticas, asignar las tareas al personal y reflejarlas en su documentación.

**Contenidos**

- Identificación de riesgos.
- Análisis de riesgos.
- Planificación de respuestas a los riesgos.
- Monitorización y control de riesgos.
- Planificación y procesos de gestión de riesgos.

Este tema se completa con un ejercicio en clase en el que los estudiantes tienen que, de manera individual, identificar y analizar los riesgos del proyecto de la asignatura, pero asumiendo que el tamaño del mismo es unas diez veces mayor (para que los riesgos tengan un mayor impacto potencial). Para ello se les da una lista de riesgos típicos, y los estudiantes tienen que:

1. Decidir si ese riesgo merece ser investigado o no (p.ej. porque no se aplica a su proyecto).
2. Estimar la probabilidad de que los riesgos a investigar ocurran.
3. Estimar el impacto (la pérdida o ganancia, que puede ser en tiempo, dinero, calidad o alcance) si los riesgos a investigar realmente ocurren (para simplificar y facilitar la comparación se traduce en una escala del 1 al 5).
4. Calcular la exposición a cada riesgo (probabilidad x impacto).

Luego cada equipo integra los análisis de riesgos que han hecho sus integrantes, los refleja en una matriz de riesgos (impacto en las filas y probabilidad en las columnas) y diseña un plan de riesgos que considere los más grandes (mucho impacto y alguna probabilidad, o algún impacto y mucha probabilidad).

**Bibliografía**

Basado principalmente en el capítulo 11 de Project Management Institute [2013] y cosas de Pandian [2007].

### 7.2.15. Técnicas de gestión 2 - Gestión del tiempo

Entregar un proyecto fuera de plazo tiene un importante impacto negativo en el coste y en la satisfacción del cliente, por ello es uno de los aspectos más importantes en la gestión de proyectos, y uno de los que más conflictos causa. A partir de la planificación de las tareas que hay que realizar en un proyecto se puede estimar el esfuerzo necesario para realizarlas. Con esa información ya se puede establecer un calendario del proyecto, que consiste en indicar quién hace qué y cuándo. Ese calendario tiene que equilibrar el tiempo, el coste y los riesgos. Con más coste se puede reducir el tiempo, pero es posible que el coste sea un factor limitante, y asumiendo más riesgos también, pero es posible que esto tenga consecuencias graves.

Para visualizar calendarios de proyectos la herramienta más común es el diagrama de Gantt, que muestra la secuencia temporal de la tareas, sus duraciones y sus dependencias. Para analizar calendarios se puede usar la técnica del camino crítico (CPM, *critical path method*), que permite determinar la secuencia de tareas de la que depende la fecha de finalización del proyecto (si se retrasa cualquier tarea de este camino crítico, el proyecto se retrasa). La técnica de revisión y evaluación de programas (PERT, *program evaluation and review technique*) permite incorporar incertidumbre sobre la duración de las tareas y estimar el tiempo de un proyecto (con un margen de error) a partir de ahí.

Dentro de la monitorización y seguimiento de un proyecto una de las tareas básicas es el control del calendario. Es necesario recopilar el esfuerzo invertido, asegurándonos de que no se está yendo por encima de las previsiones, chequear que las fechas y plazos comprometidos se cumplen, y reaccionar si no, y alterar el calendario cuando es necesario (ninguna metodología actual considera el primer calendario como algo inmutable).

**Contenidos**

- Qué es la gestión del tiempo.
- Diseño de planes y calendarios: diagramas de Gantt, CPM, PERT.
- Control del calendario: seguimiento de esfuerzos y tiempos.

Este tema se completa con un ejercicio en clase en el que los estudiantes parten de un conjunto de tareas, sus duraciones estimadas y sus dependencias, y tienen que construir el grafo de las mismas y determinar el camino crítico (técnica del *Critical Path Method*), y otro en el que la duración de esas tareas se da con un margen de error (optimista, probable, pesimista) y se les pide que creen el grafo PERT (*Program Evaluation and Review Technique*), identificando de nuevo el camino crítico pero esta vez con la desviación estándar.

**Bibliografía**

Basado principalmente en el capítulo 6 de Project Management Institute [2013] y en Brooks [1995].

### 7.2.16. Técnicas de gestión 3 - Gestión del coste

El coste de un proyecto es el montante económico que cuesta llevarlo a cabo, se mide en unidades monetarias y está formado por el valor de los recursos sacrificados para realizarlo. El presupuesto es una estimación de lo que se quiere obtener como contrapartida por llevar un proyecto a cabo. El precio es el retorno económico que se obtiene por un proyecto.

La gestión de costes del proyecto consiste en asegurarse de que el proyecto se completa dentro del presupuesto que se ha aprobado. Esto es responsabilidad de la directora del proyecto, no de la gestión financiera de la organización, y por tanto es parte de la gestión de proyectos. Algunos términos que aparecen habitualmente y que los directores de proyectos deben conocer y tener en cuenta son el beneficio, los ingresos menos los gastos más otros activos conseguidos (tecnología, conocimientos, nuevos clientes), el margen de beneficio, la razón entre beneficios e ingresos y el coste del ciclo de vida, que es el coste del desarrollo más el mantenimiento y el soporte a lo largo de la vida del producto o servicio. Algunos costes son directos, aquellos directamente relacionados con la producción de los productos y servicios del proyecto (salarios, hardware, software, viajes), y otros son indirectos (alquiler de local y otras infraestructuras y servicios de apoyo).

Las estimaciones de costes se refinan conforme se avanza en un proyecto. Para decidir si un proyecto se lleva a delante hace falta hacer una estimación inicial de grano grueso, para que la organización le pueda asignar una partida económica hay que hacer un presupuesto más ajustado (antes de empezar el proyecto o durante la redacción de la propuesta del mismo) y durante el proyecto hay que hacer una estimación definitiva. Las estimaciones pueden basarse en el juicio de expertos o en métodos paramétricos a partir de datos históricos.

El control de costes consiste primero en monitorizar los esfuerzos de desarrollo del proyecto (gastos de personal), los costes directos (gastos en viajes, licencias de software, hardware etc.) y los indirectos (infraestructuras, servicios de apoyo como la contabilidad etc.). Y después, en tener en cuenta, dentro del control de cambios del proyecto, el impacto económico que estos cambios van a tener en el proyecto y tomar las medidas necesarias (rechazar el cambio, solicitar permiso para un gasto inesperado que se sale del presupuesto etc.).

**Contenidos**

- Coste, presupuesto y precio.
- Principios básicos de la gestión de costes.
- Tipos de costes y beneficios.
- Estimación de costes y determinación del presupuesto.

- Control de costes.

El tema termina con un ejercicio en el que los estudiantes establecen el presupuesto del proyecto de la asignatura (asignar una estimación de coste a cada elemento de trabajo individual, que pueden ser resultados, o las actividades para alcanzar esos resultados), y realizan el control del coste (cuánto dinero ha costado ya) en el momento de la realización del ejercicio a partir de los esfuerzos realizados hasta ese momento y de los resultados alcanzados.

**Bibliografía**

Basado principalmente en el capítulo 7 de Project Management Institute [2013].

### 7.2.17. Técnicas de gestión 4 - Gestión de la financiación

Si diriges un proyecto, es importante conocer cómo ese proyecto está pagando tu nómina, o parte de ella. De que el cliente acepte y firme las entregas, que es responsabilidad de la directora del proyecto, suele depender que la organización pueda emitir, y luego cobrar, las facturas que al final pagarán las nóminas.

En el contexto de los proyectos de software, la contratación es el proceso por el que se acuerda con un cliente qué se va hacer, cuándo se va a entregar, cómo se va a facturar y cuándo se cobrará. Facturar consiste en emitir un documento legal (la factura) que es una solicitud de pago, normalmente por un producto o servicio entregado (o por partes de estos). El pago es el proceso por el que el cliente transfiere dinero al proveedor (normalmente tras la emisión de una factura por este). El cobro es el proceso por el que el proveedor recibe dinero tras realizar una actividad para un cliente (y normalmente haber emitido una factura por ella). Es importante ser consciente de que estas actividades van en orden y separadas en el tiempo: no se cobra hasta que pagan, no pagan hasta que se factura, y no se factura si no hay un contrato.

El problema de la financiación, que tienen que resolver todas las organizaciones, es cubrir los gastos entre el momento en que se firman los contratos y el momento en que se puede cobrar por los resultados de los mismos. Lo ideal es que el cliente pague por adelantado, quizás una parte, pero esto puede no ser suficiente. Las necesidades de financiación dependen del tipo y del número de proyectos que tiene la organización.

Cuando se trabaja para la administración pública, normalmente las facturas se emiten contra certificaciones a final del proyecto, o contra resultados parciales intermedios. Hay un límite de tiempo legal para que hagan los pagos, pero en la práctica no siempre se cumple. Se puede contar con que la administración pública es un cliente que siempre paga, pero como los pagos se pueden retrasar bastante hace falta contar con buena financiación.

Si el cliente es privado, típicamente se factura por partes: una parte a la firma del contrato, otra por entregas intermedias y una al final del proyecto (40 %, 40 % y 20 % es un buen punto de partida). El pago normalmente se vincula a la fecha de emisión de las facturas (7, 15, 30 o hasta 60 días después es habitual). A diferencia de las administraciones públicas, la posibilidad de impagos aquí es muy real, y en el peor

de los casos hay que denunciarlos en un juzgado. La forma de pago implica que hay menos necesidades de financiación.

Cuando no se trabaja para un cliente concreto sino que se desarrolla un producto para su venta directa, para licenciarlo a terceros, o para usarlo en futuros proyectos (quizás con algo de personalización), hay que conseguir dinero para cubrir los costes hasta que empieza a comercializarse (o a usarse en proyectos que puedan ir amortizándolo). La necesidad de financiación para esto es, por tanto, grande.

Cuando es necesario obtener financiación, se puede recurrir a préstamos o créditos bancarios, capital-riesgo, inversores, socios etc. Para conseguir este dinero es necesario demostrar que el proyecto hace viable devolverlo con intereses (viable técnicamente, la parte técnica de los proyectos de software se ve en muchas otras asignaturas, pero también comercial y económicamente). La viabilidad comercial implica un producto bien definido con un modelo de negocio factible (una visión realista sobre cómo y de quién se obtendrán los ingresos por ese producto, incluyendo un análisis de las ventajas frente a la competencia y las acciones de marketing previstas). La viabilidad económica suele estar más ligada a la organización que a un proyecto concreto, salvo que la empresa desarrolle un único producto. Para demostrarla, lo habitual es tener que presentar información económica de tu empresa en los últimos años (facturación, cuenta de resultados, activos, préstamos y deudas), y puede que información económica de los avalistas si son necesarios.

### Contenidos

- ¿Por qué conocer los fundamentos de la financiación es necesario para dirigir proyectos?
- Contratos y facturas, cobros y pagos.
- Financiación de proyectos con administración pública y con otras organizaciones.
- La viabilidad de un proyecto.

La unidad se completa con un ejercicio en el que los estudiantes definen brevemente un modelo de negocio y un plan de financiación para el proyecto de la asignatura. Para ello tienen que determinar potenciales clientes, que utilidad van a proporcionar a estos clientes (p.ej. diferencias con la competencia), cuánto van a cobrar, y qué financiación estiman que necesitarán para tener un producto mínimo viable, un producto completo y para tener un régimen estable de explotación.

### Bibliografía

Aunque la metodología del libro considera la financiación como un aspecto separado a la gestión de proyectos, algunos aspectos de la misma se tratan en el capítulo 7 de Project Management Institute [2013].

## 7.3. El proyecto en equipo

El proyecto consiste en la realización de un pequeño sistema informático en equipos de entre 4 y 6 personas (dependiendo del número de estudiantes matriculados puede hacer falta flexibilizar algo estos números), cuyo estado final debe ser el de "potencialmente entregable" (*potentially shippable*), que significa que debe estar lo suficientemente completo, probado y documentado como para poder ser entregado a un cliente o puesto a disposición de un grupo de usuarios finales. Estos equipos son de tipo colaborativo, descritos en la Sección 4.3, y desde el punto de vista docente el proyecto se lleva a cabo siguiendo la metodología de aprendizaje basado en proyectos (ver Sección 4.4.5).

Cada estudiante prepara una propuesta de un proyecto que quiera realizar, las presentan y defienden en clase, y después se votan. Las más votadas (tantas como equipos haya) serán las que se hagan durante el curso. Aunque este es el mecanismo por defecto, si surgen oportunidades para que los trabajos que se realicen tengan alguna aplicación más real se aprovechan (por ejemplo el curso 2014/2015 los estudiantes han realizado un pequeño proyecto para el Centro Universitario de Lenguas Modernas de la Universidad de Zaragoza). La tecnología y el dominio de problema son casi ortogonales a los contenidos de la asignatura, así que los estudiantes son libres de elegir lo que les interese, algo que busca incrementar su motivación y exige que sean autónomos a la hora de resolver problemas (por ejemplo, el profesor normalmente no puede ayudarles con problemas técnicos puesto que son los estudiantes los que han decidido la tecnología a emplear). Esto, junto con el sistema de evaluación del proyecto, busca que el proyecto resulte motivante para los estudiantes (es decir, tratan de alinearse con las propuestas de la Tabla 4.1).

El proyecto se lleva a cabo siguiendo la metodología ágil de gestión de proyectos Scrum. Más que una metodología, Scrum es un marco que requiere ser adaptado a las circunstancias de los que lo practican. Este requisito de adaptabilidad exige que los estudiantes comprendan los principios sobre los que se basa la gestión del proyecto en lugar de seguir ciegamente una serie de pasos predefinidos, algo que sin duda es importante en la formación de un ingeniero. Por comparar, en Scrum solo hay 9 prácticas requeridas (el número puede variar un poco dependiendo de como se haga la cuenta), mientras que, por ejemplo, en el *Rational Unified Process* se pueden contar más de 120 (cierto es que no todas son obligatorias, pero también es cierto que ante la duda, lo más fácil es aplicarlas "por si acaso"). Esto da una idea de la cantidad de decisiones que los practicantes de Scrum, en este caso los estudiantes de la asignatura, tienen que tomar.

### 7.3.1. Organización temporal del proyecto

El proyecto se realiza en dos sprints (el término de Scrum para las iteraciones). Dado el tiempo disponible por los alumnos y la importancia de que haya varios sprints y así se puedan llevar a cabo adecuadamente todas las fases y actividades de Scrum, el dos es un número adecuado. En una organización que aplique Scrum, lo típico es tener sprints de entre 1 semana y 1 mes de duración, con los equipos más experimentados normalmente prefiriendo sprints más cortos. Si se trabaja a tiempo completo, un sprint

de una semana son unas 40 horas por persona, que es casi la mitad del tiempo que los estudiantes tienen para el proyecto, así que en realidad realizar dos sprints en la asignatura está más cerca de ser equivalente a sprints de una semana en un contexto profesional, que a sprints más largos que serían más adecuados para equipos poco experimentados.

El hecho de que haya dos sprints, y el énfasis que Scrum pone en que al final de un sprint debe haber un "incremento de producto potencialmente entregable", criterio que se puede relajar durante los primeros sprints, conduce de manera natural a dos entregas durante el curso, una casi a mitad y otra al final.

**Hitos del proyecto en equipo**

Algunas de las actividades, reflejadas en la tabla 7.3 son supervisadas (con profesor) y se usan para orientarles en las tareas a realizar, para adelantar algunas cosas que necesitan saber pero que todavía no se han visto en teoría y para resolver dudas, mientras que otras las realizan los estudiantes por su cuenta, y luego informan de los resultados.

Los equipos tienen que realizar por su cuenta la ejecución de cada sprint (las actividades que conducen a la creación del software). Scrum requiere que todos los días se lleve a cabo una reunión rápida (*daily scrum*) en la que los miembros del equipo digan lo que hicieron el día anterior, problemas que han surgido y lo que planean hacer ese día. Esto es difícil para el caso de los equipos de estudiantes, que tienen que compaginar el trabajo con otras asignaturas, pero se alienta que lo hagan al menos una vez por semana (y se fuerza en las reuniones de seguimiento supervisadas).

Las reuniones de seguimiento supervisadas se atienen a las pautas dadas para el trabajo en grupos pequeños con formatos de tutoría o de discusión libre, ver sección 4.4.4, según la reunión sea más para comentar lo que han estado haciendo, o para darles la oportunidad de realizar por ellos mismos algunas actividades con el asesoramiento del profesor.

## 7.4. Metodologías de enseñanza-aprendizaje

Las actividades de enseñanza-aprendizaje de esta asignatura se basan fundamentalmente en:

1. Lecciones magistrales (sección 4.4.1): se usan para transmitir de forma eficiente los contenidos teórico-prácticos de la asignatura, así como para provocar un diálogo con los estudiantes en base a preguntas convergentes y, especialmente, divergentes (sección 4.4.2).

2. Resolución de problemas y casos: se trata de aplicar los conceptos teóricos a situaciones pequeñas, que puedan resolverse drante una sesión de problemas por parte de los estudiantes. Algunas veces se usa el formato de simulación o juego (sección 4.4.4).

3. Desarrollo de un trabajo en equipo siguiendo la metodología de aprendizaje basado en proyectos (sección 4.4.5). Este trabajo, descrito en detalle en la sección

| | | | |
|---|---|---|---|
| **Semana 1** | **Plan de producto** | **Supervisado** | |
| Con un horizonte temporal de un año y asumiendo que el primer lanzamiento público de su producto sería al finalizar la asignatura. Lo desarrollan por su cuenta, pero se discute después con el profesor. | | | |
| **Semana 1** | **Fase previa** | **Sin supervisión** | |
| El equipo toma algunas decisiones sobre su organización (por ejemplo si van a tener director, si se van a votar las decisiones etc.), hace un diseño arquitectural inicial del sistema construir, ponen en marcha las actividades de gestión de configuraciones y de seguimiento de esfuerzos, hacen una primera selección de la tecnología que quieren emplear y un pequeño análisis de riesgos. su cuenta, pero se discute después con el profesor. | | | |
| **Semana 2** | **Lanzamiento del proyecto** | **Supervisado** | |
| El equipo se reparte los roles de Scrum (ScrumMaster, Dueño del Producto y Equipo de Desarrollo). Se crea un esbozo inicial (lo tendrán que completar ) de la definición de hecho. Se crea una pila del producto inicial (entradas como requisitos de alto nivel, con estimaciones de esfuerzo iniciales y una primera priorización). Planificar el primer lanzamiento. | | | |
| **Semana 2** | **Lanzamiento del primer sprint** | **Supervisado** | |
| Se hace grooming de la pila (pero se resalta que esa es una actividad que se debe hacer de manera oportunista) asegurando que las entradas más prioritarias están lo bastante detalladas y estimadas, y que son lo bastante pequeñas como para que varias de ellas se puedan abordar en un único sprint. Se realiza la planifcación del sprint: decidir el objetivo, establecer cuál es la capacidad del equipo para ese sprint, seleccionar entre las entradas de la pila más prioritarias cuáles se realizarán, dividirlas en tareas y asegurarse de que pueden realizarse todas dada la capacidad disponible). El resultado de esta reunión es la pila del sprint (entradas de la pila del producto que se abordarán, y su división inicial en tareas). | | | |
| **Semana 4** | **Seguimiento del trabajo** | **Supervisado** | |
| Esta reunión es una oportunidad para que los profesores puedan ver el estado del proyecto y crear la oportunidad de que los estudiantes puedan realizar algunas actividades de manera supervisada. Al menos se realizará un "daily" scrum (esta reunión rápida debería ser diaria en una organización típica, y más o menos semanal en el caso del proyecto de asignatura) y se revisará el tablero de tareas (la pila del sprint se transforma en el tablero de tareas del sprint conforme las tareas planificada van pasando por sus distintas fases). También se puede realizar algo de grooming de la pila, o dibujar algún diagrama (burnup o burndown) para ilustrar el uso de esta herramienta gráfica en el seguimiento del trabajo (aunque en la teoría todavía no se ha visto). | | | |
| **Semana 8** | **Revisión del primer sprint** | **Supervisada** | |
| En esta reunión de revisión de los avances en el producto se demuestran las entradas de la pila completadadas (hechas y aceptadas) durante el primer sprint. Normalmente se producirá una breve discusión sobre lo que se ha visto (sin solucionar problemas graves, simplemente sacar a la luz cosas que se podrían mejorar o cambiar). Se pueden tener presentes preguntas de este tipo: ¿A clientes o usuarios les gusta lo que ven? ¿Querrían cambios? ¿Hay algo en el entorno/mercado que nos lleve a cambiar? ¿Hemos descubierto algo importante que se nos ha pasado? ¿Estamos dedicando mucho esfuerzo a algo que no es tan importante? | | | |
| **Semana 8** | **Retrospectiva del primer sprint** | **Supervisada** | |
| Para esta reunión de revisión de proceso, lo habitual es decidir un foco. En esta primera retrospectiva típicamente será todos los aspectos del proceso Scrum. Para la reunión hay que recopilar datos objetivos: entradas de la pila del sprint que no llegarno a terminarse, diagramas de burndown o burnup, bugs detectados etc. Se pueden usar algunas técnicas para dinamizar la reunión si cuesta sacar a la luz problemas con el proceso, como la línea temporal de eventos. El objetivo siempre es identificar cosas que se pueden mejorar ¿qué funcionó mal? ¿qué se puede probar de otras formas? etc. y luego elegir las que son más urgentes (se puede votar). Luego hay que decidir el tiempo que se dedicará a solucionarlos (que se restará de la capacidad del equipo en el próximo sprint) y, esto es lo principal, decidir las acciones que se llevarán a cabo. Algunas serán acciones específicas ("implementar sistema X para automatizar Y"), otras no restarán capacidad al equipo ("hay que ser puntuales"). Si un problema no se entiende bien, se puede establer como acción algún tipo de exploración. Para que las acciones se hagan, lo mejor es considerarlas como tareas en la planificación del siguiente sprint. | | | |
| **Semana 10** | **Seguimiento del trabajo** | **Supervisado** | |
| Similar a la realizada durante la semana 4. | | | |
| **Semana 14** | **Revisión del segundo sprint** | **Supervisada** | |
| Coincide con la presentación final del trabajo realizado. Mismo formato que la realizada al final del primer sprint, pero presentando los resultados tras el segundo. | | | |
| **Semana 14** | **Retrospectiva del segundo sprint** | **Sin supervisión** | |
| Similar a la realizada al final del primer sprint, los resultados deben reflejarse en el documento técnico final. | | | |

Tabla 7.3: Hitos del proyecto

7.3, es apoyado por los profesores en sesiones de discusión libre y tutorías (seccion 4.4.4).

4. Cuando es posible, se trae a profesionales externos para dar seminarios y charlas de interés dentro de los objetivos de la asignatura.

## 7.5. Planificación temporal de la asignatura

El programa de la asignatura por semanas (asumiendo 3 horas de actividades en aula por semana) está detallado en la Tabla 7.4. Aunque en la tabla figura una distri-

bución por semanas de las actividades por claridad, algunas actividades ocupan algo más de una semana, y otras pueden moverse algo por festividades u otros hechos variables, como por ejemplo encajar los horarios de los profesionales externos que imparten algunos seminarios. Se incluyen solo los hitos críticos del proyecto, que están completos en la Tabla 7.3.

| Semana | Teoría y Problemas | Actividades del Proyecto |
|---|---|---|
| 1 | Presentación de asignatura y planteamiento del producto | |
| | | |
| 2 | Scrum 1 - Introducción | Lanzamiento del proyecto |
| | Scrum 2 - Principios ágiles | Lanzamiento del primer sprint |
| | Ejercicio: Passing pennies | |
| | | |
| 3 | Scrum 3 - Sprints | |
| | | |
| 4 | Scrum 4 - Requisitos e historias de usuario | |
| | Ejercicio: Story spines | |
| | | |
| 5 | Scrum 5 - La pila del producto | |
| | Scrum 6 - Estimación y velocidad | |
| | Ejercicio: Póquer de planificación | |
| | | |
| 6 | Scrum 7 - Planificación | |
| | | |
| 7 | Scrum 8 - Los sprints en detalle | |
| | Ejercicio: Análisis de la causa raíz | |
| | | |
| 8 | Scrum 9 - Los roles en detalle | Revisión del primer sprint (presentación en aula) |
| | | |
| 9 | Scrum 10 - Arquitectura y deuda técnica | |
| | Scrum 11 - Tests en proyectos ágiles | |
| | | |
| 10 | Scrum 12 - Gestión de configuraciones en proyectos ágiles: introducción a la entrega continua | |
| | | |
| 11 | Técnicas de gestión 1 - Gestión de riesgos | |
| | Ejercicio: Identificación y análisis de riesgos del proyecto | |
| | | |
| 12 | Técnicas de gestión 2 - Gestión del tiempo | |
| | Ejercicio: Aplicación de CPM y PERT | |
| | | |
| 13 | Técnicas de gestión 3 - Gestión del coste | |
| | Ejercicio: Presupuesto del proyecto | |
| | | |
| 14 | Técnicas de gestión 4 - Gestión de la financiación | Revisión del segundo sprint (presentación en aula) |
| | Ejercicio: Modelo de negocio y plan de financiación | |
| | | |
| | | |

Tabla 7.4: Calendario por semanas

Los proyectos propuestos serán entregados al finalizar el cuatrimestre.

La asignatura está diseñada para ser cursada con una dedicación de 150 horas por parte de los estudiantes (6 ECTS), de las que unas 45 horas son para actividades presenciales, 15 para estudio y evaluación, y 90 para el trabajo en grupo.

## 7.6. La evaluación

La evaluarán de la asignatura tendrá dos partes:

1. Realización y defensa del trabajo en grupo, que valdrá un 75 % de la nota final. Los criterios de evaluación del trabajo se detallan a continuación. La nota individual de cada estudiante parte de la del trabajo, pero se tendrá en consideración la evaluación entre pares dentro del grupo y los esfuerzos individuales dedicados al proyecto.

2. Un breve examen teórico escrito sobre los conceptos impartidos en la teoría que valdrá un 25 % de la nota final.

La evaluación del proyecto se realiza a partir de una lista de resultados esperados que se entrega a los estudiantes el primer día de clase. El trabajo se evalúa de manera formativa a mitad del cuatrimestre, primer sprint, de manera que se pueda dar a los estudiantes una realimentación útil y aprovechable (siguiendo las pautas vistas en la Sección 4.5) sobre el desempeño que tienen hasta este momento. Esta realimentación consiste en indicar para cada resultado de aprendizaje esperado el estado en que se encuentra cada trabajo, y hacer sugerencias sobre cómo se puede mejorar. A continuación se presenta la lista de resultados de aprendizaje esperados que se usan para la evaluación del proyecto:

**Lo mínimo (se quiere aprobar)**

- El código del proyecto se aloja en Github. Se trabaja de forma habitual contra git.
- Cobertura de tests automáticos de al menos el 10 % del código (tests unitarios y/o de integración).
- Al menos una de las historias de usuario o requisitos implementados en el sprint siendo evaluado tiene buenos criterios de aceptación.
- Definición de "hecho" clara (puede incluir algunos/todos los RNF), y se usa. Esta definición incluye que se pasen todos los tests automáticos.
- Documentación arquitectural adecuada al momento del proyecto, incluyendo al menos dos vistas, una de módulos o de componentes y conectores, y la otra de despliegue del sistema. La documentación arquitectural es un fiel reflejo del sistema (y viceversa).
- Cumplimento suficiente de requisitos/características.
- Se llevará un control de esfuerzos con horas dedicadas por persona a cada tarea. Se trabaja un número de horas en el entorno de lo requerido para la asignatura (90 horas dedicadas al trabajo en grupo por persona). Se entregará un resumen cada dos semanas.

**Se valorará positivamente (se quiere sacar notable) si además de lo anterior**

- La cobertura de tests es de al menos el 20 % del código (no cuentan los tests de aceptación).
- Al menos el 50 % de las historias de usuario o requisitos implementados en el sprint tienen buenos criterios de aceptación.
- Compilación y gestión de dependencias (p.ej. descarga de bibliotecas externas) están basada en scripts (Gradle, Maven etc.).
- La documentación arquitectural incluye una discusión adecuada sobre razones arquitecturales.
- Toda la documentación del proyecto está bajo control de versiones.

**Se valorará especialmente (se quiere sacar sobresaliente) si además de lo anterior**

- La cobertura de tests supera el 30 % del código (sin contar los tests de aceptación).
- Todas las historias de usuario o requisitos implementados tienen buenos criterios de aceptación.
- Hay tests de aceptación automáticos para al menos un criterio de aceptación implementado en el sprint.
- La construcción se ha automatizado más allá de la compilación y gestión de dependencias (posibilidades: trigger automático al hacer commit, se hace un análisis estático del código en cada construcción, lanzamiento de tests automáticos, generación de binarios (jars, wars, apks...) automática, despliegue automático en servidores etc.).

## 7.7. Bibliografía comentada

- **Rubin KS (2012). *Essential Scrum: A Practical Guide to the Most Popular Agile Process.* Addison-Wesley Signature Series (Cohn). Addison-Wesley Professional, 1st edition:** Este libro, orientado al público profesional, es un excelente manual de de la metodología ágil de gestión de procesos Scrum. Partiendo de los fundamentos ágiles de la metodología, el libro describe en detalle y justifica la validez y aplicabilidad de los distintos elementos de Scrum. Además de estar bien escrito y resultar bastante fácil de leer a pesar de su longitud, merece la pena mencionar también algunos de los diagramas que acompañan al texto, que consiguen transmitir de forma sucinta pero muy clara muchos de los fundamentos.
- **Larman C, Vodde B (2010). *Practices for Scaling Lean & Agile Development: Large, Multisite, and Offshore Product Development with Large-Scale Scrum.* Addison-Wesley Professional, 1st edition:** Un libro

sobre gestión de proyectos ágiles con Scrum bastante centrado en como tener en cuenta distintos aspectos técnicos en proyectos grandes. La estructura no es muy lineal y en general los capítulos son colecciones de ideas sueltas relacionadas, pero aún así es un libro que logra transmitir bien la experiencia de los autores, y plantea y defiende muchas ideas interesantes sobre como combinar por ejemplo buenas prácticas de diseño de software o pruebas con metodologías ágiles.

- **Pham A, Pham PV (2011). *Scrum in Action: Agile Software Project Management and Development.* Course Technology, 1st edition:** En este caso nos encontramos frente a un libro que en mi opinión es de inferior calidad al de Rubin y que en general no va más allá en contenidos. Salvo algunos aspectos sobre la adaptación de Scrum a distintas circunstancias (organizaciones, prácticas anteriores etc.) que tienen cierto interés, el resto del libro no me parece especialmente recomendable.

- **Kniberg H, Skarin M (2010). *Kanban and Scrum: Making the Most of Both.* InfoQ Enterprise Software Development. C4Media Inc:** Un libro corto, poco más de cien páginas, que consigue presentar todos los conceptos esenciales de Kanban y Scrum a lectores con poco tiempo. Trata todos los temas importantes pero, necesariamente, de manera muy somera, así que por sí solo se queda bastante corto de contenidos. De todas formas es una lectura recomendable como primera aproximación a las metodologías de gestión de proyectos ágiles.

- **Kniberg H (2011). *Lean from the Trenches: Managing Large-Scale Projects with Kanban.* InfoQ Enterprise Software Development. The Pragmatic Programmers, LLC:** Aunque en principio más focalizado en la metodología Kanban, el libro hace constantes referencias a Scrum mientras explica cómo se llevó a cabo un sistema de información para la policía nacional de Suecia siguiendo estas metodologías, con un equipo que en la segunda mitad del proyecto ya contaba con más de 60 personas. Una referencia muy buena para entender cómo se aplican estas metodologías a proyectos de cierto tamaño, y que recoge muy bien las experiencias adquiridas en aquel proyecto.

- **Humble J, Farley D (2010). *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation.* Signature Series. Addison Wesley:** Este libro describe con todo detalle lo que es la entrega continua, una metodología de automatización del despliegue de proyectos de software que ha ido evolucionando a partir de las técnicas de integración continua que a su vez se basan en las aproximaciones de integración y compilación muy frecuentes que algunas empresas grandes, como Microsoft, pusieron en marcha a finales de los 90. El libro es denso de contenidos y desde luego no es una lectura ligera, pero condensa mucha experiencia en despliegue de proyectos de software y es la lectura imprescindible para cualquiera que tenga que desplegar un sistema mediano o grande y no quiera re-descubrir las técnicas existentes.

Además de estos, muchos de los textos utilizados como referencia en la asignatura de Proyecto Software y ya comentados en la Sección 6.7 son también útiles en esta asignatura.

# Bibliografía


37Signals (2006). *Getting Real*. https://gettingreal.37signals.com/.

Abran A (2010). *Software Metrics and Software Metrology*. IEEE Computer Society & John Wiley and Sons Inc.

ACM/IEEE-CS Joint Task Force on Computing Curricula (2013). Computer science curricula 2013. Technical report, ACM Press and IEEE Computer Society Press.

Alcubilla EA, (ed) (2016). *Código de Universidades*. Códigos electrónicos. Agencia Estatal Boletín Oficial del Estado.

Allegre C, Berlinguer L, Blackstone T, Rüttgers J (1998). Sorbonne Joint Declaration. Joint declaration on harmonisation of the architecture of the European Higher Education System.

Ambler SW (2014). The non-existent software crisis: Debunking the chaos report. *Dr. Dobb's Journal*.

ANECA (2005). Título de Grado en Ingeniería Informática. Libro Blanco, Agencia Nacional de Evaluación de la Calidad y Acreditación (ANECA).

Armstrong JS, (ed) (2001). *Principles of Forecasting: A Handbook for Researchers and Practitioners*. International Series in Operations Research & Management Science. Springer.

Brooks FP (1995). *The Mythical Man-month: Essays on Software Engineering*. Essays on software engineering. Addison-Wesley Publishing Company.

CCII (2015). Estudio nacional sobre la situación laboral de los profesionales del sector de tecnologías de la información. Technical report, Consejo General de Colegios Profesionales de Ingeniería Informática.

Chacon S, Straub B (2014). *Pro Git*. Apress, second edition.

Chemuturi M, Cagley, Jr. TM (2010). *Mastering Software Project Management. Best Practices, Tools and Techniques*. J. Ross Publishing.

Chen CH (2009). Reframing narrative cases for ill-structured contexts: The design with learners in mind. *Journal of Learning Design*, 3(1).

Clement MC (2010). *First Time in the College Classroom. A Guide for Teaching Assistants, Instructors, and New Professors at All College and Universities*. Rowman & Littlefield Education.

Dabbagh N, Dass S (2013). Case problems for problem-based pedagogical approaches: A comparative analysis. *Computers & Education*, 64:161–174.

de Educación y Ciencia M (2006). Ficha Técnica de Propuesta Título Universitario de Grado de Ingeniero/a en Informática. Consejo de Coordinación Universitaria, Ministerio de Educación y Ciencia (MEC).

Denning P, Comer DE, Gries D, Mulder MC, Tucker A, Turner AJ, Young PR (1989). Computing as a discipline. *Communications of the ACM*, 32(1).

DIIS (2015). Memoria de la actividad docente e investigadora curso 2014/2015. Technical report, Departamento de Informática e Ingeniería de Sistemas (DIIS), Universidad de Zaragoza.

Dym CL, Agogino AM, Eris O, Frey DD, Leifer LJ (2005). Engineering design thinking, teaching, and learning. *Journal of Engineering Education*, 94(1):103–120.

EHEA ministerial conference (2015). Yerevan communiqué.

European Commission/EACEA/Eurydice (2015). *The European Higher Education Area in 2015: Bologna Process Implementation Report*. Luxembourgh.

Eveleens JL, Verhoef C (2010). The rise and fall of the chaos report figures. *IEEE Software*, pp 30–36.

EY (2014). Spotlight on oil and gas megaprojects. Technical Report EYG No. DW0426, EYGM Limited.

EY (2015). Opportunities to enhance capital productivity: mining and metal megaprojects. Technical Report EYG No. ER0235, EYGM Limited.

Fogel K (2005). *Producing Open Source Software*. http://producingoss.com/en/index.html, 1st edition.

Fry H, Ketteridge S, Marshall S, (eds) (2009). *A Handbook for Teaching and Learning in Hihger Education. Enhancing Academic Practice*. Routledge. Taylor & Francis Group, third edition.

Group S (2013). CHAOS manifesto 2013. think big, act small. Technical report, Standish Group.

Hughes J (2009). *Philosophy of Technology and Engineering Sciences*, cap. Practical Reasoning and Engineering. Elsevier.

Humble J, Farley D (2010). *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation*. Signature Series. Addison Wesley.

ITU (2015). Measuring the information society report 2015. Technical report, International Telecommunication Union.

Johnson L, Adams Becker S, Estrada V, Freeman A (2015). NMC horizon report: Edición educación superior 2015. Technical report, The New Media Consortium, Austin, Texas.

Joint declaration of the European Ministers of Education (1999). The Bologna Declaration of 19 June 1999.

Jonassen D (2011). Supporting problem solving in PBL. *The Interdisciplinary Journal of Problem-Based Learning*, 5(2):95–119.

Jonassen D, Strobel J, Lee CB (2006). Everyday problem solving in engineering: Lessons for engineering educators. *Journal of Engineering Education*, 95(2):139–151.

Jonassen DH (1997). Instructional design models for well-structured and ill-structured problem-solving learning outcomes. *Educational Technology Research and Development*, 45(1):65–94.

Kniberg H (2011). *Lean from the Trenches: Managing Large-Scale Projects with Kanban.* InfoQ Enterprise Software Development. The Pragmatic Programmers, LLC.

Kniberg H, Skarin M (2010). *Kanban and Scrum: Making the Most of Both.* InfoQ Enterprise Software Development. C4Media Inc.

Koen BV (2003). *Discussion of the Method. Conducting the Engineer's Approach to Problem Solving.* Oxford University Press.

Larman C, Vodde B (2010). *Practices for Scaling Lean & Agile Development: Large, Multisite, and Offshore Product Development with Large-Scale Scrum.* Addison-Wesley Professional, 1st edition.

Lopez-Pellicer FJ, Béjar R, Latre MÁ, Nogueras-Iso J, Zarazaga-Soria FJ (2015). GitHub como herramienta docente. En *Actas de las XXI Jornadas de la Enseñanza Universitaria de la Informática, JENUI 2015.*, pp 66–73.

Mas M, de Guevara Radoselovics JF (2015). The 2015 PREDICT Report. An Analysis of ICT R&D in the EU and Beyond. Technical Report EUR 27510 EN, Institute for Prospective Technological Studies. Joint Research Center. European Commission.

Menéndez Mato JC, Gayo Santa Cecilia ME (2014). *Derecho e informática: ética y legislación.* JMB Bosch Editor.

Nagl M (2013). Informatics doctorates in europe: Some facts and figures. Technical report, Informatics Europe.

Nicol DJ, MacfarlaneDick D (2006). Formative assessment and selfregulated learning: a model and seven principles of good feedback practice. *Studies in Higher Education*, 31(2):199–218.

OASI (2014). Análisis económico-financiero de las empresas del sector TIC en Aragón. Resumen ejecutivo. Technical report, Observatorio Aragónes de la Sociedad de la Información. Gobierno de Aragón.

Ortega y Gasset J (2007). *Misión de la universidad*. Biblioteca Nueva.

Pandian CR (2007). *Applied Software Risk Management: A Guide for Software Project Managers)*. Auerbach Publications, Taylor & Francis Group.

Perrenet JC, Bouhuijs PAJ, Smits JGMM (2000). The suitability of problem-based learning for engineering education: Theory and practice. *Teaching in Higher Education*, 5(3):345–358.

Petroski H (2011). *An Engineer's Alphabet. Gleanings from the Softer Side of a Profession*. Cambridge University Press.

Pham A, Pham PV (2011). *Scrum in Action: Agile Software Project Management and Development*. Course Technology, 1st edition.

Pressman RS (2010). *Software Engineering: A Practitioner's Approach*. McGrawHill Higher Education. McGrawHill, 7th edition.

Project Management Institute (2013). *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*. Project Management Institute, Inc., 5th edition.

Rogers GFC (1983). *The Nature of Engineering. A philosophy of technology*. The MacMillan Press Ltd., third edition.

Rubin KS (2012). *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Addison-Wesley Signature Series (Cohn). Addison-Wesley Professional, 1st edition.

Savin-Baden M, Major CH (2004). *Foundations of Problem-based Learning*. Society for Research into Higher Education & Open University Press. McGraw-Hill Education.

Shrestha PP, Burns LA, Shields DR (2013). Magnitude of construction cost and schedule overruns in public work projects. *Journal of Construction Engineering*.

Simon HA (1996). *The Sciences of the Artificial*. MIT Press, Cambridge, MA, USA, third edition.

Simonson M, Smaldino S, Zvacek S (2015). *Teaching and Learning at a Distance. Foundations of Distance Education*. Information Age Publishing, Inc., sixth edition.

Sommerville I (2011). *Software Engineering*. Addison-Wesley - Pearson, 9th edition.

Strobel J, Wang J, Weber N, Dyehouse M (2013). The role of authenticity in design-based learning environments: The case of engineering education. *Computers & Education*, 64:143–152.

Universidad de Zaragoza (2011). *Universidad Zaragoza: Innovadora, Comprometida, Internacional, Universidad de todos*. Vicerrectorado de Relaciones Institucionales y Comunicación. Universidad de Zaragoza.

Vásquez GH, (ed) (2012). *Filosofía de la educación*. Number 29 in Enciclopedia Iberoamericana de Filosofía. Editorial Trotta. Consejo Superior de Investigacionese Científicas.

Weaver P (2007a). The origins of modern project management. En *Fourth annual PMI college of scheduling conference, Vancouver*.

Weaver P (2007b). Trends in modern project management, past, present & future. En *PM OZ, Queensland*.