Trabajo de Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Metodologías de Gestión de Proyectos. Estudio comparativo y propuesta de guía de elección.

Autor: Ángel Jesús Girón Sevillano
Tutoras: Irene Fondón García
Auxiliadora Sarmiento Vega

Dpto. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Trabajo de Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Metodologías de Gestión de Proyectos. Estudio comparativo y propuesta de guía de elección.

Autor:
Ángel Jesús Girón Sevillano

Tutoras:
Irene Fondón García
Profesora Titular

María Auxiliadora Sarmiento Vega
Profesora Contratada Doctora

Dpto. de Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2021

Trabajo de Fin de Grado: Metodologías de Gestión de Proyectos. Estudio comparativo y propuesta de guía de elección.

Autor: Ángel Jesús Girón Sevillano

Tutoras: Irene Fondón García

Auxiliadora Sarmiento Vega

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2021

El Secretario del Tribunal

A mi familia

A mis amigos

A mis maestros

Agradecimientos

Con este trabajo de investigación y análisis he tenido la oportunidad de hacer una aportación muy personal, fruto de mi paso por la Escuela Técnica Superior de Ingeniería de Sevilla y de mi experiencia laboral, para cerrar una etapa esencial de mi vida.

Por lo tanto, es inevitable agradecer a todos y cada uno de los profesores y profesoras que he tenido durante mi etapa de estudiante, así como a todos los compañeros y compañeras de clase. Del mismo modo, son responsables de lo que soy hoy en día, profesionalmente hablando, mis compañeros, compañeras, responsables y jefes de las empresas por las que he pasado.

Como si de un triángulo invertido se tratara, voy a ir acotando la extensa lista de agradecimientos que podría extenderse de forma incontrolada.

En el primer nivel de referencia personal, agradecer a Irene Fondón y Auxiliadora Sarmiento la ayuda y conocimientos aportados, brindándome el privilegio de realizar una aportación personal al elemento culmen de mi etapa de estudiante.

Vital, desde mi entrada en la escuela, ha sido contar con mis *peppers*, compañeros de carrera que se convirtieron en amistades imperecederas, al mismo tiempo que nos apoyábamos y nos hacíamos mejores, no sólo profesional e intelectualmente, sino humanamente. Como se titula la canción, "sabéis quiénes sois".

Necesaria en todo momento es la confianza y el empujoncito que nos hace ver a dónde hemos llegado y lo que somos capaces de conseguir. Y para eso, siempre he tenido a la mejor compañera.

Y todo la anterior sería ficción sin el vértice del triángulo invertido y, por lo tanto, la base de todo… mis padres y mis hermanos.

Resumen

La clave del éxito en el resultado de un proyecto es la gestión que se haya hecho de éste durante su ciclo de vida. Determinará la diferencia para obtener un producto de valor, que cumpla los objetivos dentro de unos tiempos y con unos recursos concretos, de forma que el interesado quede satisfecho y el proveedor consiga un balance positivo.

Es necesario conocer cómo ha evolucionado la tecnología para entender cómo, de forma paralela, la gestión de proyectos también evoluciona y se hace más compleja y específica. Para entender dónde nos encontramos y cómo hemos llegado hasta aquí, es obligatorio realizar una contextualización de esta disciplina, apoyada en el avance tecnológico y en la evolución de la industria.

Con el objetivo de aportar una guía o claves subjetivas para elegir una metodología de gestión adaptada a un proyecto concreto, me apoyo en un trabajo de documentación, investigación y experiencia personal en el marco de la gestión de proyectos, haciendo un recorrido por las metodologías más utilizadas y popularizadas.

La recopilación de metodologías, sus elementos distintivos y la comparativa entre éstas, así como el análisis y evaluaciones empíricas publicadas de manera oficial, nos proporcionan una base de conocimientos para llevar a la práctica este arte, pues, aunque se nos proporcionen las herramientas, somos nosotros quienes debemos usarlas para crear la obra.

Abstract

The key to success in the result of a project is the management that has been made of it during its life cycle. It will determine the difference to obtain a product of value, which meets the objectives within a few times and with specific resources, so that the interested party is satisfied, and the supplier achieves a positive balance.

It is necessary to know how technology has evolved to understand how, in parallel, project management also evolves and becomes more complex and specific. To understand where we are and how we got here, it is mandatory to contextualize this discipline, supported by technological progress and the evolution of the industry.

With the aim of providing a guide or subjective keys to choose a management methodology adapted to a specific project, I rely on a work of documentation, research and personal experience in the project management framework, taking a tour of the most used methodologies and popularized.

The compilation of methodologies, their distinctive elements and the comparison between them, as well as the analysis and empirical evaluations published officially, provides us with a knowledge base to put this art into practice, because, even if the tools are provided, we are we who should use them to create the work.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN A LA GESTIÓN DE PROYECTOS DE TELECOMUNICACIONES

Trabajar mantiene a todos alerta, la estrategia proporciona una luz al final del túnel, pero la gestión del proyecto es el motor del tren que hace avanzar a la organización.

- Joy Gumz -

El objetivo de este primer bloque es contextualizar y conocer los elementos principales de un proyecto de Telecomunicaciones. Para ello, es necesario definir qué es un proyecto, de forma que podamos relacionarlo conceptualmente con su gestión. De esta forma, podremos describir la evolución en el tiempo de la gestión de proyectos en paralelo al desarrollo de la tecnología.

Tomando como referencia la guía *PMBOK*[1][1], un proyecto se define como *un esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único*. Parafraseando la continuación de esta misma definición del *Project Management Institute*, un proyecto debe tener un principio y un final establecidos y definidos, pudiendo estar marcado el final por el logro de los objetivos, por la imposibilidad de cumplirlos, o cuando ya no existe la necesidad que lo originó.

Es decir, el concepto de "proyecto" no puede existir sin el de "tiempo". Por lo tanto, para poder gestionar un proyecto, es vital hacer una gestión del tiempo en función de los requerimientos de éste y los recursos disponibles para abordarlo. Evidentemente, entre el inicio y el final del proyecto hay más fases y elementos, que veremos en detalle en un apartado de este primer bloque.

Ya situados, vamos a comenzar con una pequeña descripción de la evolución de la tecnología base de las Telecomunicaciones en paralelo a la evolución de la gestión de proyectos, para seguir con el impacto en el mercado y organismos públicos y la definición de los elementos y fases en un proyecto, tomando como referencia la *Guía ComSoc* [2] para la gestión de proyectos de Telecomunicaciones. Es necesario conocer la evolución de la tecnología y su industria para asimilar la especialización de la gestión de proyectos.

[1] *PMBOK* son las siglas de *Project Management Body of Knowledge*, una guía desarrolada por el *Project Management Institue (PMI)*, que establece un criterio de buenas prácticas relacionadas con la gestión, la administración y la dirección de proyectos mediante la implementación de técnicas y herramientas.

1.1 Evolución de la tecnología de las Telecomunicaciones

Las tecnologías de Telecomunicación son, relativamente, las más jóvenes, puesto que son resultado de la evolución, aplicación y fusión de otras ciencias básicas, como la física y la matemática. Conforman una industria clave en esta nueva era de la comunicación y la información, que podríamos considerar como una revolución tecnológica, incluso social, por la expansión y el condicionamiento, ya necesidad, al uso de las mismas.

Actualmente, es impensable imaginar nuestro día a día sin productos basados en las telecomunicaciones, que van desde el primer teléfono por cable, a los actuales smartphones, así como cualquier comunicación a distancia (geolocalización, redes IP, radiocomunicación, …). Vivimos en la era de la comunicación, fruto de una evolución muy intensa en un pequeño periodo de tiempo. No hay más que comparar dos generaciones para comprobar el inmenso avance de las tecnologías de telecomunicación. Y todo este desarrollo y expansión requiere de una gran cantidad de proyectos bien gestionados, y de personas lo suficientemente formadas que proporcionen calidad a los productos.

Para remontarnos a sus orígenes y establecer un punto de partida de esta tecnología, podemos considerar su nacimiento con la aparición de la telegrafía óptica, con la cual empezaron a construirse las primeras redes de telecomunicación [3]. Este hecho tuvo lugar en la revolución francesa de mano de Claude Chappe, quien ideó una red óptico-mecánica en el año 1792.

A continuación, vamos a hacer un recorrido por los avances e inventos más relevantes en el terreno de las telecomunicaciones hasta el día de hoy [4].

En 1829, Joseph Henry, basándose en los avances y descubrimientos de Ampère y Faraday, creó el primer telégrafo, siendo Samuel Morse quien, en 1844, realizó la primera transmisión telegráfica entre las ciudades de Washington y Baltimore.

El siguiente avance relevante fue la instalación del primer canal de comunicación por cable entre América y Europa en 1866, propiciando que, diez años después, Alexander Graham Bell estableciera y registrara la patente del teléfono. Al año siguiente se funda la empresa Bell, dentro de la cual se patenta un transmisor mejorado, por Thomas Edison.

No fue hasta 1878 cuando apareció lo que se puede considerar la primera centralita de conmutación de llamadas, que no era más que un tablero de conmutación manual con capacidad para 21 abonados. La conmutación de llamadas se automatizó por primera vez en Indiana en 1892, con la instalación de la primera central telefónica automática por Almon Brown Strowger.

El primer canal por radiocomunicación apareció en 1901, de la mano de Marconi, que realizó el primer enlace transatlántico a través de ondas de radio.

La línea telefónica de mayor longitud hasta ese momento, que iba de Nueva York a San Francisco, se inauguró en 1915. Este hecho demuestra lo jóvenes que son las telecomunicaciones y la evolución exponencial que han vivido en tan corto periodo de tiempo.

Es, por manos del francés Alec Reeves en 1927, cuando se crea la Modulación por Pulsos Codificados o PCM, uno de los pilares básicos o fundamentos de la tecnología digital.

Las computadoras hacen su primera aparición en la década de los 40’s, momento a partir del cual empiezan a evolucionar de forma titánica, clave para el desarrollo de las telecomunicaciones.

En esta misma década, en concreto en el año 1946, AT&T saca a la luz el primer sistema de telefonía móvil. Antes de cerrar la primera mitad de siglo, en 1948, nace un elemento esencial en la optimización de las telecomunicaciones, el transistor.

Si la primera mitad del siglo XX fue impresionante en cuanto a creaciones y avances, la segunda mitad se puede considerar una auténtica revolución. En este momento, no sólo siguen teniendo lugar hitos claves en las telecomunicaciones, sino que aparecen y se desarrollan las bases de la gestión de proyectos, debido a la envergadura y complejidad de los nuevos avances tecnológicos, que requieren de una mayor planificación y gestión de recursos.

Ejemplo de ello es la instalación, en 1956, del primer cable transatlántico de teléfono, o el lanzamiento del primer satélite de comunicaciones, el Telstar 1, y la instalación del primer sistema de transmisión digital, el

llamado T1, ambos en 1962.

En 1966 aparece de nuevo en escena AT&T, esta vez para crear el primer módem telefónico.

Antes de cerrar esta década, en 1969, se comienzan a dar los primeros pasos en la creación de redes, siendo Arpanet la primera red de computadoras conocida. Justo un año después tiene lugar otro de los momentos clave para optimizar las redes, la fabricación de las primeras fibras ópticas. Poco tiempo después, en 1973, se crea la tecnología Ethernet gracias al aporte de Bob Metcalfe.

Aunque este mismo año vio nacer el primer teléfono móvil o celular, no se permitió su uso comercial hasta el año 1982.

En 1988 se lleva a cabo la instalación del primer canal transatlántico de fibra óptica, entre Estados Unidos y Francia, que permitió que, al año siguiente, naciera Internet. Este hito, esencial en la evolución de las telecomunicaciones, tuvo lugar en el CERN o Instituto Europeo de Investigación de Física, por Tim Barners Lee.

La tecnología GSM se crea en 1991, y la tecnología VoIP en 1996, básicas en la comunicación inalámbrica, que se regulan en el estándar IEEE-802.11, aprobado en 1997.

A partir de este momento, los aportes se centran en la optimización del rendimiento de las comunicaciones, como el lanzamiento de VDSL2 en 2005, el estándar IEEE-802.11n (transmisiones inalámbricas de 600 Mpps) en 2009, y la investigación hasta ahora sobre las NGN o Redes de Nueva Generación y servicios Cloud.

Y, ¿por qué recordamos la historia de las Telecomunicaciones en este documento? Pues porque es imprescindible tener presente su evolución en el tiempo para entender la evolución de su gestión. El aumento de la complejidad en los proyectos debido a la evolución de la tecnología conlleva un refinamiento y una especialización de los métodos para gestionar los proyectos.

En el siguiente apartado, realizaremos un recorrido a través del tiempo por la gestión de proyectos, de la misma forma que acabamos de hacer con el nacimiento y evolución de las telecomunicaciones.

De esta manera, podremos visualizar, en paralelo, el desarrollo tanto de la tecnología como de la gestión de los proyectos basados en esta tecnología.

1.2 Evolución de la Gestión de Proyectos

Aunque cualquier obra o producto en el transcurso de la humanidad haya requerido, de manera más o menos explícita, de una gestión, no es hasta después de la Segunda Guerra Mundial, a mediados del siglo XX, cuando nace la Gestión de Proyectos como tal, consecuencia y necesidad tras la evolución de la tecnología. En este momento se reconoce como herramienta necesaria e indispensable para poder llevar a cabo actividades tecnológicas de complejidad [5].

Es momento de hacer una definición más detallada de la gestión de proyectos, parafraseando una publicación de la Revista Académica e Institucional de la UCPR [6]:

La gestión de proyectos permite a las organizaciones administrar y optimizar sus recursos, para la entrega de un producto de cierto alcance, en un periodo de tiempo predeterminado. Por lo tanto, un proyecto puede ser percibido como un sistema de procesos interrelacionados integrados por un objetivo común. La dirección de proyectos, desde esta perspectiva, es la disciplina que hace más eficientes, efectivos y armónicos a estos procesos y al sistema en su totalidad.

Para hablar del nacimiento de la Gestión de Proyectos modernos, hay que remontarse a su evolución [7]. Imprescindible nombrar a Polaris Bernard Schcriever, arquitecto de desarrollo de misiles balísticos de la Fuerza Aérea y el Programa Militar Espacial de EEUU, que en 1954 hizo converger todos los elementos de un proyecto en la programación y presupuesto del mismo. El objetivo era ejecutar por elementos de tiempo en vez de por fases, método que conllevó a una reducción considerable de los tiempos de ejecución de los proyectos. En este mismo año, Peter Drucker, nacido en Austria, implementa y populariza la primera guía de prácticas de

administración por objetivos o MBO.

Pero, previamente, cabe destacar la creación del Diagrama de Gantt, en 1910, por Henry Laurence Gantt, que recoge el tiempo previsto de dedicación para cada tarea en un intervalo total de tiempo (que podría ser la duración del proyecto, o entregas, versiones o subdivisión del mismo). Es una herramienta que se sigue utilizando actualmente, pues permite ilustrar el tiempo de inicio, fin y duración de una tarea dentro del tiempo total de un proyecto. Este elemento es digno de ser mostrado, pues es mucho más fácil comprender su objetivo visualmente que con una definición, como se muestra en la *Figura 1-1*:

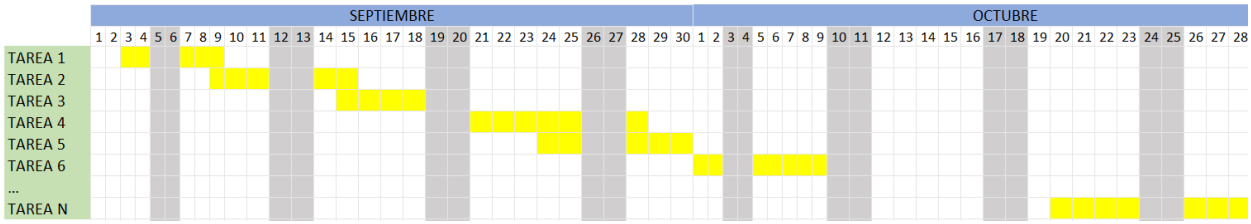

Figura 1-1. Diagrama de Gantt.
Realización propia

Otro evento histórico asociado a la evolución de la gestión de proyectos fue el Proyecto Manhattan, iniciado en 1942, dirigido por Leslie Groves, general del Cuerpo de Ingenieros del Ejército de EEUU. Consistía en la investigación y desarrollo de las primeras armas nucleares durante la Segunda Guerra Mundial. Debido a la complejidad del proyecto, se empezaron a fortalecer los cimientos de la Teoría de la Organización para gestionarlo. Además del gran peso tecnológico, requería de una robusta cooperación entre varios países, pues fue un proyecto conjunto de Estados Unidos, Canadá e Inglaterra.

En 1956 se funda la AACE Internacional o Asociación Americana de Ingenieros de Costos, institución para normalizar la gestión de planificación de proyectos. Por primera vez, nace un organismo planteado exclusivamente para gestionar proyectos de ingeniería.

A medida que se va focalizando la gestión del proyecto en la optimización del tiempo con el consiguiente aumento de beneficios, se van desarrollando métodos, como el famoso Método del Camino Críticos, por DuPony y Remington Rand en 1957. Este método se basa en el cálculo de las tareas necesarias para completar un proyecto en función de cada fase. Lo aplicaron por primera vez en la gestión del mantenimiento de plantas químicas, con resultados extraordinarios, ahorrando un millón de dólares a la compañía el primer año de su aplicación. Como consecuencia, la Marina de los Estados Unidos la desarrolla y aplica en el programa de Evaluación y Revisión Técnica o PERT, en el año 1958, para el programa de misiles del anteriormente nombrado Polaris. El método está enfocado en el análisis del tipo de tarea y la cantidad específica de tiempo necesaria para abordar cada tarea.

Como en todas las tecnologías, su perfilamiento y desarrollo ocurre con la necesidad del posicionamiento en el área de defensa. Es por ello que, en 1962, el Departamento de Defensa de Estados Unidos sigue indagando en las metodologías de gestión de proyectos, creando un modelo que divide el groso del proyecto en subproyectos o bloques más pequeños que se organizan y relacionan entre sí en estructuras de árbol jerárquicas o WBS (Work Breakdown Structure), con el fin de centrarse en la ejecución de módulos concretos.

Pocos años después, se fundan dos organismos relevantes que promueven y tratan la gestión de proyectos como una disciplina específica y una profesión con identidad propia, vigentes y con un peso considerable a día de hoy: el IPMA o Asociación Internacional de Gestión de Proyectos, en Suiza, en el año 1965, y el PMI o Instituto de Gestión de Proyecto, en EEUU en el año 1969.

En 1989 se crea, para ser aplicada inicialmente en el gobierno de Reino Unido, la Metodología de Dirección de Proyectos en Entornos Controlados PRINCE2.

A partir de este momento, se trabaja en la mejora de la planificación y optimización en la ejecución de los proyectos, a través de la descomposición en tareas para cubrir los requisitos iniciales en función de los recursos de los que se dispone, pudiendo monitorizar desviaciones para su corrección. Es decir, se busca constantemente

optimizar los procedimientos para lograr un producto dentro de los tiempos establecidos, con los recursos disponibles, de forma organizada y con capacidad de adaptación y correcciones. Para ello, no sólo se avanza en las metodologías tradicionales ya instauradas y definidas, sino que nace la necesidad de formalizar las claves y principios para la gestión adaptativa e iterativa.

Fruto de esta búsqueda incesante es la publicación del Manifiesto Ágil en 2001, que recoge las bases de todo método adaptativo e iterativo. Además, cabe destacar la aparición de innumerables métodos, guías y normas, como son el Método de Gestión de Proyectos V-MODELL. CMMI (en Alemania en el año 2002), la norma ISO10006 para Gestión de Calidad en Proyectos en 2003, la versión 4 de la Guía PMBOK en 2008, la norma ISO21500 basada en el desarrollo del PMI y de IPMA en 2012, la versión 5 de la Guía PMBOK en 2013 o la versión 6 de esta misma guía en 2017.

Esta evolución histórica de la gestión de proyectos podríamos clasificarla en siete bloques, basados en el desarrollo de la tecnología, en las necesidades de mercado y en las principales fuentes de cambio en el pasado y en el presente:

- **Década de los 50's: Ingeniería Hardware** → Desarrollo hardware con fines militares por parte de grandes organizaciones. El mayor peso lo tiene el hardware o maquinaria en sí, mucho mayor que el tiempo por persona. Es por ello que no hubiera aún metodologías ni estándares aplicados.
- **Década de los 60's: Desarrollo Software** → Aparición de lenguajes de programación debido a programas de la NASA, departamentos de Universidades y empresas privadas. El tiempo invertido por persona empieza a tomar un mayor peso, por lo que surge la necesidad de gestionarlo.
- **Década de los 70's: Formalización y procesos Waterfall** → Se invierten los costes, siendo el tiempo por persona el de mayor coste. Se avanza en la disciplina de gestión con métodos predictivos y modelos de estimaciones.
- **Década de los 80's: Productividad y escalabilidad** → Se mejora la escalabilidad y productividad de los métodos de la década anterior, creándose estándares y modelos internacionales. Además, el avance de la tecnología (procesadores más potentes, lenguajes de programación de alto nivel, optimización de canales de comunicación) aportan una productividad muy superior.
- **Década de los 90's: Procesos concurrentes VS procesos secuenciales** → Entre otros motivos, se expande y populariza Internet y la World Wide Web, que obliga a las corporaciones a sacar al mercado el producto lo más rápido posible, usando métodos concurrentes que aumentaran la eficacia y redujeran los riesgos. Además, crece el uso de la tecnología para por parte de no profesionales para uso personal y corporativo.
- **Primera década del 2000: Procesos Ágiles y aumento del valor** → Se empieza a utilizar masivamente Internet con la aparición de redes sociales, app's, servicios en streaming, redes corporativas, … que necesitan una integración de sistemas, un producto fiable en el que confiar y un anticipo por parte de las multinacionales a sacar un producto de calidad con el menor coste posible. Por ello, se apunta al valor del producto, que cubra con calidad las necesidades de las personas por el menor coste posible. Para ello es necesario crear métodos de gestión de proyectos que se adapten, reciclen y puedan ofrecer productos terminados cuanto antes, que se mejoren continuamente.
- **De 2010 hasta ahora: Globalización y sistemas de sistemas (macrosistemas)** → Es el momento de las súper-estructuras de redes y sistemas en todas las áreas: industriales, económicas y sociales. Nace la era de la información, del big data, de la inteligencia artificial, de la Smart life.

Estas alusiones son sólo algunas de las más relevantes, que nos van a servir, junto a los hechos históricos nombrados en el apartado anterior, para visualizar en un eje temporal cómo se ha desarrollado la gestión de proyectos en paralelo a la tecnología. Así podremos ilustrar y contextualizar cómo, a medida que la tecnología avanzaba y los proyectos tomaban una mayor complejidad, surgen nuevos métodos, normas y guías para poder abordarlos.

De esta manera, podemos recoger todos los acontecimientos citados en la *Figura 1-1*:

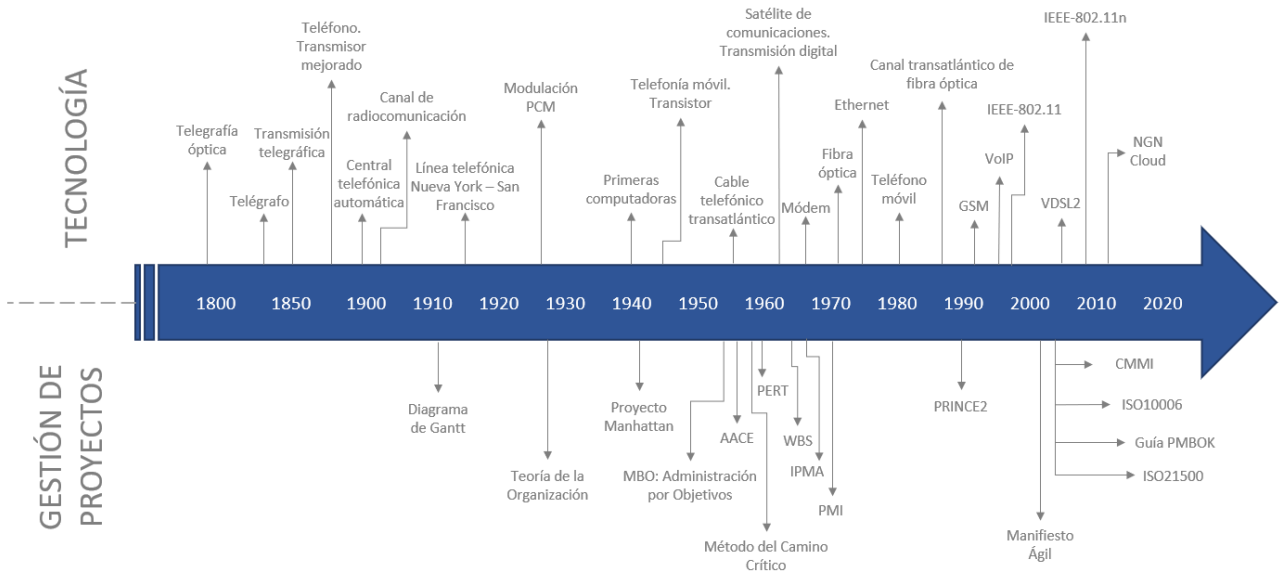

Figura 1-2. Eje cronológico de la evolución de la tecnología, en paralelo a la evolución de la gestión de proyectos. Realización propia

1.3 Evolución de la Industria de las Telecomunicaciones y su regularización

Para comprender la importancia y el impacto de las telecomunicaciones de cara a estudiar las metodologías de gestión de proyectos, es interesante hacer un pequeño recorrido de la evolución de su industria y su regularización por parte de las instituciones públicas [2].

Es evidente que, una industria que ofrece una necesidad en nuestro día a día, como es la comunicación a distancia entre personas, tenga un impacto económico tan considerable como para ser controlado por las instituciones y organismos públicos. Por ello, con el fin de ofrecer un servicio esencial a la ciudadanía, los gobiernos tuvieron que regularizar su mercado y sus precios en función del nivel de necesidad de cada servicio.

Antes de realizar el recorrido histórico, es interesante hacer una distinción entre los diferentes bloques de servicios para poder entender su regularización, pues no se aplicarían las mismas cuotas a un servicio esencial, como llamadas de emergencia que, a un servicio complementario, como podría ser el envío de archivos multimedia. Por este motivo, podríamos diferenciar los siguientes bloques de servicios en la industria de las telecomunicaciones:

- Servicios Locales
- Servicios de Larga Distancia
- Internet y Multimedia
- Nuevas tecnologías

En función del nivel de necesidad, su regularización ha tenido un trato diferente.

A continuación, basándonos en la mencionada *Guía ComSoc para la Gestión de Proyectos de Telecomunicaciones* [2], vamos a comentar el estado del monopolio de la industria y las diferentes competencias según los servicios y la tecnología.

1.3.1 Estado del monopolio

Puesto que cada país tiene su propia legislación y consideraciones, incluso hay países que no la tienen aún para esta industria, no vamos a entrar en detalles cuantitativos, sino que vamos a describir la tendencia generalizada de las organizaciones.

Inicialmente, la tendencia global de los países que disponían de una red de telecomunicaciones básica, pero suficiente y al alcance de la población, consideraron este servicio como un derecho básico que debía estar disponible a un precio razonable, a pesar te conllevar unos costes elevados. Este servicio básico eran las llamadas de voz y la capacidad de usar el teléfono a la hora de necesitar asistencia inmediata o alguna emergencia, como ayuda médica, ambulancias, bomberos o policial.

Obviamente, ofrecer una fiabilidad en la capacidad de este servicio lleva a cumplir con unos requisitos en la red, de modo que no se produzcan interrupciones en momentos críticos. Ya que es matemáticamente imposible confiar en el 100%, se impuso a las empresas de telecomunicaciones una disponibilidad de las líneas y de la red el 99,999% del tiempo. Es la causa por la que, cuando no tenemos línea, nuestro terminal sólo nos permita llamadas de emergencias.

Originalmente, ubicándonos en América del Norte y Europa, este servicio era proporcionado directamente por los gobiernos, o éstos imponían fuertes requisitos y restricciones a las empresas proveedoras del servicio. La regulación trataba de proteger a ambas partes, proveedor y cliente, controlando las tarifas de cada tipo y nivel de servicio, de manera que el básico fuese asequible para la población.

Ejemplo de esta regularización es que, la inmensa mayoría de los hogares tienen acceso a la red fija de telefonía, aunque desde hace unos años esté disminuyendo su uso por los servicios inalámbricos. Pero es innegable que era imposible no tener acceso a una conexión de teléfono fijo, fruto de la gestión del gobierno, que inicialmente controlaba la industria y trató de asegurar el servicio a toda la población a través de una infraestructura robusta y de calidad.

El primer servicio básico proporcionado fueron las llamadas locales, dentro de un área considerada y cubierta por una red independiente. Estas redes independientes o locales podían unirse entre ellas por conexiones controladas por empresas privadas. Evidentemente, las llamadas a otras áreas locales o internacionales tenían un coste mayor al no considerarse básicas, costes con los que las empresas equilibraban las pérdidas por el servicio básico local.

En la *Figura 1-3* se puede visualizar el servicio local o básico, y cómo pasaría a ser un servicio de larga distancia con coste añadido al permitir la comunicación entre distintas redes locales.

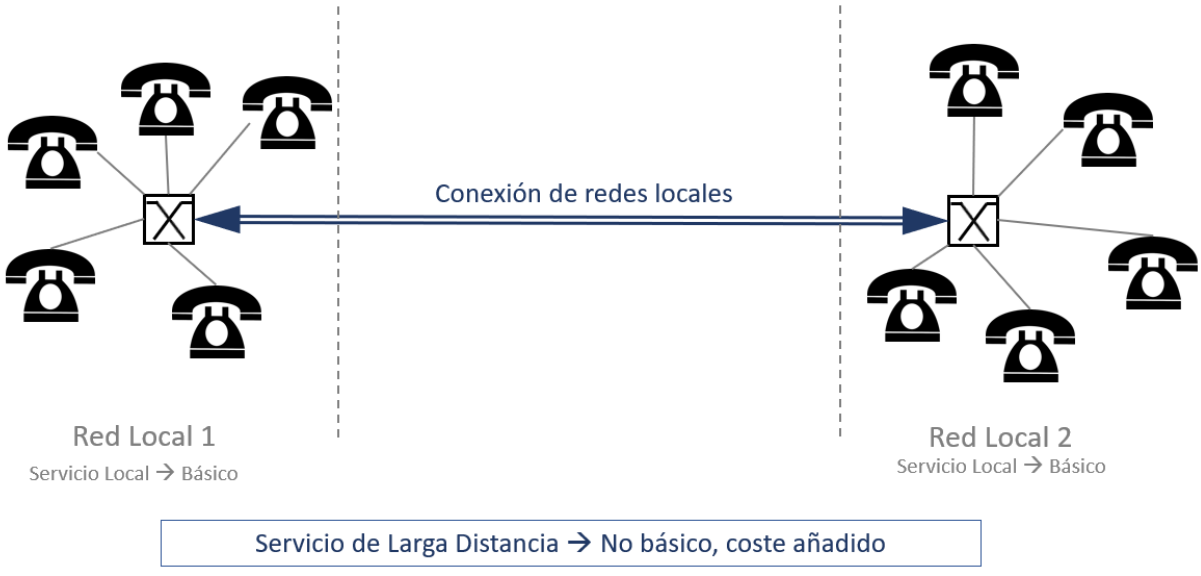

Figura 1-3. Servicio básico y de larga distancia en el estado inicial del monopolio.

Realización propia

La provisión de los servicios de datos (cuando aparecieron), tampoco fueron considerados básicos, sino servicios mejorados con un sobrecoste, quedando exentos de las restricciones del servicio básico. Aun así, también estaban regulados, aunque no conformaban una cantidad tan elevada para darle importancia.

En resumen, al inicio, la industria de las telecomunicaciones estaba monopolizada y controlada por los gobiernos, con el fin de proporcionar servicios de voz locales a tarifas bajas, que las empresas compensaban con el servicio de llamadas de larga distancia y otros servicios mejorados, que tenían un sobrecoste. Como consecuencia, podríamos decir que, a finales del siglo XX, las redes eran mayoritariamente analógicas y transportaban un 80% de servicios de voz y un 20% de servicios de datos, ya que el paso a digital se empezó a considerar a finales de los 80's.

1.3.2 Competencia por los Servicios de Larga Distancia

Esta siguiente etapa se basa en el cambio de prestación de servicios de larga distancia y datos.

Se comenzó a atacar al monopolio establecido con la aparición de nuevas tecnologías que permitían nuevos métodos de transporte del tráfico de voz y datos. Surgieron nuevos proveedores de servicios de larga distancia, empresas privadas que implementaron nuevos métodos y aportaban mayor eficiencia, que empezaron a ofrecer precios más competitivos en las llamadas de larga distancia. El ataque al monopolio se centró en el valor agregado, alquilando la capacidad de sus nuevas redes a los operadores establecidos inicialmente, además de usarlo para ofrecer sus propios servicios. Puesto que es muy reciente, para muchos nos es fácil comparar los precios de llamadas internacionales en un intervalo de diez años.

La sección anterior se cerró nombrando el paso del servicio de analógico a digital. Este hito se hizo posible con la llegada de la conmutación de paquetes, permitiendo optimizar el tráfico de voz, la calidad de las llamadas y la eficiencia del servicio de larga distancia. Esta optimización cambió las tornas de la proporción en el tráfico de la red, ya digital, pasando a ser de un 80% para los datos en el año 2000.

Como se puede intuir, el siguiente hito es el crecimiento de Internet, que pasó a ser un servicio en manos de la población. De forma paralela, se produjo un desarrollo considerable en el tráfico multimedia y streaming. Por lo tanto, la monopolizada red inicial enfocada en el tráfico de voz empezó a tambalearse, pues el desarrollo tecnológico empezó a ofrecer servicios de datos mucho más óptimos y a precios más competitivos que las empresas asentadas originalmente.

1.3.3 Competencia por los Servicios Locales

El siguiente paso fue el ataque a los servicios locales por parte de los nuevos proveedores de servicios o compañías.

Como se ha comentado en el apartado anterior, el desarrollo tecnológico permitió a las nuevas compañías posicionarse, crecer y hacerse conocer, poniéndose a la cabeza en los servicios de datos y larga distancia. Estos mismos proveedores, que ya ofrecían telefonía móvil por redes digitales, comenzaron a ofrecer telefonía fija a un precio razonable. Por lo tanto, sacaron al mercado un paquete de servicio completo, con línea fija, móvil y datos, que competían con ventaja sobre las tradicionales. Es más, la tendencia del usuario era tener todos los servicios unificados en una única factura y proveedor.

Se tambaleó el servicio de emergencias tradicional del monopolizado mercado, ya que, a través de la red de datos, no sólo podían realizar llamadas de emergencias, sino que era posible compartir la ubicación.

Es el punto de inflexión en el estado comercial de los servicios de telecomunicaciones. Evidentemente, los servicios siguen regularizados por los gobiernos, pero el avance de la tecnología, los paquetes de servicios y los precios competitivos de las emergentes compañías, acabaron con el monopolio de las empresas originarias y tradicionales.

1.3.4 Nuevo entorno de Telecomunicaciones: IP

Con el crecimiento de Internet, se comenzó a plantear el uso del protocolo IP para transportar, no sólo datos, sino servicios de voz. Era posible, ya que el tráfico de la voz en la red era de forma digital, idea de la que nace VoIP.

Sin embargo, esta transición fue muy compleja y para nada trivial. El descontrolado mercado de la competitividad, con compañías que se alimentaban las unas de las otras, no era un marco ideal para llevar a cabo la siguiente transformación de la red.

En este momento, el mercado digital abre sus puertas a la imaginación. Aparecen ofertas creativas sobre VOIP, como es el caso de Skype, Second Life, Facebook, … Las llamadas son totalmente gratuitas, pero se requiere de un terminal (una computadora y, con el tiempo, también smartphones) y tener servicio de Internet.

Estas empresas de servicios sobre VoIP también empiezan a crear y plantear nuevas fuentes de ingresos, como es la publicidad o las suscripciones premium. Es importante mencionar las nuevas plataformas de consumo multimedia, como Spotify, YouTube, Netflix, HBO, Amazon, …

La *Figura 1-4* es una representación de la red IP actual, los servicios y plataformas mostrados, y los dispositivos conectados.

Figura 1-4. Red IP de servicios actual.

Realización propia

Con esta nueva situación, ¿cómo es posible plantear, si quiera una monopolización? Es más, los servicios de nivel usuario o aplicaciones son los que comienzan a condicionar a los servicios de provisión de red. Hasta las propias compañías introducen en sus paquetes de ofertas suscripciones a plataformas.

Como conclusión, se puede afirmar que las telecomunicaciones están pasando a ser una parte del ecosistema general y cada vez menos una infraestructura independiente. Por ello, las nuevas puertas al mercado y la competitividad requieren de una gran idea y, sobre todo, de una gestión de proyectos óptima que dé como resultado un producto de calidad de forma rápida, con el menor coste posible, pero que a su vez genere beneficios.

Para cerrar este bloque es interesante hacer una reflexión para llevarnos toda esta información al terreno de la gestión de proyectos.

Los servicios de telecomunicaciones en sí son producto de la gestión de muchos proyectos: instalación de redes analógicas por áreas e interconexión entre ellas, paso de las redes de analógicas a digitales, optimización de canales, …

A su vez, es vital una buena gestión de proyectos para ofrecer productos desplegados o que necesitan las infraestructuras de telecomunicaciones: App's.

Al igual que la tecnología, la gestión de proyectos también ha tenido que evolucionar, refinando técnicas y metodologías para cada tipo de proyectos, necesidades y recursos.

La gestión es el resultado de etapas, elementos y procesos interconectados entre sí, recogidos en diferentes guías o métodos. En el siguiente apartado vamos a hablar de los elementos que tienen en común todos los proyectos, sean de la naturaleza que sean.

1.4 Elementos de la Gestión de Proyectos

La primera idea a transmitir es la de ver la gestión de proyectos no como una caja negra, sino como un conglomerado de gestiones de los distintos elementos que lo conforman. No debemos olvidar que un proyecto es un esfuerzo temporal que requiere del control y gestión de cada una de las variables que impactan en el tiempo: integración, alcance, dedicación, costes, calidad, recursos humanos, comunicaciones, riesgos, adquisiciones e interesados.

A su vez, es vital definir y hacer un seguimiento de los procesos que forman la dirección de proyectos, que podríamos clasificar en cinco grupos: inicio, planificación, ejecución, control y cierre.

La gestión y dirección de proyectos es el resultado de aplicar conocimientos, técnicas, habilidades y herramientas para el cumplimiento de unos requisitos. Para ello, es necesario estructurar, dividir y clasificar el trabajo a realizar.

Basándonos en la *Guía PMBOK* [1][2]y en la *Guía ComSoc* vamos a hacer una descripción tanto de las gestiones como de los grupos de procesos que conforman la gestión y dirección de un proyecto. No vamos a entrar en el detalle y la definición cerrado que nos ofrecen las guías anteriormente mencionadas, pues el propósito es dar a conocer y entender los elementos comunes en la gestión de proyectos para, posteriormente en otro bloque, describir cómo son tratados por cada una de las metodologías.

1.4.1 Gestión de gestiones

Es evidente que no existe un único elemento que afecte al tiempo y al esfuerzo necesarios para cerrar un proyecto y obtener un producto que cubra los requerimientos establecidos y acordados con el/la interesado/a.

Partiendo de esta idea, es necesario identificar qué factores provocan una alteración o condicionamiento del tiempo para tratarlos de forma explícita, de manera que sea posible equilibrar el tiempo total con los reajustes en el empleado en cada uno de estos factores.

Con este objetivo, podríamos considerar las siguientes gestiones elementales y comunes a cualquier proyecto [8]:

- **Gestión de la Integración del Proyecto**: Enfocado en la dirección del proyecto, se trata de las prácticas y acciones a llevar a cabo para poder identificar, definir, consolidar, y coordinar todos los procesos, de forma que se tenga un control sobre la realización del mismo y asegurar su finalización cubriendo los requisitos y cumpliendo con las expectativas de los/as interesados/as.

- **Gestión del Alcance del Proyecto**: Definición y control de qué se incluye en el proyecto y qué no para garantizar que el producto abarque los requerimientos acordados, obviamente, sin dejar nada fuera porque se incumplirían los requisitos, pero tampoco yendo más allá, pues se estaría invirtiendo un tiempo no acordado ni contemplado en la planificación ni el presupuesto.
- **Gestión de la Dedicación o Tiempo del Proyecto**: Manejo y control del cumplimiento de los plazos del proyecto.
- **Gestión de los Costes del Proyecto**: Planificación, estimación, financiación y control de los costes del proyecto para cumplir con el presupuesto acotado y aprobado.
- **Gestión de la Calidad del Proyecto**: Prácticas, políticas y procedimientos para asegurar la calidad del producto en la fase de ejecución o implementación del mismo. Es esencial para anteponer y dominar los objetivos y responsabilidades por parte del proveedor durante la realización del proyecto, asegurando un resultado óptimo.
- **Gestión de los Recursos Humanos del Proyecto**: Enfocado en el equipo del proyecto, aboga por la correcta jerarquización, definición de roles y responsabilidades individuales para que, de forma conjunta, todas las personas implicadas en el proyecto trabajen de forma eficiente, con una función y objetivos transparentes. La implicación de cada miembro es uno de los factores más importantes en la toma de decisiones de cara a la planificación, ejecución y resultados.
- **Gestión de las Comunicaciones del Proyecto**: Comunicación clara y eficiente entre los distintos implicados en el proyecto. Apunta hacia una correcta recopilación de la información y puesta en común, además de su disposición, con el fin de garantizar la comunicación entre los equipos y miembros de los mismos, trabajando sobre una información acordada conjuntamente y accesible en todo momento.
- **Gestión de los Riesgos del Proyecto**: Planificación de los riesgos a partir del análisis previo para identificarlos y asegurar una respuesta efectiva y solución de éstos. Así se evitan, en una mayor probabilidad, futuros contratiempos y desvíos en la planificación y estimación iniciales.
- **Gestión de las Adquisiciones del Proyecto**: Consideración de los costes externos, ya sea por compras de productos o servicios a terceros, así como subcontratación para implementaciones. Conlleva una administración y capacidad de gestionar contratos.
- **Gestión de los/as Interesados/as del Proyecto**: Impacto y participación de las personas, organizaciones o grupos involucrados en el proyecto para asegurar un análisis, estimación y definición eficientes de los requerimientos, un desarrollo eficaz de estrategias de gestión y una realimentación o *feedback* que permita mejoras en la ejecución.

En la *Figura 1-5* se agrupan las gestiones definidas anteriormente.

Figura 1-5. Factores a gestionar.

Realización propia

1.4.2 Etapas o conjuntos de procesos

En este apartado se generalizan los bloques, etapas o conjuntos de procesos comunes a cualquier proyecto [8]. Dichas etapas conformarían el llamada ciclo de vida del proyecto, que abarca desde el inicio (análisis, planteamiento, planificación) hasta su cierre (entrega del producto). Como su propio nombre indica, el ciclo de vida es el intervalo en el que el proyecto nace y se desarrolla, el plazo que hay para moldearlo antes de su cierre.

Al igual que los factores descritos en el apartado anterior, la ejecución de cada una de las etapas del ciclo de vida del proyecto definirá el resultado, hablando tanto a nivel de producto como de balance. En este caso, vamos a cambiar de nivel de abstracción en los conceptos, puesto que cada etapa requiere una gestión de los factores anteriores.

Podríamos considerar un proyecto como un programa o conjunto de etapas que, gestionadas y ejecutadas de forma coordinada, da como resultado un producto. Es decir, se requiere de una coordinación y un cumplimiento estricto en cada una de las etapas, pues un desajuste en una de éstas provoca, como mínimo, desajuste en la siguiente, ya que el tiempo y los recursos para la totalidad del proyecto son limitados y están acotados. Por ejemplo, una mala definición de requerimientos imposibilitaría una implementación óptima, que al final se traduce en contratiempos, sobrecostes, producto de mala calidad o proyecto sin beneficios.

Por este motivo, es necesario identificar las etapas o conjuntos de procesos. Una vez definidas, si gestionamos el tiempo dedicado a cada una, los riesgos, los recursos y el resto de los factores, conseguiremos avanzar secuencialmente hasta el final del proyecto, cumpliendo con las expectativas del cliente, los tiempos y los costes.

Las etapas comprenderán los conjuntos de procesos diferenciados, pero realmente interactúan y se relacionan entre ellos, de forma que, incluso el avance de etapa no siempre es secuencial, sino iterativo. Además, el conjunto de todas las etapas se puede considerar un patrón en la gestión total del proyecto, si éste se divide y se trata en distintas fases, pues cada una de éstas sería como un pequeño proyecto.

Una vez realizada estas aclaraciones, de forma general, podríamos identificar las siguientes etapas o conjuntos de procesos en un proyecto (o cada fase de un proyecto):

- **Inicio**: definición del proyecto o fase. Finaliza con la aprobación de los/as interesados/as y la autorización de la dirección.
- **Planificación**: definición de alcance, acotación y detallado de los objetivos y elección de la metología y acciones a aplicar para cumplir los objetivos. Finaliza con la aprobación y cierre de métodos a emplear, fechas por objetivos y requerimientos. Es una etapa dinámica, a la que se puede volver para una replanificación debida a cambios en la ejecución.
- **Ejecución**: materialización del trabajo planificado, implementación del producto como tal. Finaliza con un producto, versión o fase de éste. Va íntimamente ligado con la siguiente etapa, que sirve de realimentación.
- **Control**: evaluación y revisión del progreso en la ejecución para evitar riesgos y anticiparse a cambios necesarios. Es, a la vez, entrada y salida de la etapa de ejecución, pues permite la adaptabilidad de la ejecución ante cambios, a la vez que recoge las modificaciones a realizar con la replanificación correspondiente. Finaliza con la aceptación de la ejecución, tras su evaluación reiterada.
- **Cierre**: finalización de todas las etapas, que hace converger todos los procesos en un todo, un producto, versión o fase evaluada y apta, que se inició y se ejecutó en base a una planificación.

Podríamos visualizar, de forma generalizada, las etapas del ciclo de vida de un proyecto, versión o fase en la *Figura 1-6*.

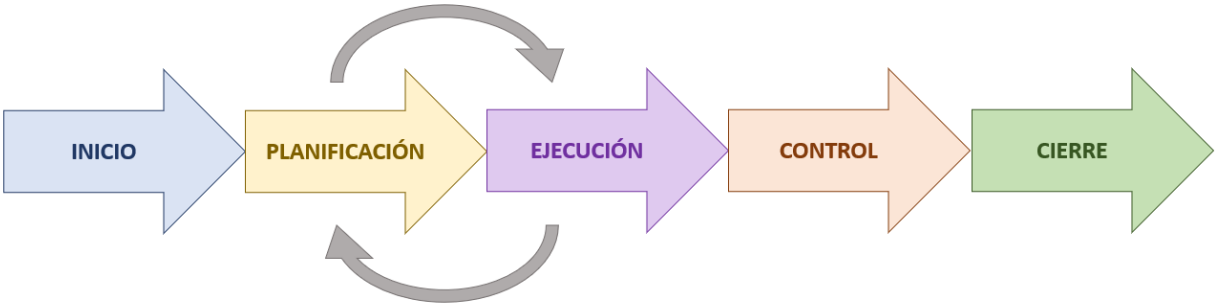

Figura 1-6. Etapas o conjuntos de procesos en el ciclo de vida de un proyecto, versión o fase.

Realización propia

Como se ha indicado, es una posible generalización o idea sobre las etapas que conforman un proyecto. Es evidente que cada proyecto en sí o cada metodología elegida para abordarlo, presenta una serie de particularidades. La metodología de gestión y ejecución del proyecto debe ser seleccionada en la etapa de planificación, pues cada una ofrece unas acciones, hitos y formas de abordar el trabajo.

Se vuelve a hacer énfasis en que es una visión generalizada de las etapas de un proyecto. En este caso se mezcla un modelo de ciclo de vida secuencial con uno iterativo. Los distintos modelos se expondrán en una sección posterior.

En el siguiente punto del documento haremos un recorrido por las metodologías más usadas y popularizadas, realizando una clasificación y distinción por los dos bloques o métodos que las define: metodologías tradicionales o Waterfall, y metodologías ágiles o Agile [10][9][11].

2 METODOLOGÍAS

En este apartado vamos a recopilar y hacer una descripción de las metodologías o marcos de trabajo más popularizados, haciendo una distinción en los dos bloques en los que podemos categorizar éstos. Como se puede intuir, hay un gran número de metodologías, algunas fruto de la fusión entre varias, por lo que intentar plasmarlas todas a un nivel de detalles extremo sería ineficiente y nos desvirtuaría del objetivo del proyecto, que es tener el conocimiento y las herramientas necesarias para saber qué metodología emplear en función de las características del proyecto cuando nos enfrentemos a un proyecto real en el ámbito de las telecomunicaciones.

Evidentemente, para elegir de forma correcta el método, hay que tener un conocimiento suficiente sobre los tipos que hay y soluciones que ofrecen, así que vamos a presentar los métodos, guías y normas estandarizados más populares y utilizados, divididos en dos bloques:

- Metodologías Tradicionales o *Waterfall*: secuenciales, anticipativas y basadas en procesos.
- Metodologías ágiles o *Agile*: iterativas, adaptativas y basadas en personas sobre procesos.

Para dar el detalle correcto de cada solución a la gestión y ejecución de un proyecto, vamos a acudir directamente al organismo y norma reguladora y estandarizada, además de utilizar el material de la asignatura de *Proyectos de Sonido e Imagen*, cursada en el cuarto curso de *GITT*.

2.1 Visión General: ubicación de la metodología en el proyecto

En el último apartado del punto anterior, hicimos una identificación generalizada de las etapas o grupos de procesos que conforman un proyecto, versión o fase de éste. Pero, para poder ubicar correctamente el momento en el que se decide la metodología a emplear, vamos a ascender un poco más para tener una vista aérea del ciclo de vida de un proyecto desde antes incluso de su nacimiento.

Para ello, voy a realizar mi primera aportación personal, a través de un esquema que he elaborado en base a mi experiencia por distintos proyectos en distintas empresas y para distintos clientes, en los que se recogen las etapas desde la vía de entrada u oferta al cliente, hasta la puesta en marcha del producto implementado.

Así que vamos a cambiar de nivel de abstracción, con el objetivo de contextualizar e introducir la metodología o marco de trabajo necesario para abordar un proyecto, anticipándonos al inicio de lo que anteriormente definimos como ciclo de gestión de un proyecto, sea cual sea su naturaleza y método.

Con esta intención, partimos de la *Figura 2-1* que es un esquemático que recoge, a muy alto nivel y de forma simplificada, las diferentes etapas o elementos en la gestión de un proyecto, desde antes incluso de su confirmación e inicio.

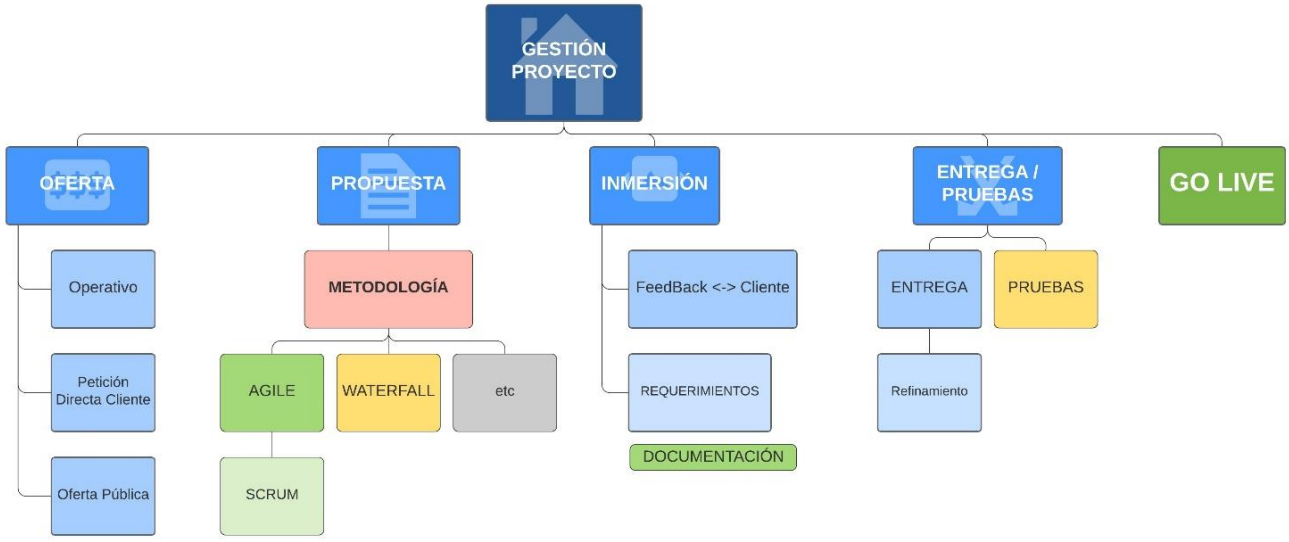

Figura 2-1. Visión personal del ciclo de vida de un proyecto.

Realización propia

- **OFERTA** → Es el canal o medio de acceso al proyecto. Dependiendo de la naturaleza del proyecto, del cliente o entidad que solicita el proyecto, podemos clasificar la oferta como:
  - **Operativo**: conocido también como marketing táctico, es una estrategia para dar a conocer los productos propios de la empresa a posibles compradores. Se basa en la anticipación para cubrir o crear necesidades en el cliente final.
  - **Petición Directa del Cliente**: este tipo de ofertas se realizan directamente del cliente al proveedor. En el ámbito empresarial, muchos clientes trabajan con proveedores fijos, con los que tienen cerrados proyectos de forma temporal (como podría ser una asistencia técnica, aseguramiento de la calidad) o un proyecto que englobe otros sub-bloques o proyectos más pequeños (en este caso, la oferta se estaría haciendo para evolutivos o cambios de alcance).
  - **Oferta Pública**: el cliente lanza una oferta, describiendo las necesidades u objetivo final, con una descripción funcional y técnica. Las empresas proveedoras interesadas pujan, ofreciendo una solución con una estimación a la necesidad planteada.

- **PROPUESTA** → Una vez ganada la oferta, se empiezan a concretar los detalles para el planteamiento del proyecto. La primera y más importante, que supone el objetivo de este estudio o trabajo, es la metodología a seguir. Puesto que ésta influirá en la comunicación proveedor-cliente, en los plazos y las entregas, se debe acordar qué método de trabajo se va a seguir: Agile, Waterfall. Los requerimientos por parte del cliente y los recursos por parte del proveedor serán gestionados de forma más o menos eficiente a través de la metodología elegida.
- **INMERSIÓN** → También conocida como etapa de Análisis, es donde se concreta, detalladamente, las necesidades funcionales y técnicas del cliente. Se trata del momento en el que el proveedor "se sienta" con el cliente para profundizar y crear el detalle de los requerimientos. Aunque se explicará en un punto posterior, para la metodología SCRUM es la etapa clave, pues aquí se define el Product Backlog o conjunto de requerimientos, así como los diferentes Sprints o entregas. La finalidad de la Inmersión es la elaboración de la documentación técnica y funcional del producto a entregar por el proveedor, quedando aprobado por el cliente.
- **EJECUCIÓN/ENTREGA/PRUEBAS** → En esta etapa, el proveedor desarrolla y entrega el producto al cliente, sobre el que se realizan pruebas de calidad y rendimiento para poder dar el visto bueno. Se realizan tanto pruebas por parte del proveedor, como pruebas por el cliente. Es otra etapa clave en la metodología SCRUM, ya que la entrega se llevaría de forma iterativa.
- **GO LIVE** → Una vez realizadas las pruebas y confirmado que el producto cubre todos los requerimientos cumpliendo con la calidad esperada, se entrega y se pone en funcionamiento.

2.2 Ciclos de vida: modelos

Para poder asimilar y tener una idea generalizada de las etapas en un proyecto o fases del mismo, se ha plasmó un modelo de ciclo de vida idealizado en la *Figura 1-6*, que mezcla la secuencialidad con la iteración.

No obstante, hay que aclarar que todos los modelos se basan, o más bien, llevan implícitas las 5 fases descritas anteriormente: inicio o análisis, planificación, ejecución, control y cierre. La diferencia entre estos es la organización y la formar que tienen las fases de interactuar entre ellas, dándole más o menos importancia a una u otra, y agrupándolas y tratándolas como una sola.

En esta sección se recogen las alternativas en cuanto al modelo o diseño de ciclos de vida para mostrar las posibilidades, pero a su vez, la complejidad a la hora de tomar un rumbo y decisión en la gestión de proyecto.

Pese a que nos interesan los modelos secuenciales e iterativos sobre el resto (ya que estos últimos surgen para solventar y optimizar la gestión en ingeniería de software, aunque sean extrapolables a otras áreas), no está de más conocerlos para abrir el abanico de posibilidades y opciones con el fin de afrontar un proyecto de forma eficiente.

La regulación de los diferentes modelos de ciclo de vida recae sobre el estándar internacional ISO/IEC 12207:2008 *Software Life Cycle Processes*. Basándonos en éste, vamos a hacer un breve recorrido sobre los diferentes modelos [12].

2.2.1 Modelo en Cascada

El modelo en cascada es el primer modelo implementado y aplicado, diseñado y propuesto en 1970 por Winston Royce.

De una gran sencillez, se basa en el avance secuencial, de manera que no comienza una fase posterior hasta que no queda completamente cerrada la actual, sin posibilidad de retorno ni iteración.

Podemos decir que su base es una exquisita definición y diseño, que no deje detalles sin recoger ni explicar, pudiendo planificar de forma precisa, pues no se dejan márgenes de incertidumbre ni error.

Este modelo se muestra en la *Figura 2-2*.

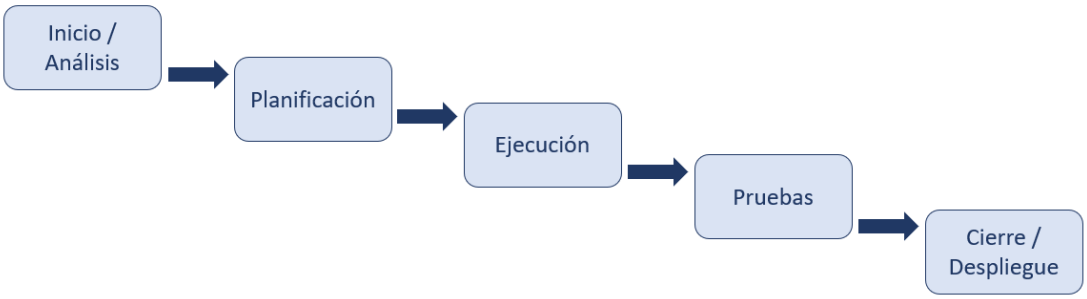

Figura 2-2. Modelo en Cascada.

Realización propia

2.2.2 Modelo Incremental

Aunque entremos en detalles posteriormente, debido a su complejidad y potencial, podemos definirlo como un modelo iterativo y evolutivo, fundamento de las metodologías ágiles, cuyo fin es mostrar un producto mínimo viable o *MVP* al cliente, que se completará y mejorará en las continuas iteraciones con éste.

Combina secuencias lineales en paralelo de forma que, en el tiempo, cada una de estas secuencias produce un incremento del producto, como se recoge en la *Figura 2-3*.

Éste es el principal motivo por el que se reduce el impacto ante modificaciones; es más, se convierte en su punto fuerte, permitiendo la adaptabilidad en todo momento.

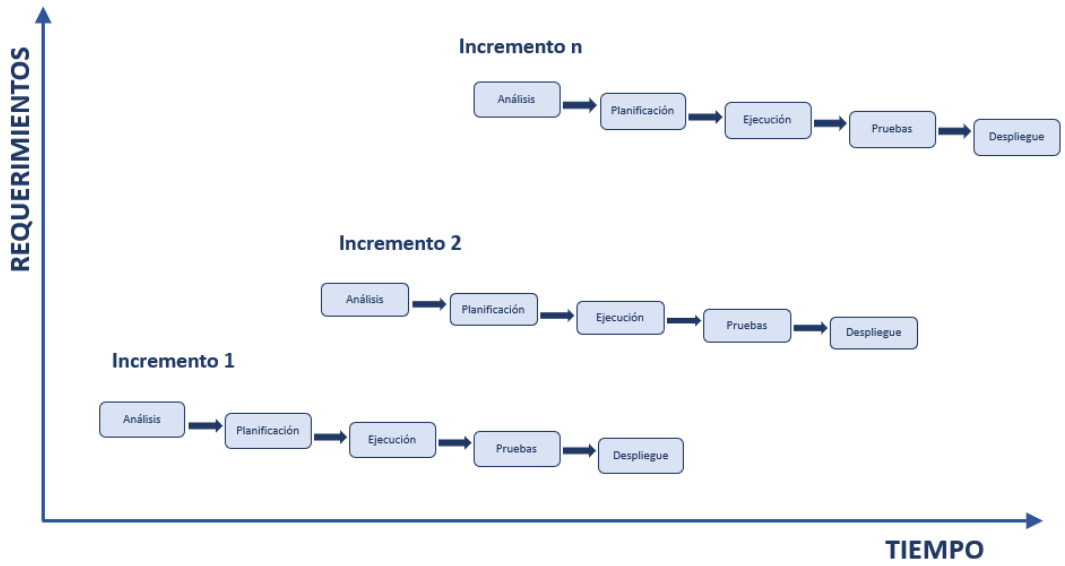

Figura 2-3. Modelo Incremental.

Realización propia

2.2.3 Modelo en V

Parte del modelo en cascada, con una pequeña variación de cara a conseguir una mayor efectividad en la fase de pruebas o control.

Intenta paralelizar la elaboración de los planes de pruebas y el control de calidad con el avance del desarrollo o ejecución del proyecto. Así se consigue darle robustez al desarrollo, intentando asegurar la validación del resultado, que se ha ido planteando en paralelo. El esquema de este modelo se muestra en la *Figura 2-4*.

Prácticamente ofrece los mismos resultados que el modelo en cascada, pero da más importancia a las pruebas y validación pues, como hemos visto, son modelos estáticos que se ven afectados de forma crítica por cualquier cambio o error.

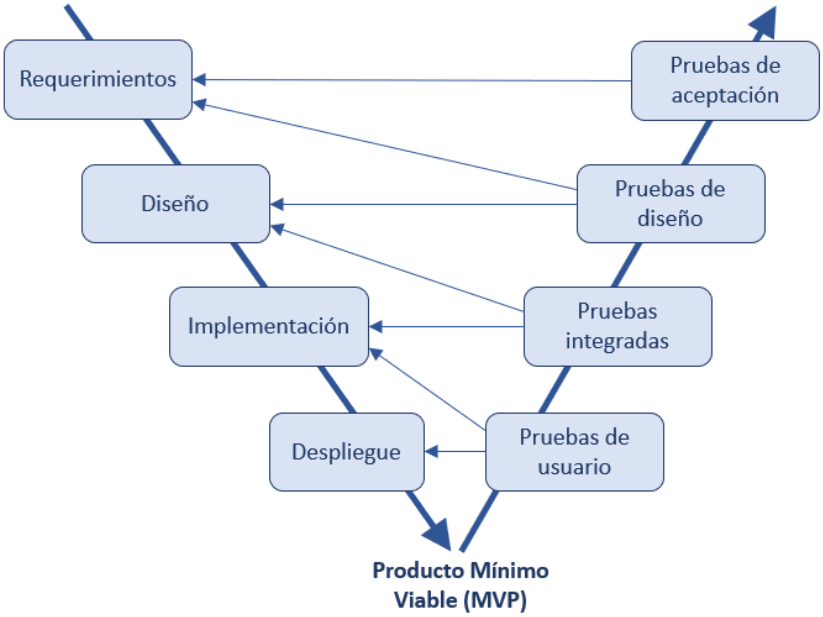

Figura 2-4. Modelo en V.

Realización propia

2.2.4 Prototipo

Modelo evolutivo cuya primera etapa es la comunicación para definir el *prototipo* o producto general, sin funcionalidades ni características concretas detalladas. El objetivo de tener un primer producto elemental o básico es posibilitar una mejor concepción del objetivo del proyecto por ambas partes.

Sirve para trabajar sobre la idea generalizada cuando los requisitos a cubrir no están del todo claros, de forma que, sobre el prototipo o producto básico, se puedan definir en detalle los requerimientos.

Es muy importante el refinamiento del prototipo con la evaluación del cliente, producida en la fase de comunicación. Permite una replanificación y diseño, iniciándose de nuevo el ciclo, como se muestra en la *Figura 2-5*.

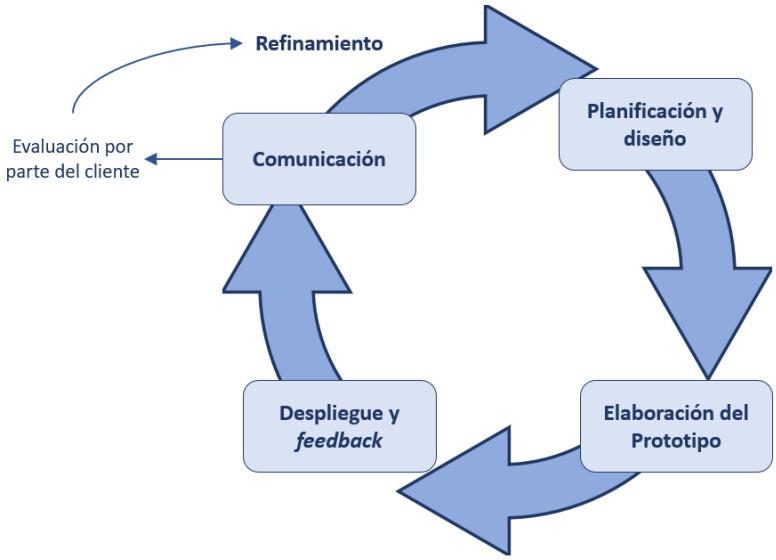

Figura 2-5. Modelo de Prototipo.

Realización propia

2.2.5 Modelo en Espiral

Implementado en 1988 por Barry Boehm, es un modelo que fusiona el prototipo, en cuanto a la capacidad de iteración y dinamismo, y el modelo en cascada, por el acotamiento e importancia en la secuencialidad.

Como modelo evolutivo que es, se basa en la generación de versiones del producto, que dan como resultado un producto cada vez más completo, complejo y adaptado. Cada bucle de la espiral sería una fase, de modo que, a medida que el proyecto avanza, el bucle se va agrandando.

Es una representación muy realista de la evolución de un producto, en la que cada vuelta completa sería una versión dividida en las 5 fases genéricas, que cada vez abarcan más, a medida que el producto se va completando y volviendo más complejo, como puede contemplarse en la *Figura 2-6*. Permite, tanto al desarrollador como al cliente, tener una estimación y anticipación mucho más certera de los riesgos, minimizando el impacto.

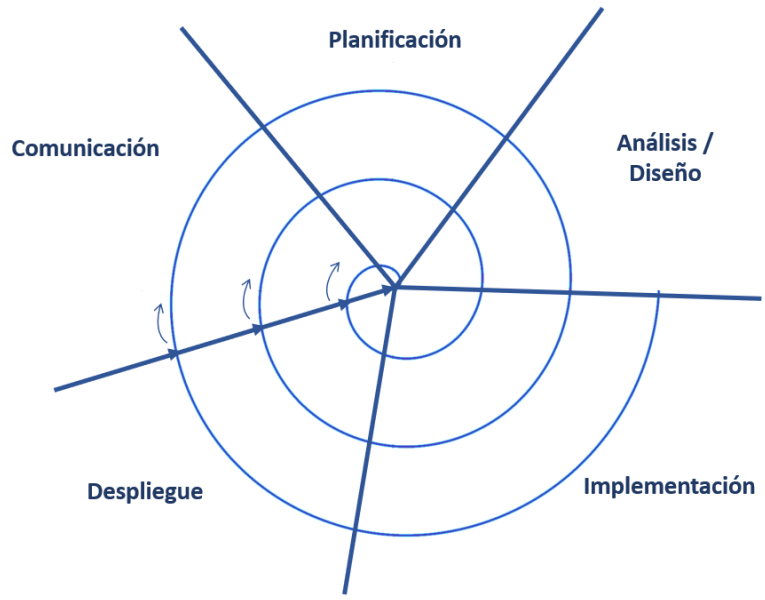

Figura 2-6. Modelo en Espiral.

Realización propia

2.2.6 Modelo Concurrente

Rompiendo con la dinámica anterior, este modelo presenta un funcionamiento por transiciones entre estados, propiciadas por eventos. El estado va moviéndose entre diferentes valores (etapas) hasta que sale, por la única transición posible, al estado "Finalizado", tal y como se ilustra en la *Figura 2-7*.

Cada estado es una actividad, y estará activo sólo cuando se inicie, quedando desactivado cuando se pase a otro estado. Con esta alternativa, siempre se tiene una foto actualizada del estado del proyecto.

Figura 2-7. Modelo Concurrente.

Realización propia

2.3 Metodologías Tradicionales o *Waterfall*

Las Metodologías Tradicionales también se conocen con el término "pesadas", pues se fundamentan en una robusta planificación y estimación para avanzar de forma secuencial en las fases del ciclo de vida del proyecto.

Basadas en la realización de una documentación exhaustiva y en el cumplimiento de la planificación del proyecto, requieren de una primera fase de inmersión y análisis muy trabajada para una buena predicción, que evite cualquier corrección en etapas posteriores, pues supondría un enorme impacto dada su naturaleza "estática" y poco adaptable.

Han sido las metodologías más utilizadas en los últimos treinta años y se centran en un desarrollo secuencial, con una aproximación lineal, que se inicia con las fases de análisis y diseño, y termina con la fase de pruebas (o testing) y despliegue o puesta en producción.

Se le conoce como Waterfall porque cada fase es una etapa o nivel diferenciada del resto y debe cerrarse antes de seguir con la siguiente, como se plasma en la *Figura 2-8*. Es decir, no hay una realimentación, sino que hay que concluir y cerrar cada etapa completamente para avanzar a la siguiente.

Por este motivo, por ejemplo, la fase de análisis debe hacerse con mucha cautela, sin saltarse detalles, para hacer una predicción lo más certera posible. Hasta que no se tiene el análisis cerrado por completo, con los requisitos aprobados por el cliente, no se empezaría con el diseño.

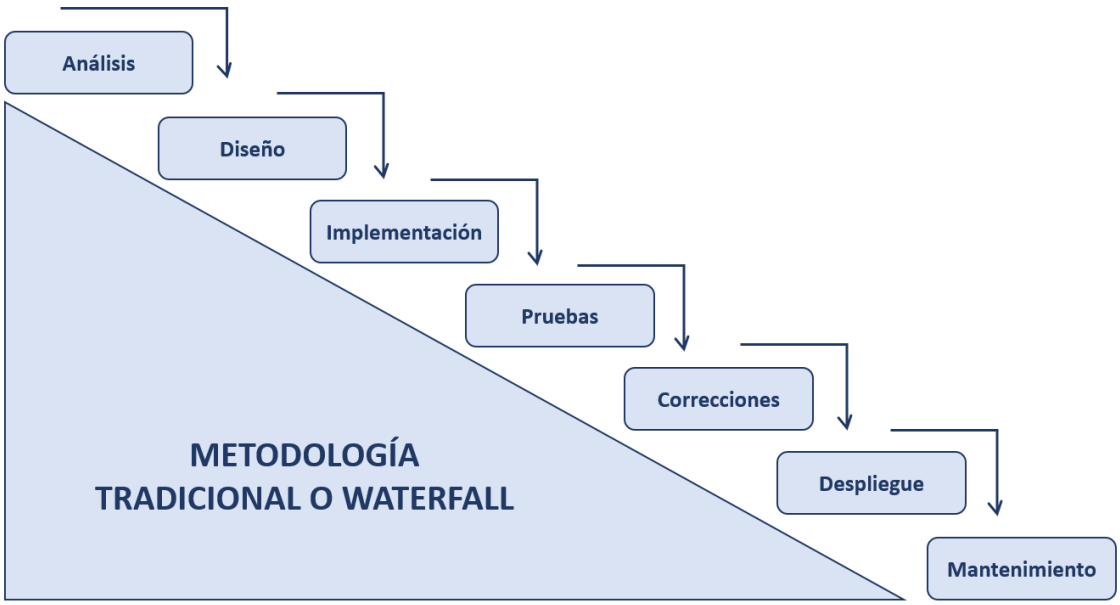

Figura 2-8. Cascada en Metodología Tradicional.

Realización propia

Aunque ya vimos en una sección anterior las etapas o fases de un proyecto de forma generalizada, vamos a concretar cuáles son las más comunes en el ciclo de vida de un proyecto basado en una metodología Tradicional:

1. Análisis → Recogida de requisitos y elaboración de la documentación
2. Diseño → Propuesta de solución a los requisitos documentados
3. Implementación → Desarrollo del diseño elaborado
4. Pruebas → Fase de pruebas del producto ya implementado. Según las exigencias y el tipo de producto, hay varios tipos de pruebas:

   a. Integradas: las realiza internamente el proveedor o proveedores que han desarrollado el producto
   b. UAT's o de Usuario: las realiza el cliente para dar por válido el producto, comprobando todas las funcionalidades solicitadas
   c. De Rendimiento: las realiza el proveedor o el equipo técnico del cliente para verificar la eficiencia del producto
5. Correcciones → Tras las pruebas realizadas, se corrigen o solucionan los errores detectados
6. Despliegue → Instalación o paso a producción (PaP) del producto. En esta fase, además, se suelen hacer pruebas de regresión si el proyecto ha sido un evolutivo o mejora sobre un producto ya existente, verificando que todo lo anterior al despliegue sigue funcionando correctamente.
7. Mantenimiento → Servicio para solventar futuros errores, averías o dudas que tenga el usuario sobre el producto.

A continuación, realizaremos un recorrido por las Metodologías Tradicionales más conocidas y empleadas, acudiendo directamente a su norma y usando el material impartido en la asignatura correspondiente del GITT.

Las metodologías, guías o normas que se van a describir son: PMBOK, PRINCE2, ICB, SWEBOK SSADM, MÉTRICA y MERISE.

2.3.1 PMBOK

PMBOK son las siglas de la guía *Project Management Body Of Knowledge*, que es un estándar de gestión de proyectos del PMI (Project Management Institute) y el único que está acreditado por ANSI (American Standards Institute). Actualmente, se usa la quinta y sexta versión de la guía, siendo en 1996 cuando se publicó la primera. Además de la guía, PMBOK tiene tres extensiones que complementan a ésta:

- **Construction Extension**: su última versión fue lanzada en 2016, y proporciona una guía específica de las áreas de conocimiento de PMBOK, así como orientaciones en otras áreas adicionales:
  - Recursos del proyecto (no sólo recursos humanos)
  - Gestión de salud, seguridad, protección y medio ambiente
  - Gestión financiera (aparte del costo)
  - Gestión de siniestros o fallos

- **Government Extension**: actualmente disponible en su tercera edición, publicada en 2006, amplía la información referente a los procesos de gobernanza de proyectos en sectores públicos. Define los términos clave, describe los contextos y atmósferas en los que se ubican los proyectos gubernamentales, y revisa la gestión del ciclo de vida de los programas gubernamentales.
- **Software Extension**: su edición más reciente, la quinta, se liberó en 2013, y se trata de una ampliación y puntualización de prácticas para proyectos de software.

Como todos los marcos de trabajo en la gestión de un proyecto, suministra unas pautas estructuradas que se pueden aplicar a cualquier tipo de proyecto.

Este método se basa en definir el trabajo por paquetes, llevados a cabo en procesos que se superponen e interactúan en las diferentes fases del proyecto, describiéndose o clasificándolos en base a los siguientes bloques:

- Entradas → documentación, diseños, planos, esquemas, …
- Herramientas y técnicas → transformaciones u operaciones que se aplican a las entradas para obtener las salidas.

- Salidas → documentación, diseños, planos, esquemas, … (entradas transformadas).

Estos procesos los podemos visualizar en la *Figura 2-9*.

Figura 2-9. Bloques E/S en PMBOK.

Realización propia

La guía recoge un total de 47 procesos, divididos en 5 procesos básicos o macroprocesos, y 10 áreas de conocimiento, siendo las siguientes:

A. Procesos básicos:

1. Inicio: para definición de un nuevo proyecto o fase del mismo. Incluye 2 procesos.
2. Planificación: para acotar el alcance, detallar objetivos y proyectar la estrategia de la definición en el inicio, Incluye 23 procesos.
3. Ejecución: para realizar el trabajo definido en la planificación. Incluye 9 procesos.
4. Monitorización y control: para revisión y control en el seguimiento de la ejecución. Incluye 11 procesos.
5. Cierre: para cerrar o finalizar las tareas de todos los procesos. Incluye 2 procesos.

B. Áreas de conocimiento (definidas en el primer capítulo del TFG, de cara a explicar los factores a tener en cuenta para la gestión de un proyecto):

1. Gestión de la Integración: relacionada íntimamente con la dirección de proyectos, establece los criterios para la gestión, coordinación y administración de todos los procesos.
2. Gestión del Alcance: para acotar los procesos y actividades del proyecto, definiendo y determinando el alcance de éste.
3. Gestión de Tiempo: asignación y gestión del tiempo dedicado a la ejecución de cada proceso del proyecto para cumplir con los plazos.
4. Gestión de Costes: control de los costes para cumplir con el presupuesto acordado.
5. Gestión de la Calidad: para garantizar la calidad del resultado, determina las responsabilidades de las actividades y procesos, y establece las políticas y evaluaciones del resultado.
6. Gestión de Recursos Humanos: dirección y coordinación del o de los equipos de personas implicados.
7. Gestión de la Comunicación: administración y gestión de la comunicación entre los implicados, ya sea de forma interna como externa, garantizando siempre una vía y estrategias de comunicación.
8. Gestión de Riesgos: detección y solución de los riesgos en cada proceso.
9. Gestión de Adquisiciones: control y gestión de gastos derivados de compras de herramientas, subcontrataciones o servicios externos.

10. Gestión de Interesados o Stakeholders: correcta administración de los intereses y expectativas del proyecto, así como posibilidades de intervenciones de terceros.

Conocidas las áreas y los conjuntos de procesos o macroprocesos, para aplicar esta guía o método es necesario organizar las siguientes ideas:

- La agrupación de procesos o macroprocesos determinarán los procesos a llevar a cabo por cada rol o perfil. De esta forma, se distribuye el trabajo y las responsabilidades, asociando cada bloque (llevado a cabo por una persona con un rol concreto) con unos procesos concretos. Pueden identificarse, asociándolos a una acción o verbo:
  - Inicio → Empezar
  - Planificación → Pensar
  - Ejecución → Dirigir
  - Monitorización → Observar
  - Cierre → Terminar
- Las áreas de conocimiento se centran en el tipo de trabajo que hay que realizar. Son la especialización, dentro de cada macroproceso, que tendrían que llevarse a cabo y gestionar por el rol encargado del mismo.

Para entender y visualizar la estructura de este método basado en la convergencia de macroprocesos y áreas de conocimiento, se muestra la *Tabla 2-1* con los procesos de PMBOK y la figura *Figura 2-10* con las interacciones entre los grupos de procesos, tomada directamente de la guía [1]:

MACROPROCESOS		Inicio	Planificación	Ejecución	Monitoreo y Control	Cierre
ÁREAS DE CONOCIMIENTO	**Integración (6)**	4.1 Desarrollar el acta de constitución del proyecto	4.2 Desarrollar el plan para la dirección del proyecto	4.3 Dirigir y gestionar el trabajo del proyecto	4.4 Monitorear y controlar el trabajo del proyecto 4.5 Realizar el control integrado de cambios	4.6 Cerrar el proyecto o fase
	Alcance (6)		5.1 Planificar la gestión de alcance 5.2 Recopilar requisitos 5.3 Definir alcance 5.4 Crear la EDT		5.5 Validar alcance 5.6 Controlar el alcance	
	Cronograma (6)		6.1 Planificar la gestión del cronograma 6.2 Definir las actividades 6.3 Secuenciar las actividades 6.4 Estimar la duración de las actividades 6.5 Desarrollar el cronograma		6.6 Controlar el cronograma	

	Costos (4)		7.1 Planificar la gestión de los costos 7.2 Estimar los costos 7.3 Determinar el presupuesto		7.4 Controlar los costos	
	Calidad (3)		8.1 Planificar la gestión de la calidad	8.2 Gestionar la Calidad	8.3 Controlar calidad	
	Recursos (5)		9.1 Planificar la gestión de recursos	9.2 Adquirir los recursos 9.3 Desarrollar el equipo 9.4 Dirigir al equipo	9.5 Controlar los recursos	
	Comunicaciones (3)		10.1 Planificar la gestión de las comunicaciones	10.2 Gestionar las comunicaciones	10.3 Monitorear las comunicaciones	
	Riesgos (6)		11.1 Planificar la gestión de riesgos 11.2 Identificar los riesgos 11.3 Realizar el análisis cualitativo de riesgos 11.4 Realizar el análisis cuantitativo de riesgos 11.5 Planificar la respuesta a los riesgos	11.6 Implementar la respuesta a los riesgos	11.7 Controlar los riesgos	
	Adquisiciones (4)		12.1 Planificar la gestión de las adquisiciones	12.2 Efectuar las adquisiciones	12.3 Controlar las adquisiciones	12.4 Cerrar procedimientos
	Interesados (4)	13.1 Identificar a los interesados	13.2 Planificar el involucramiento de los interesados	13.3 Gestionar la participación de los interesados	13.4 Monitorear el involucramiento de los interesados	

Tabla 2-1. Procesos de PMBOK.

Realización propia, basada en la Guía PMBOK (PMI, 2017)

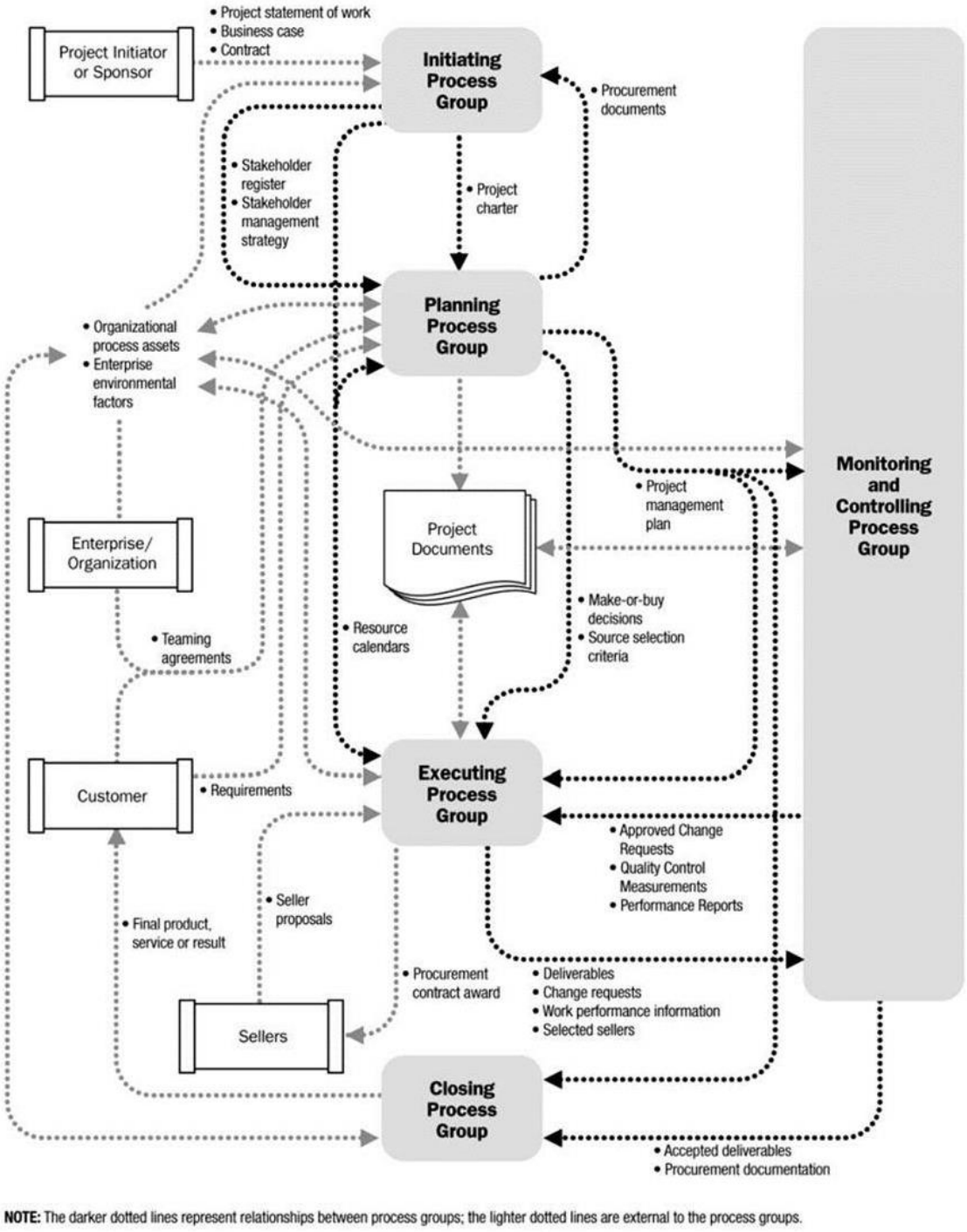

Figura 2-10. Grupos de procesos en PMBOK e interacciones.

Guía PMBOK (PMI, 2017)

2.3.2 PRINCE2

PRINCE2, como se puede intuir, es una metodología que deriva de otra anterior, PRINCE, que fue desarrollada por la CCTA (Central Computer and Telecommunications Agency) en 1989 en Reino Unido, como un estándar para la gestión de proyectos en sistemas IT [13]. Está íntimamente ligada a los proyectos de Telecomunicaciones.

El acrónimo PRINCE viene de "Proyectos en ambientes controlados", aunque también se aplicó fuera del marco TIC.

PRINCE2 nace en el año 1996, ya como método genérico (no específico de TIC), consiguiendo una inmensa popularidad en la gestión de proyectos. La última versión se revisó en el año 2017.

Es una metodología estructurada que se basa en procesos, cubriendo todos los aspectos en organización, gestión y control de los proyectos. Su potencial es garantizar el cumplimiento de plazos, planificación de forma viable y rentable. Para ello, se define un ciclo de vida de forma clara, basado en principios, grupos de procesos y áreas de gestión.

La definición de tareas a cubrir por los diferentes roles, en cada etapa del ciclo de vida del proyecto, se basa en 7 principios y se estructura en 8 grupos de procesos y 8 áreas de gestión.

A. Principios:

1. Justificación comercial continua: justificación del motivo para realizar el proyecto y su estabilidad.
2. Aprendizaje de la experiencia: lecciones y experiencias de proyectos anteriores.
3. Roles y Responsabilidades: definición de involucrados para tomar decisiones.
4. Gestión por Fases: una fase comienza cuando se cierra la anterior.
5. Gestión por Excepción: en función de las responsabilidades definidas, se escala la consulta a roles superiores.
6. Orientación a Productos: prioridad al producto sobre tareas o procesos.
7. Adaptación: aplicación correcta en función de volumen, dificultad, riesgo y capacidad del proyecto.

B. Grupos de procesos:

1. Emprender Proyecto (SU): para preparar el inicio del proyecto, en función de su gestión, control y viabilidad, garantizando los recursos.
2. Dirigir Proyecto (DP): para definición de responsabilidades y la dirección del proyecto, siendo el proceso que está por encima de todos, presente en cada inicio y cierre de etapa.
3. Iniciar Proyecto (IP): para documentar, antes de empezar, la gestión del proyecto entero.
4. Planificación (PL): para identificación de actividades, recursos y entregables durante el proyecto.
5. Control de Etapa (CS): para gestionar diariamente el proyecto, realizándose en cada etapa.
6. Gestión del Producto (MP): para acordar el trabajo a realizar.
7. Gestión de los límites de Etapa (SB): para garantizar el cierre de una etapa para comenzar con la siguiente.
8. Cierre del Proyecto (CP): para cerrar el proyecto, ya sea por finalización satisfactoria, prematura o cancelación, tomando nota de la experiencia o lecciones aprendidas.

C. Áreas de Gestión (definidas con preguntas):

1. Caso de Negocio: *¿Por qué quieres hacer …?* Describe la viabilidad de la idea propuesta, manteniendo la atención en los objetivos y beneficios.
2. Organización: *¿Quién se encarga de …?* Define las responsabilidades y la función del equipo.

3. Calidad: *¿Qué hay que hacer?* Aborda los requerimientos de calidad, basados en las descripciones y detalles del producto.
4. Planes: *¿Cómo, cuánto y dónde?* Describe los pasos a realizar, así como las técnicas a aplicar y la comunicación del equipo e interesados.
5. Controles: *¿Se ajusta a la planificación?* Describe los criterios de aceptación y la planificación en base a los recursos y costes, garantizando la viabilidad.
6. Riesgos: *¿Qué ocurriría si …?* Aborda la gestión de incertidumbres y riesgos a asumir.
7. Cambio: *¿Cuál es el impacto?* Determine el impacto y el cálculo debido al control de cambios de alcance.
8. Progreso: *¿Dónde estamos?, ¿a dónde vamos?, ¿debemos continuar?* Aborda el proceso de toma de decisiones de cara a realizar aprobaciones para valorar la continuación o no del proyecto.

Por último, para plasmar este modelo y ver cómo interactúan los procesos en las etapas, se muestra la *Figura 2-11*:

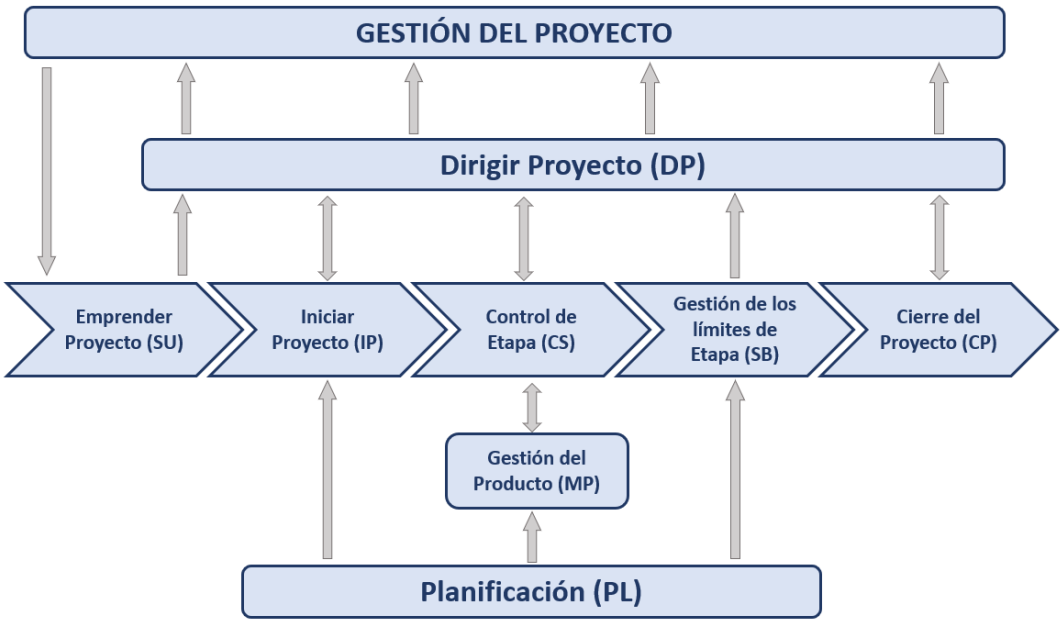

Figura 2-11. Grupos de procesos en PRINCE2 e interacciones.

Realización propia (basada en la sexta edición de PRINCE2, 2017)

2.3.3 ICB

ICB es un estándar internacional para la competencia en dirección de proyectos [14], acrónimo de *IPMA Competence Basline*, cuya primera versión fue publicada en el año 1995 en Suiza por la organización IPMA[2]. Esta organización tiene representación en España, llamada AEIPRO, que se autodefine como una asociación sin ánimo de lucro con el objetivo de promover la excelencia en la profesión de la Dirección e Ingeniería de Proyectos.

ICB se centra en la dirección de proyectos, detallando las competencias y funciones necesarias para una buena gestión. Es una metodología muy útil de cara a los profesionales e interesados o *stakeholders*

La versión inicial de ICB se publicó en 1995, y en ella se definen y validan las competencias en la dirección de proyectos. La siguiente versión, ICB 2.0, fue publicada en 1999; en 2006 se volvió a renovar con la llamada ICB 3.0 y la versión actual, ICB 4.0 se publicó en 2015.

La estructura con respecto a la certificación y reparto de conocimientos para el rol de dirección de proyectos se reparte en un modelo progresivo de cuatro niveles, que son, de menos avanzado a más:

- IPMA nivel D: técnico en dirección de proyectos.

  Este nivel acredita y dota de capacidad suficiente para dirigir proyectos complejos y portafolios.

- IPMA nivel C: profesional en dirección de proyectos.

  Capacidad para dirigir proyectos complejos, requiriéndose una experiencia mínima de 5 años en esta área.

- IPMA nivel B: director de proyecto.

  Nivel que dota de capacidad para dirigir proyectos de cierta complejidad y especialización. Requiere una experiencia mínima de 3 años.

- IPMA nivel A: director de cartera de proyectos.

  Este nivel certifica la capacidad para aplicar todos los conocimientos de la dirección de proyectos.

El objetivo de ICB es hacer una estandarización y una reducción de las tareas necesarias y fundamentales para cerrar un proyecto de la manera más efectiva y eficiente. Para ello, contiene y describe los términos, habilidades, tareas, procesos, funciones, técnicas, herramientas y métodos a usar, de forma teórica y práctica.

A diferencia de las metodologías anteriormente descritas, ICB se estructura por competencias en vez de procesos [15]. Considera un total de 46 elementos de competencia, que se agrupan en los siguientes 3 grandes grupos:

A. **Competencias técnicas**: relacionadas con todo lo referente al cumplimiento de los requisitos del proyecto en función de los interesados o involucrados, así como la integración de las tareas de la organización y la producción de entregables en la organización del proyecto.

   En este bloque se recogen 20 competencias:

   1. Éxito en la dirección de proyectos
   2. Partes involucradas
   3. Requisitos y objetivos
   4. Riesgos y oportunidades

[2] *IPMA* son las siglas de *Inernacional Project Management Association*, organización fundada en Suiza en 1965, siendo la más antigua en el ámbito de gestión de proyectos y formada por una red de asociaciones nacionales. La asociación española suscrita al convenio de colaboración con el PMI es la *Asociación Española de Ingeniería de Proyectos* o *AEIPRO*, creada en 1992.

5. Calidad
6. Organizaciones
7. Trabajo en equipo
8. Resolución de problemas
9. Estructuras
10. Alcance y entregables
11. Tiempo y fases
12. Recursos
13. Coste y financiación
14. Aprovisionamiento y contratos
15. Cambios
16. Controles e informes
17. Información y documentación
18. Comunicación
19. Arranque
20. Cierre

B. **Competencias de comportamiento**: abarcan y cubren todo lo relacionado a las actitudes y destrezas del gerente del proyecto. Este bloque recoge todas las competencias íntimamente ligadas al director del proyecto, la gestión de éste, gestión global (interna y con el resto de involucrados o interesados) y aquellas asociadas a la economía, sociedad, cultura y la historia.

   En este bloque se recogen 15 competencias:

1. Implicación
2. Autocontrol
3. Asertividad
4. Relajación
5. Accesibilidad
6. Creatividad
7. Liderazgo
8. Orientación al resultado
9. Eficiencia
10. Consultable
11. Negociación
12. Crisis y conflictos
13. Credibilidad
14. Apreciación de valores
15. Ética

C. **Competencias contextuales**: describen todos los elementos referentes al contexto del proyecto. Están relacionadas con la gestión de proyectos en organizaciones permanentes y las interrelaciones con la administración de negocios.

Son 11 las competencias de este bloque:

1. Orientación al programa
2. Orientación al portfolio
3. Implementación de proyecto
4. Programa y portfolio
5. Legalidad
6. Negocio
7. Organización permanente
8. Sistemas, productos y tecnología
9. Gestión de personal
10. Salud, seguridad, prevención y entorno
11. Financiación

En la *Figura 2-12* se recogen todas las competencias, agrupadas en sus grupos correspondientes.

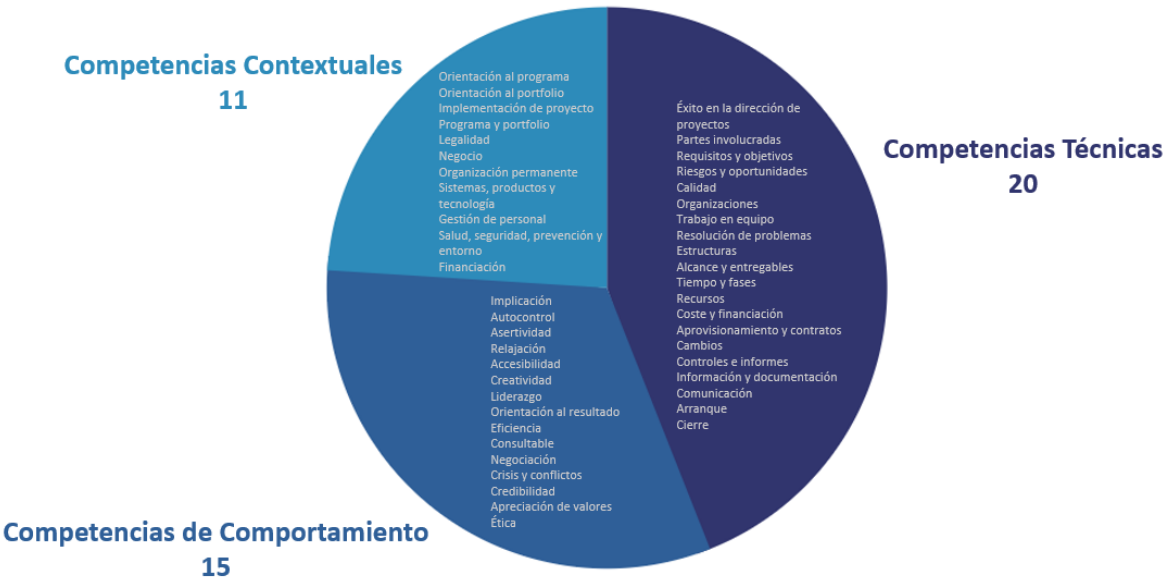

Figura 2-12. Estructura de competencias en ICB.

Realización propia (basada en IPMA, 2015)

2.3.4 SWEBOK

SWEBOK es un estándar internacional [16], en concreto el *ISO/IEC TR 19759:2005*, cuya tercera y última versión fue publicada en 2014. La primera versión se lanzó en 2001, y fue el resultado de la colaboración de 150 personas de 33 países distintos. Tres años después, en 2005, se publicó la segunda versión.

Implementado y publicado por la *IEEE Computer Society*, siglas de *Institue of Electrical and Electronics Engineers*, son las siglas de *Software Engineering Body of Knowledge*.

Para entender y contextualizar esta guía, es interesante hacer mención de varios hitos relacionados con la *IEEE Computer Society*, que hicieron posible la redacción del estándar *SWEBOK*:

- 1976 → fundación del comité para el desarrollo de normas de Ingeniería de Software por la *IEEE Computer Society*.
- 1979 → publicación del estándar *IEEE-730* con la norma para el aseguramiento de la calidad, siendo la base para las siguientes normas enfocadas en gestión, requisitos, diseños y fase de pruebas.
- 1986 → publicación del estudio *Taxonomy of Software Engineering Standards* (IEEE Std. 1002), que detalla la forma y el contenido de los estándares en Ingería de Software, en lo que a taxonomía se refiere. Se entiende por taxonomía los criterios, principios, métodos y fines para realizar una clasificación e identificación de estándares y metodologías en función de las características del proyecto.
- 1993 → creación del comité responsable de la redacción de la primera versión de SWEBOK, el "Comité para el establecimiento de los conjuntos de criterios y normas para la práctica profesional de la Ingeniería de Software, en la que puedan estar basados las decisiones industriales, la certificación profesional y los programas de estudio". Este organismo nació como la unión de la *IEEE Computer Society* con la *ACM* o *Association for Computing Machinery*.
- 1996 → publicación de uno de los pilares básicos de SWEBOK, el *Standard for Software Life Cycle Processes* (*ISO/IEC 12207*), que recogía los distintos ciclos de vida estandarizados para la gestión de proyectos de software.

Partiendo de su propio nombre, SWEBOK es un "cuerpo de conocimiento", es decir, un conjunto de conceptos, términos y actividades para abarcar un dominio profesional, estandarizado y respaldado por la sociedad científica y organismos pertinentes. No obstante, no sólo se limita y dirige a la Ingeniería de Software, sino que proporciona una metodología para la gestión de proyectos de forma general.

Puesto que el enfoque inicial del estándar es la gestión de un proyecto de software, se van a nombrar y definir las 15 áreas de conocimiento en las que se estructura SWEBOK en el terreno de la Ingeniería de Software. Para su entendimiento y aplicación generalizada, sólo habría que sustituir software por el tipo de producto a implementar.

Las 15 áreas de conocimiento son:

1. **Requisitos**: son las necesidades y la limitaciones acotadas y definidas para ofrecer una solución (producto). Esta área agrupa los procesos y tareas de obtención, análisis, especificación y validación de los requisitos. Además, va más allá, abarcando la gestión de los requisitos durante todo el ciclo de vida del proyecto.
2. **Diseño**: proceso de análisis de los requisitos acordados con el objetivo de ofrecer el detalle y descripción de la estructura y funcionalidades del proyecto para su correcta implementación. Es decir, es el área que abarca el proceso de diseño para el producto resultante.
3. **Construcción o desarrollo**: creación del producto, partiendo del diseño realizado. Cubre la gestión del desarrollo, tecnologías a aplicar, consideraciones y herramientas, así como las pruebas internas para dar

por cerrada la implementación y avanzar en el ciclo de vida.

4. **Pruebas**: evaluación de la calidad del resultado para darlo por bueno o mejorarlo. Para ello es necesaria la elaboración de un plan de pruebas, acordado entre los interesados, y que se realizará por parte del usuario para su validación. Esta área recoge los fundamentos de técnicas, pruebas y consideraciones prácticas para la evaluación del producto.
5. **Mantenimiento**: recoge los fundamentos para mantenimiento del producto, gestión, técnicas, estimación de costes y métricas, así como las técnicas y herramientas de recuperación y reducción de impactos ante errores. El mantenimiento es la fase en la que se solventan posibles errores cuando el producto está en funcionamiento, y en la que se añaden mejoras.
6. **Gestión de la configuración**: auditoría y control en la gestión de versiones del producto, donde se hace una monitorización y trazabilidad con pruebas temporales para identificar cambios en la configuración. El objetivo de las técnicas de esta área es el control de cambios y el seguimiento e integridad del producto.
7. **Gestión de la Ingeniería**: es el área que más nos interesa, pues abarca el inicio, alcance del proyecto, planificación, ejecución, aceptación, revisión y cierre. Desde un plano superior, este bloque controla la gestión del proyecto desde su inicio hasta su cierre, realizando la planificación y control de forma sistemática, disciplinada y cuantificada.
8. **Procesos**: todo lo relacionado a la definición y gestión de los procesos del ciclo de vida del proyecto. En concreto cubre la implementación y cambios de procesos, su definición (ciclos de vida), métodos de evaluación y medición, y herramientas.
9. **Modelos y métodos**: aseguramiento del éxito, de forma sistemática y repetible, ofreciendo una estructura de modelado a partir de las condiciones, análisis de integridad, consistencia, exactitud, calidad y trazabilidad, y métodos para el desarrollo.
10. **Calidad**: fundamentos de la calidad, procesos para gestionar, controlar, verificar y validar el producto, y consideraciones prácticas. Se pueden destacar como procesos de gestión de calidad las revisiones y auditorías, y el análisis de costes y mejoras a partir de la evaluación del resultado.
11. **Práctica Profesional**: asuntos de profesionalidad para garantizar el conocimiento, actitudes y habilidades de los trabajadores, garantizando profesionalidad, responsabilidad y ética. Por ello, se incluyen en este bloque las revisiones de contratos, conductas, dinámicas de grupo y códigos éticos, interactuación y las habilidades de comunicación.
12. **Economía**: alineación y sincronización de las decisiones técnicas dentro del proyecto con los objetivos empresariales. Involucra las actividades que impactan directamente en el recurso económico, como la toma de decisiones, propuestas, planificación, análisis y estimación de costes y beneficios, riesgos e incertidumbre.
13. **Fundamentos de Computación** (o Desarrollo): base de conocimiento en la ciencia del desarrollo e implementación específica para poder resolver problemas y algoritmos complejos, programar o configurar, diseñar canales de comunicaciones, redes, infraestructuras, …
14. **Fundamentos Matemáticos**: obviamente, se necesita garantizar una base matemática para desempeñar una tarea en un proyecto de ingeniería. Aporta, además, robustez funcional, posibilitando y asegurando el análisis con probabilidades, máquinas de estados, lógicas y funciones, y demostraciones y gráficos.
15. **Fundamentos de la Ingeniería**: base concreta de la ingeniería que cubre la necesidad del proyecto. Agrupa los métodos empíricos y las técnicas experimentales, diseños y análisis, simulaciones y modelado, así como estudio estadístico.

Una vez citadas y definidas las 15 áreas de conocimiento, se puede hacer una diferenciación a distintos niveles para considerar cuáles formarían parte del ciclo de vida secuencial del proyecto, y cuáles estarían en un nivel de abstracción superior, asegurando y controlando los factores que impactan e intervienen en las anteriores.

Con esta idea, podríamos considerar elementos del ciclo de vida de un proyecto basado en SWEBOK: requisitos, diseño, construcción o desarrollo, pruebas, mantenimiento, gestión de la configuración y gestión de

la ingeniería.

Concretando para Ingeniería de Software (detalle que ofrece SWEBOK), podríamos recoger los elementos principales de manera esquemática, tal y como se muestra en la *Figura 2-13*.

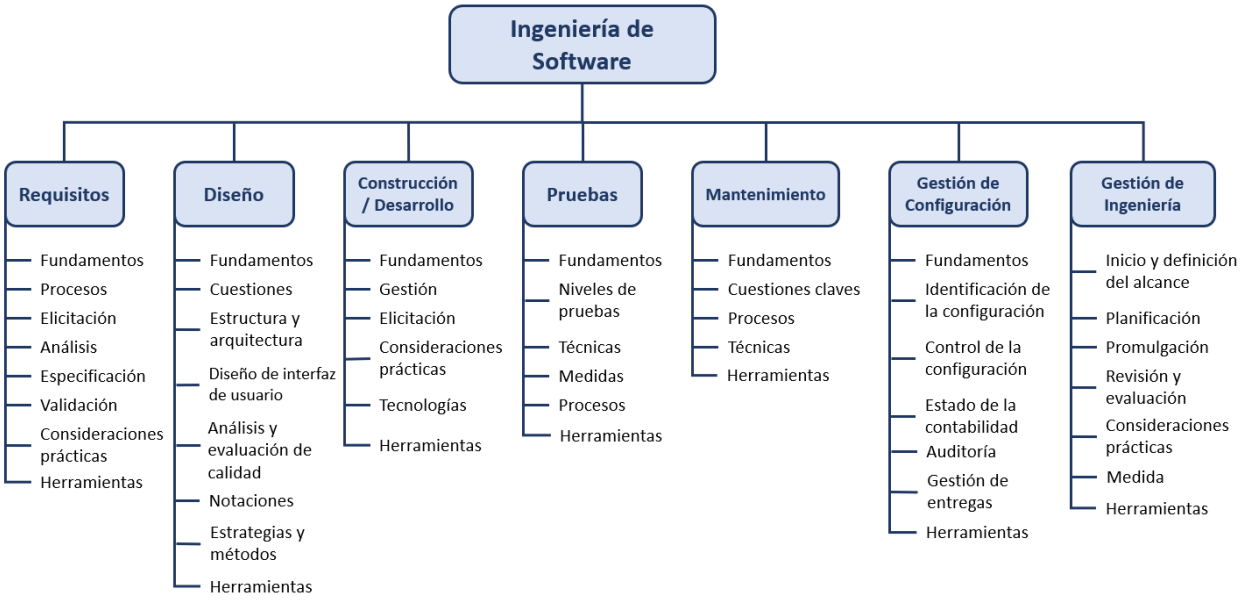

Figura 2-13. Elementos en el ciclo de vida de un proyecto de Ing de Software, basado en SWEBOK.

Realización propia (basada en SWEBOK V3.0 de IEEE, 2014)

El resto de áreas recaen sobre el nivel de control de las fases, garantizando la correcta coordinación entre ellas: procesos (gestión de las diferentes etapas), modelos y métodos (metodología para diseño, desarrollo, pruebas y mantenimiento), calidad (buenas prácticas para ofrecer un resultado evaluado y de calidad), economía (consideraciones para la rentabilidad del proyecto), prácticas profesionales y fundamentos (bases de conocimientos del equipo involucrado en el proyecto, de manera que se aseguren las habilidades, conocimientos y actitudes para llevar a cabo el proyecto).

Es decir, son necesarias y van implícitas en todas las fases del proyecto para la correcta gestión del mismo y la efectividad en el desempeño de cada elemento del ciclo de vida.

2.3.5 SSADM

SSADM son las siglas de la metodología pública *Structured Systems Analysis and Design Method* [17], desarrollada por LBMS (Learmoth y Burchet Management Systems), continuada por *CCTA* (Central Computing and Telecommunications Agency) y publicada por primera vez en 1981 por el gobierno de Reino Unido.

En 1983 SSADM fue considerada de uso obligatorio por el gobierno británico para llevar a cabo el desarrollo de la totalidad de proyectos del gobierno. Cinco años más tarde, en 1988, se lanzó y promocionó como estándar abierto y, en el año 2000, la agencia que la continuó, *CCTA*, la renombró como *Business System Development*, reorganizándola en 15 módulos, además de añadir 6 nuevos. En este momento, sigue siendo la metodología estándar para desarrollo de proyectos en el gobierno de Reino Unido.

Se trata de una metodología de aproximación en cascada para su uso en el desarrollo de proyectos de IT, siendo considerada la más completa de las metodologías estructuradas por su riguroso enfoque en la documentación para el diseño y por la garantía comprobada y experiencia de sus usuarios durante todos estos años. Podríamos considerarla como la más opuesto o antónima a las metodologías ágiles, pues su objetivo y

marco de trabajo es el exquisito nivel de detalles y trabajo sobre la documentación, planificación y diseño, de cara a dar una gran efectividad en su ejecución, completamente estática, nada iterativa ni abierta.

Es una metodología "de aproximación" porque consiste en una arquitectura de tres capas o esquemas, a distinto nivel. Va de un alto nivel o capa superior a capas inferiores, hasta dos niveles más bajos, en los cuales se va haciendo una descomposición y una especificación más detallada de los requerimientos del negocio, como se puede ver en la *Figura 2-14*.

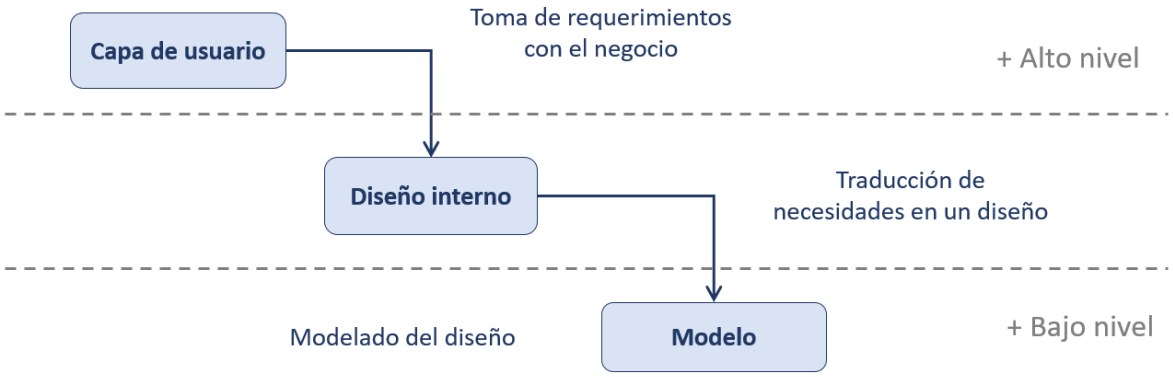

Figura 2-14. Aproximación en 3 capas de SSADM.

Realización propia

El foco y consideración de SSADM son las estructuras de datos sobre los procesos, pudiendo tipificarla como una metodología orientada a los datos.

Para ello, se apoya en tres bloques fundamentales, llamados "vistas":

1. **Modelo lógico de datos**: con la finalidad de cerrar la estructura lógica de datos o LDS con su documentación, es el proceso para la identificación, modelado y documentación de los requerimientos.

   El resultado es la obtención de diagramas de datos lógicos, siendo las dos técnicas más empleadas las siguientes:

   a. **LDS** (estructura lógica de datos) → modelo entidad/relación, que parte de un diagrama simple que se va completando y afinando a medida que se van conociendo los detalles de las relaciones entre entidades. Esta técnica es conocida como modelo de CHEN.

   b. **Diagramas E/R**: diagramas de entidad/relación empleados para plasmar estructuras de datos reales, en vigencia y conocidas actualmente, para la integración de las nuevas implementaciones y su adaptación.

2. **Modelo de flujo de datos**: da como resultado los diagramas de flujo de datos o DFD con su documentación, que representarían todos los procesos, flujos de datos y entidades externas. Cubre la identificación, modelado y documentación de los flujos de datos en la estructura o sistema.

   Un diagrama de flujo de datos o DFD recoge el sentido de los procesos en un sistema de información, usando elementos normalizados, como rectángulos para las entidades, flechas para los flujos y su sentido, círculos para procesos y líneas paralelas para almacenar datos. En la *Figura 2-15* se puede ver la estructura y elementos de un diagrama DFD.

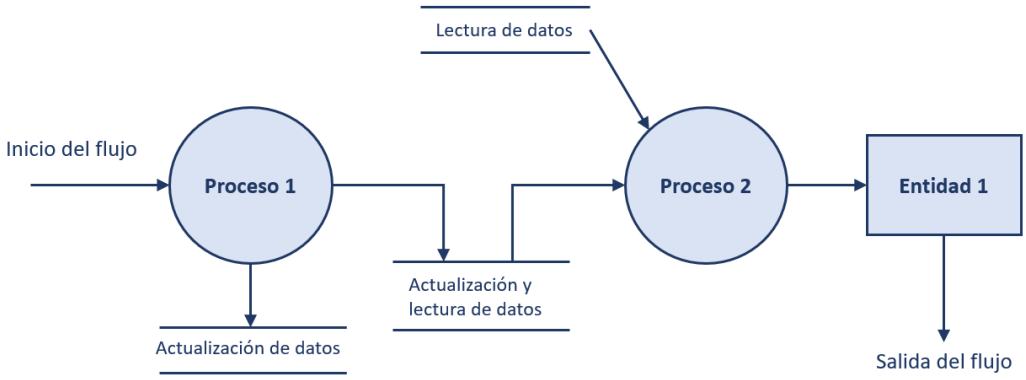

Figura 2-15. Elementos de un DFD.

Realización propia

3. **Modelo de eventos de entidad**: abarca el comportamiento del sistema temporalmente, para lo que se necesita cerrar un historial de eventos y acciones entre las entidades definidas en la vista anterior. La formalización del comportamiento y su documentación se realiza con la identificación, modelado y documentación de los eventos que involucran a cada entidad y la forma de ocurrir o secuencia.

   El resultado es la elaboración de diagramas de comportamientos de entidad, para lo que se utilizan, principalmente, dos tipos:

   a. **ELH** (diagrama de historia de vida de entidades) → recoge el proceso desde que una entidad se crea hasta que desparece, pasando por todas las actualizaciones en el tiempo.

   b. **ECD** (diagrama de correspondencia) → recoge el total de ocurrencias de una entidad que interviene en una relación.

Como se ha indicado, se centra y presenta pautas y herramientas para las primeras fases del ciclo de vida de un proyecto: viabilidad, análisis y diseño. Su fuerte focalización en una precisa documentación, apoyada en un riguroso análisis que permita un diseño viable del proyecto, hace que queden fuera de su alcance las fases de ejecución y mantenimiento.

Para cubrir las primeras etapas del ciclo de vida del proyecto que dan como resultado una documentación rigurosa, la metodología SSADM se descompone en 7 niveles. Previo a la definición de cada uno, es necesario ubicarlos dentro de la etapa del ciclo de vida a la que corresponden, para la que usamos la *Tabla 2-2*:

FASE DEL CICLO DE VIDA	NIVEL SSADM
Análisis de Viabilidad	Nivel 0
Análisis de Requerimientos	Nivel 1
	Nivel 2
Definición de Requerimientos	Nivel 3
Definición del Sistema Lógico	Nivel 4
	Nivel 5
Diseño	Nivel 6

Tabla 2-2. Etapas y niveles en SSADM.

Realización propia

A continuación, se describen las tareas que se realizan en cada uno de los niveles:

- **Nivel 0: Estudio de Viabilidad** → definición del alcance del proyecto, investigación y elección de las opciones de desarrollo en función de la estimación de costes y beneficios, y análisis de riesgos.
- **Nivel 1: Investigación del entorno actual** → definición del producto como nueva implementación o versión, análisis de requerimientos, modelado, y búsqueda y detección de puntos débiles.
- **Nivel 2: Opciones de negocio del sistema** → estudio de los requerimientos del nivel anterior y definición técnica y funcional de las necesidades del cliente.
- **Nivel 3: Definición de requerimientos** → paso de los requerimientos estudiados a especificaciones y paso de análisis a diseño, mediante el modelado.
- **Nivel 4: Opciones técnicas del sistema** → propuesta de alternativas técnicas al nivel anterior, selección de opciones más adecuadas de desarrollo.
- **Nivel 5: Diseño lógico** → se realiza en paralelo al nivel anterior para obtener un diseño lógico en un entorno de desarrollo, que cumpla los requerimientos del cliente, definidos y analizados en niveles inferiores.
- **Nivel 6: Diseño físico** → paso del diseño realizado con anterioridad al entorno productivo, usando técnicas de selección de entornos.

2.3.6 MÉTRICA

MÉTRICA es una metodología pública, promovida por el Ministerio de Administraciones Públicas del Gobierno de España [18], para sistematizar las actividades que dan soporte al ciclo de vida de un proyecto de software; es decir, para la planificación, desarrollo y mantenimiento de sistemas de información.

Recoge el conjunto de normas, herramientas, técnicas y documentación para desarrollar y mantener software de distintos niveles de complejidad, tamaño y ámbito.

Se basa en el modelo de procesos del ciclo de vida de desarrollo *ISO/IEC 12205 Information Technology – Software Life Cycle Processes*, y en la *norma ISO/IEC 15504 Software Process Improvement And Assurance Standards Capability Determintaion*.

Su primera versión fue publicada en 1989, estando actualmente vigente su tercera versión. MÉTRICA ha ido adaptándose a los avances tecnológicos, necesidades y complejidades que han ido surgiendo desde su primera publicación a través de las distintas versiones:

- **MÉTRICA V1**: publicada en 1989, ofrecía una guía y métodos para optimizar la definición, costes y tiempos a la hora de obtener un producto.
- **MÉTRICA V2**: publicada en 1993, es una guía técnica, de referencia y de usuario.
    - **MÉTRICA V2.1**: publicada en 1995 como una nueva versión que sustituía a la V2, ofreciendo una mayor claridad y estructuración, en 5 fases, enfocada al soporte ante cambios en el sistema.
- **MÉTRICA V3**: publicada en 2001, ofrece procesos o interfaces con orientación a la organización y desarrollo, considerando diferentes tecnologías actuales, que aparecieron y evolucionaron desde la versión anterior de la metodología. Es la versión vigente, que se va a describir a continuación.

La finalidad de MÉTRICA V3 es mejorar la productividad y la calidad de los servicios, siendo una metodología de orientación a procesos para la normalización de éstos en las organizaciones. Permite la identificación de las actividades a llevar a cabo, los elementos de entrada de estas actividades y los resultados o salidas.

Los procesos a los que está orientada se descomponen en un nivel más detallado, actividades, y éstas en tareas, que son la descripción de las acciones, prácticas, técnicas, productos y participantes. Estos procesos que conforman la estructura de MÉTRICA V3 son agrupados por bloques, definidos a continuación e ilustrados en la *Figura 2-16*.

- **PSI** → Planificación de Sistemas de Información
- **EVS, ASI, DSI, CSI, IAS** → Desarrollo de Sistemas de Información
- **MSI** → Mantenimiento de Sistemas de Información

Figura 2-16. Procesos de MÉTRICA V3.

Realización propia

A continuación, se describe cada uno de estos 7 procesos:

- **PSI: Planificación de Sistemas de Información** → su objetivo es ofrecer un marco estratégico de referencia para los Sistemas de Información concretos de una organización. Para ello, se realiza un estudio de las necesidades de la organización para definir los requisitos y los modelos conceptuales, se evalúan las opciones y propuestas, se analizan las prioridades, y se elabora una planificación con un gran nivel de detalles, manteniéndose actualizada de forma sistemática.
- **EVS: Estudio de la Viabilidad del Sistema** → con el objetivo de analizar las necesidades en función de las condiciones económicas, técnicas, legales y operativas, realiza una valoración, a través de un estudio táctico, de las alternativas en base al riesgo e impacto. Como resultado, da la respuesta a si abordar o no el desarrollo según la viabilidad analizada, basada en estimación de riesgos, costes, contexto y planificación.
- **ASI: Análisis del Sistema de Información** → su finalidad es dar el detalle y especificación de los requisitos y modelos que cubran las necesidades del cliente para, en el paso posterior, comenzar el diseño sobre seguro. Es necesario, por tanto, una descripción del Sistema de Información, acotando su alcance, detallando y clasificando los requisitos funcionales y no funcionales. De esta forma, se pueden elaborar los modelos, que se analizan y modifican hasta cubrir por completo los requisitos. Finalmente, se inicia la elaboración del plan de pruebas.

  Este proceso da como resultado una lista detallada de los requisitos, los estándares y normas a aplicar, definición del sistema y subsistemas, y los modelos de procesos y lógicos.
- **DSI: Diseño del Sistema de Información** → da la definición del sistema, sus componentes y el entorno que le va a dar soporte, mediante el diseño detallado del sistema, subsistemas y entornos en los que se va a probar y desplegar. También se recogen todas las especificaciones para su construcción y pruebas, dando como resultado el detalle de construcción, pruebas, implantación y procesos de despliegue.
- **CSI: Construcción del Sistema de Información** → partiendo del diseño del paso previo, su objetivo es la construcción y prueba de los componentes del sistema, dando como resultado el código de cada componente, los procedimientos, control y seguridad, y los manuales de usuario y formación de éstos. Es necesario, en este proceso, desarrollar el código y procedimientos, diseñar y ejecutar las pruebas

(unitarias, de integración y del sistema), y la realización de los manuales de usuario para darle formación.

- **IAS: Implantación y Aceptación del Sistema** → su objetivo es realizar la entrega y el despliegue del sistema, tras la aceptación total del resultado. Se realiza un nuevo análisis de viabilidad (EVS), previo a la preparación del despliegue, para asegurar la calidad y validez del sistema. Una vez preparadas las infraestructuras y definido el plan de implantación, se despliega y, por último, se realizan pruebas de aceptación y mantenimiento.
- **MSI: Mantenimiento del Sistema de Información** → es el proceso que tiene como finalidad obtener una nueva versión del sistema actual, partiendo de las peticiones de los usuarios, errores y problemas detectados, o una actualización necesaria. Las peticiones por parte del cliente se tratan de forma diferente a los problemas o necesidades, pues sería necesario un nuevo análisis a partir de los requerimientos. En cuanto a los problemas o necesidades, deben formar parte de la estimación de riesgos e impactos que, igualmente, se debe analizar y valorar, además de someter a regresión (reprobar para garantizar que el problema se ha solventado).

  De esta forma, el proceso da como salida un listado de peticiones y/o modificaciones analizadas, valoradas, dentro de una planificación, con un plan de pruebas de regresión, y una evaluación de éstas.

MÉTRICA V3 ofrece 4 interfaces de orientación y ayuda en actividades esenciales para la gestión de proyectos basados en esta metodología. Las interfaces, que aportan un perfeccionamiento y mejora de los procesos descritos anteriormente, se muestran en la *Figura 2-17* y se detallan a continuación.:

- **GP: Gestión de Proyectos** → su objetivo es mejorar la planificación, seguimiento y control de los recursos y actividades del desarrollo. Esta interfaz se estructura en 3 conjuntos de actividades:
  - **GPI**: Inicio del Proyecto → análisis de viabilidad, estimación de esfuerzo y planificación.
  - **GPS**: Seguimiento y Control del Proyecto → asignación de tareas, gestión de incidencias, impacto ante cambios. Se realiza en los procesos de desarrollo para garantizar la planificación y el correcto desempeño de las tareas.
  - **GPF**: Finalización del Proyecto → cierre del proyecto (entrega, manuales, formación).
- **SEG: Seguridad** → incorpora mecanismos adicionales de seguridad en el sistema para dar seguridad extra al desarrollo, dotándolo de mayor consistencia y robustez**.**
- **CAL: Aseguramiento de la Calidad** → proporciona un marco común para el diseño y ejecución de planes de pruebas para asegurar la calidad de los procesos y productos. Se realizan por un grupo independiente a los responsables para añadir otro filtrado más a la hora de aceptar el resultado.
- **GC: Gestión de la Configuración** → asegura el mantenimiento de la integridad del sistema, controlando los cambios y teniendo la versión más actualizada y clara de documentación e información. Permite realizar un mantenimiento eficiente del sistema, reduciendo el tiempo de implementación de un cambio, reduciendo los errores y aumentando la calidad del producto.

Figura 2-17. Interfaces de MÉTRICA V3.

Realización propia

Cabe destacar que esta metodología, además de usar técnicas conocidas para hacer diagramaas, modelos y análisis, añade otras técnicas interesantes para la interfaz de Gestión de Proyectos (GP):

- **Métodos de estimación** *Albrecht* y *MARKII* para calcular el tamaño, funcionalmente hablando, de una aplicación o sistema. Descompone el sistema en elementos de entrada, de proceso y de salida.
- **WBS** (Work Breakdown Structure) o técnica de descomposición en árbol para estructurar las actividades de un proyecto en función de su naturaleza, facilitando la tarea de planificación y realización de diagramas (como los que se describen a continuación).
- **PERT** (Program Evaluation and Review Technique) o método que permite plasmar las tareas del proyecto y las dependencias entre éstas, controladas por acontecimientos o eventos que marcan el inicio y fin de cada tarea. Así, consigue ubicar las distintas tareas en la planificación.
- **Diagrama de Gantt**, que permite representar el plan de trabajo, recogiendo en un mismo diagrama las tareas a realizar y el intervalo de tiempo que ocupan, marcado su inicio y fin. Como ya se indicó en el apartado de evolución de la gestión de proyectos en el primer bloque de este documento, el diagrama de Gantt es la representación clásica del plan de un proyecto.

2.3.7 MERISE

MERISE es una metodología pública [19] creada por el Ministerio de Industria de Francia para la administración del país. Ofrece un método integrado de análisis, concepción y gestión de proyectos, dentro de un marco y lenguaje común para desarrollos informáticos.

Comenzó a implementarse y desarrollarse en 1972, siendo publicada su primera versión en 1976 por parte del CTI (Centre Technique Informatique) del Ministerio de Industria de Francia.

En 1977 surgió otra metología llamada RACINES que aporta una estrategia para manejar la informatización de forma ordenada. Mientras tanto, MERISE, va tomando un rumbo más orientado al análisis y diseño de sistemas de información a través de técnicas de modelado, planes de trabajo y técnicas de integración de sistemas en el marco definido por RACINES.

Era de esperar que, posteriormente, ambas metodologías acabaran fusionándose en una, que mantuvo el nombre de MERISE, y que fue publicada por el Ministerio de Industria de Francia en 1982.

MERISE abarca las cuatro primeras fases del ciclo de vida de desarrollo de un proyecto de software (viabilidad, análisis, diseño e implementación), pero deja fuera de su método la fase explícita de mantenimiento. No obstante, quedaría cubierta mediante la documentación detallada y completa.

Esta metodología se apoya en la idea de 2 ciclos complementarios: el ciclo de **abstracción** y el ciclo de **decisión**, como puede verse en la *Figura 2-18*. A su vez, el ciclo de abstracción está basado en 3 niveles:

1. **Conceptual**: nivel donde se definen los objetivos y limitaciones del proyecto. En él se realiza un tratamiento de los datos y de los procesos en base a los modelos conceptuales. Se pasa de lo abstracto o ideas a lo concreto, haciéndose una definición conceptual.
2. **Organizativo**: nivel donde se define la implantación de una organización concreta, y donde se realiza un tratamiento de los datos en base al modelo organizativo, enfocado a los procesos. En concreto, se definen los objetivos relacionados con el orden y las secuencias a seguir.
3. **Físico u Operativo**: nivel donde se integran los medios técnicos, usándose el modelo físico para los datos, y el modelo operativo para los procesos. Se decide quién realiza cada tarea o proceso, y se definen recursos como la documentación y el hardware.

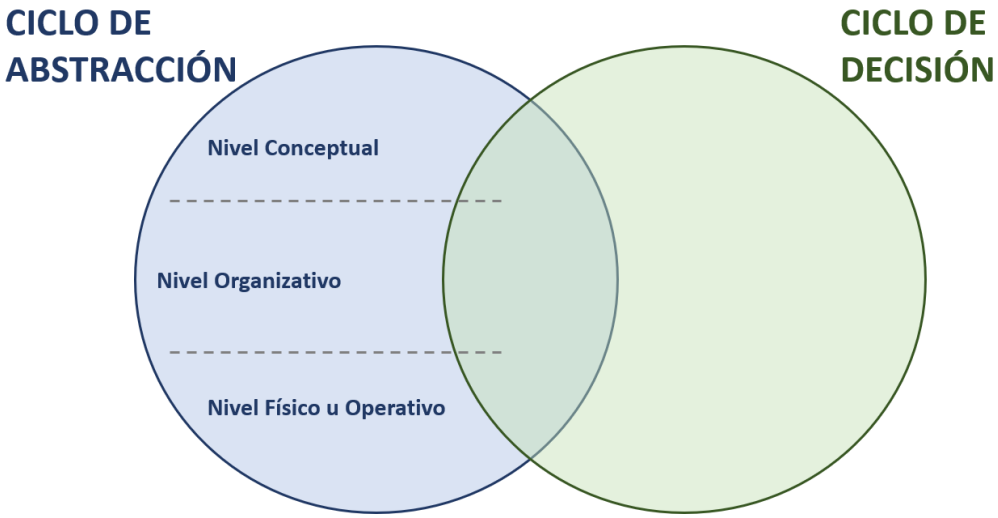

Figura 2-18. Ciclos y niveles de MERISE.

Realización propia

Como se ha indicado, MERISE cubre explícitamente todas las etapas del ciclo de vida de desarrollo de un proyecto software, excepto la de mantenimiento, que se documentaría rigurosamente.

Para poder ubicar cada una de las fases que ofrece esta metodología dentro del ciclo de vida del proyecto, se muestra la *Tabla 2-3*.

FASE DEL CICLO DE VIDA	FASE DE MERISE
Viabilidad	Estudio preliminar
	Estudio detallado
Análisis	Realización y puesta en marcha
Diseño	Diseño físico
Implementación	Implementación

Tabla 2-3. Fases de MERISE en el ciclo de vida.

Realización propia

A continuación, se indican las actividades que se llevan a cabo en cada una de las fases:

- **Estudio preliminar**: análisis de la situación actual y diseño de la propuesta de solución global, apoyado en los criterios de gestión y de la organización, en las decisiones del comité directivo del proyecto.
- **Estudio detallado**: definición de la solución a nivel funcional.
- **Realización y puesta en marcha**: instalación y despliegue por parte del equipo de desarrollo, formación de los usuarios, implantación de los medios técnicos y organizativos, y proceso de recepción por el cliente.
- **Diseño físico**: descripción del entorno técnico, distribución de los datos en el sistema, y tratamientos de programas.
- **Implementación**: codificación de los programas y pruebas.

Puesto que se mencionan en las actividades de las fases y son necesarios para entender la metodología, se

definen 3 equipos o comités en el desarrollo de un proyecto basado en *MERISE*:

- **Comité Directivo**: define los objetivos y toma decisiones, controlando cada etapa para realizar el seguimiento del proyecto. Está formado por los directivos de las partes interesadas, el jefe del proyecto, los responsables de los servicios implicados y el responsable del servicio de informática.
- **Comité de Usuarios**: comprueba los diseños, valida informes y listados, resuelve dudas y problemas, y realiza un seguimiento de forma sistematizada para comprobar el cumplimiento de todas las actividades planificadas y asignadas. Está formado por los responsables de los servicios implicados.
- **Equipo de Desarrollo**: elaboración de informes y documentación de cada fase del desarrollo, realización de análisis, programación, instalación, despliegue, pruebas y puesta en funcionamiento. Está formado por el jefe del proyecto, analistas y programadores, además de un representante del comité de usuarios.

Como las demás metodologías, MERISE hace uso de diferentes técnicas para analizar, planificar y diseñar. Entre estas técnicas, es interesante mencionar las siguientes:

- Diagrama de Flujo de Datos (DFD) → representación gráfica de los flujos de información entre las diferentes entidades de la organización.
- Modelo conceptual de datos → como se ha visto en otras metodologías, se trata de un modelo entidad/relación, para la representación con estructuras o sistemas del mundo real.
- Modelo lógico de datos → transcripción o adaptación del modelo conceptual de datos al sistema de datos real elegido.
- Modelo conceptual de tratamientos → representación de acciones a realizar sobre los datos, basándose en redes para obtener resultados.
- Modelo organizativo de tratamientos → descripción de la ejecución de las acciones.
- Modelo operacional de tratamientos → obtención de los procedimientos manuales, fase en tiempo real y en tiempo diferido. Permite una descomposición de los procedimientos y fases anteriores.

2.4 Metodologías Ágiles

Las metodologías ágiles, enfocadas a proyectos de entrega de software, surgieron por la frustración en los tiempos de entrega y el no poder cambiar decisiones tomadas al principio del proyecto, lo que hacía que no se pudiera mejorar el producto, ni pudiera adaptarse a modificaciones y nuevas necesidades a medida que se avanzaba en su desarrollo.

Esta necesidad de adaptación, flexibilidad y retroalimentación rápida y necesaria, obviamente, era compartida por muchos líderes, que comenzaron a reunirse hasta que, a principios de 2001 en la Reunión de Snowbird en Utah, se recogieron estas necesidades en el famoso Manifiesto Ágil, quedando recogidos los principios de este método, y que engloba las metodologías que hasta ese momento se les conocía como "Metodologías de Desarrollo de Software de peso liviano", en contraposición a las metodologías tradicionales de la sección anterior, que se conocen como "metodologías pesadas".

No obstante, aunque el Manifiesto se redactó en 2001 y las metodologías ágiles se puedan entender como algo más actual y novedoso, el ciclo de vida incremental e iterativo se planteó y empezó a usarse en proyectos de software en la década de los 60's, por lo que su base es igual e incluso más antigua que los ciclos en cascada.

La que sí es más moderna y reciente es la definición o clasificación de estas metodologías, que tuvo lugar en 1990, como contraposición o respuesta a la definición de las "metodologías pesadas" que, como hemos visto, son muy robustas y estrictas en cuanto a estructuración y burocracia, lo que las convierte en lentas y "pesadas". Por este motivo, los métodos ágiles se llamaron, al principio, "métodos de peso liviano", pues descargaban el procedimiento tan estricto y cerrado de las otras.

Como también ocurre con las metodologías tradicionales, las ágiles no tienen que cubrir sí o sí todas las etapas de un proyecto, sino que se utilizan para ciertas fases y se fusionan para cubrir el proyecto en función de su tipo, cliente, producto o intereses.

Existen muchos métodos ágiles, que tienen en común la disminución de riesgos, basándose en iteraciones y la colaboración de todos los involucrados como un único equipo. A continuación, en la *Figura 2-19* se muestra un gráfico de las metodologías ágiles más usadas en el pasado año 2020, basado en los resultados del estudio llamado "State of Agile" que realiza todos los años la empresa *Version One* para conocer el uso a nivel profesional de los distintos métodos:

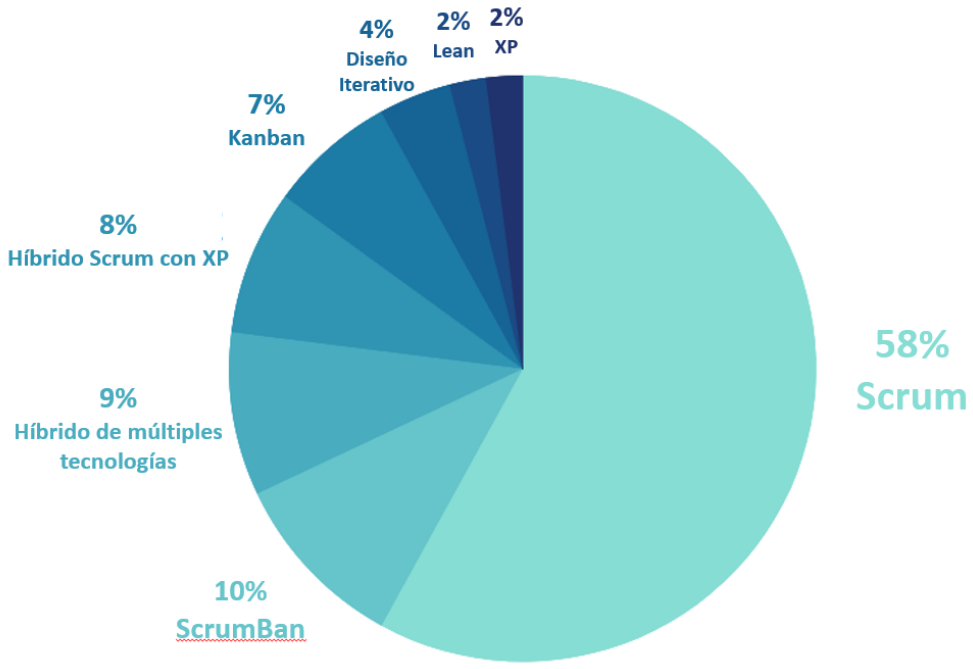

Figura 2-19. Metodologías ágiles más usadas en 2020.

Realización propia (basada en "State of agile" de Version One, consultado en la web oficial: https://stateofagile.com)

Todos los métodos tienen en común una serie de principios, reglas o buenas prácticas, que recogen la filosofía de esta familia de metodologías. Dicha filosofía podríamos definirla en base a 4 conceptos o términos claves:

1. **Incremento** → pequeñas entregas con ciclos de desarrollo rápido, que sirven de realimentación para los posteriores ciclos.
2. **Cooperación** → todos los involucrados (clientes, proveedor, terceros) interaccionan y trabajan como un único equipo.
3. **Sencillez** → métodos fáciles de entender y de aplicar, con una documentación suficiente para consultarla y usarla de manera correcta.
4. **Adaptación** → reacción ante modificaciones, mejoras o errores en cualquier momento del ciclo de vida del proyecto.

En las siguientes subsecciones se van a describir las metodologías ágiles más conocidas y utilizadas. Como se ha podido ver en el último gráfico, el método más usado actualmente, con mucha diferencia, es Scrum, motivo por el que se profundizará a mayor nivel de detalles. Además, puedo hacer una mayor aportación, al estar certificado de forma oficial, y por ser una de las herramientas en las que me apoyo profesionalmente en mi día a día.

Además, haremos un recorrido por varias metodologías ágiles para conocer otras posibilidades y tener una base de conocimiento de cara a realizar una comparativa y tener un criterio para poder elegir una, en función de las características del proyecto. Estas metodologías, además de SCRUM, son: eXtreme Programming (XP), Crystal, Agile Modeling (AM), Adaptative Software Development (ASD), Agile Unified Process (AUP), Dynamic Systems Development Method (DSDM), Feature Drive Development (FDD), Lean Software Development (LSD) y Kanban.

Como se ha recogido, todas ellas tienen en común la necesidad de adaptación y contacto frecuente con el cliente para conseguir un producto de calidad, basándose en la retroalimentación y las interacciones que permiten dar una solución adaptada 100% a los requerimientos iniciales y a los problemas surgidos durante su desarrollo.

Pero, antes de empezar el recorrido por los diferentes métodos, es necesario hacer referencia al Manifiesto Ágil y los Principios Ágiles recogidos en éste.

A. MANIFIESTO ÁGIL

El "Manifiesto Ágil" es el documento donde se recogen los principios de las metodologías ágiles, firmados y aprobados en 2001 por los siguientes líderes empresariales [20]: Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas.

El Manifiesto pretende recoger y hacer prevalecer **valores** sobre procesos, herramientas, negociaciones y planificación, quedando resumido en estos cuatro valores o fundamentos:

1. Individuos e interacciones sobre procesos y herramientas
2. Software funcionando sobre documentación exhaustiva
3. Colaboración con el cliente sobre negociación contractual
4. Respuesta ante el cambio sobre seguimiento de un plan

Es decir, podríamos decir que se deja a un lado la burocracia y el método tradicional y estático para dar

prioridad al dinamismo y la necesidad de adaptación.

B. PRINCIPIOS ÁGILES

Los doce principios del Manifiesto, basados en los cuatro postulados, son los siguientes:

1. Nuestra principal prioridad es satisfacer al cliente a través de la entrega temprana y continua de software de valor.
2. Son bienvenidos los requisitos cambiantes, incluso si llegan tarde al desarrollo. Los procesos ágiles se doblegan al cambio como ventaja competitiva para el cliente.
3. Entregar con frecuencia software que funcione, en periodos de un par de semanas hasta un par de meses, con preferencia en los periodos breves.
4. Las personas del negocio y los desarrolladores deben trabajar juntos de forma cotidiana a través del proyecto.
5. Construcción de proyectos en torno a individuos motivados, dándoles la oportunidad y el respaldo que necesitan y procurándoles confianza para que realicen la tarea.
6. La forma más eficiente y efectiva de comunicar información de ida y vuelta dentro de un equipo de desarrollo es mediante la conversación cara a cara.
7. El software que funciona es la principal medida del progreso.
8. Los procesos ágiles promueven el desarrollo sostenido. Los patrocinadores, desarrolladores y usuarios deben mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica enaltece la agilidad.
10. La simplicidad como arte de maximizar la cantidad de trabajo que no se hace, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos que se auto-organizan.
12. En intervalos regulares, el equipo reflexiona sobre la forma de ser más efectivo y ajusta su conducta en consecuencia.

2.4.1 SCRUM

SCRUM es la metodología *AGILE* más utilizada, sobre todo en desarrollo de software, ya que permite adaptación, rapidez, flexibilidad y eficacia, garantizando transparencia en la comunicación. La clave es fusionar al cliente y proveedor (encargado de ofrecer una solución y desarrollar el producto al cliente) en un único equipo, creando un ambiente de responsabilidad colectiva y progreso continuo, basado en un modo de trabajo iterativo e incremental. En esta estrategia incremental se lleva a cabo un solapamiento de las diferentes fases de desarrollo, en vez de ejecutarlas en cascada o de forma secuencial.

Su estructuración lo hace compatible con el desarrollo y mantenimiento de productos de todo tipo de industrias, siendo un marco de trabajo adaptativo con la finalidad de ofrecer un resultado lo más eficiente y optimizado posible.

Para entender mejor el concepto, se debe hacer referencia a la definición del término SCRUM, que es una formación de rugby en la que ambos equipos se unen y atenazan para poder obtener el balón sin tocarlo con la mano, como se puede visualizar en la *Figura 2-20*. Es decir, se podría considerar una metáfora de cooperación y unión entre diferentes equipos para conseguir un objetivo común.

Figura 2-20. Formación SCRUM en el rugby.

Imagen tomada de la url: https://www.mediotiempo.com/mas-deportes/coronavirus-proponen-cambios-rugby-reducir-contacto-contagio

Al ser la metodología más empleada actualmente, sobre la que tengo una mayor formación y experiencia, y la que usaremos en el caso práctico, se va a realizar una definición exhaustiva y detalla de ésta.

Aunque los términos de este marco de trabajo se desarrollen en profundidad en los siguientes subapartados, para un mejor entendimiento y contextualización, se podría dar una visión general de la siguiente forma:

SCRUM es un marco de trabajo basado en una serie de principios que permiten la cooperación y transparencia entre dos equipos para crear o mantener un producto, de forma adaptativa, estructurando el trabajo en ciclos o *sprints* acotados en el tiempo (no se prorrogan). Estos ciclos se acuerdan por el Equipo Scrum (desarrolladores y *Scrum Master*), de modo que, al final de cada *sprint* se pueda entregar algo tangible y "terminado". Diariamente, se revisa el avance y se priorizan las tareas para cumplir el objetivo y, al final del *sprint*, el cliente o *Product Owner* pueda probar el entregable y dar un *feedback* (valoración con comentarios de mejorar) que será añadido en el siguiente *sprint*. Ésta es la clave de nuestro marco de trabajo, la iteración para adaptar y mejorar el producto final.

En los siguientes subapartados se definen y detallan los **principios** de SCRUM, los **roles**, **artefactos** y **eventos**, que se pueden visualizar, resumidos, en la *Figura 2-21*:

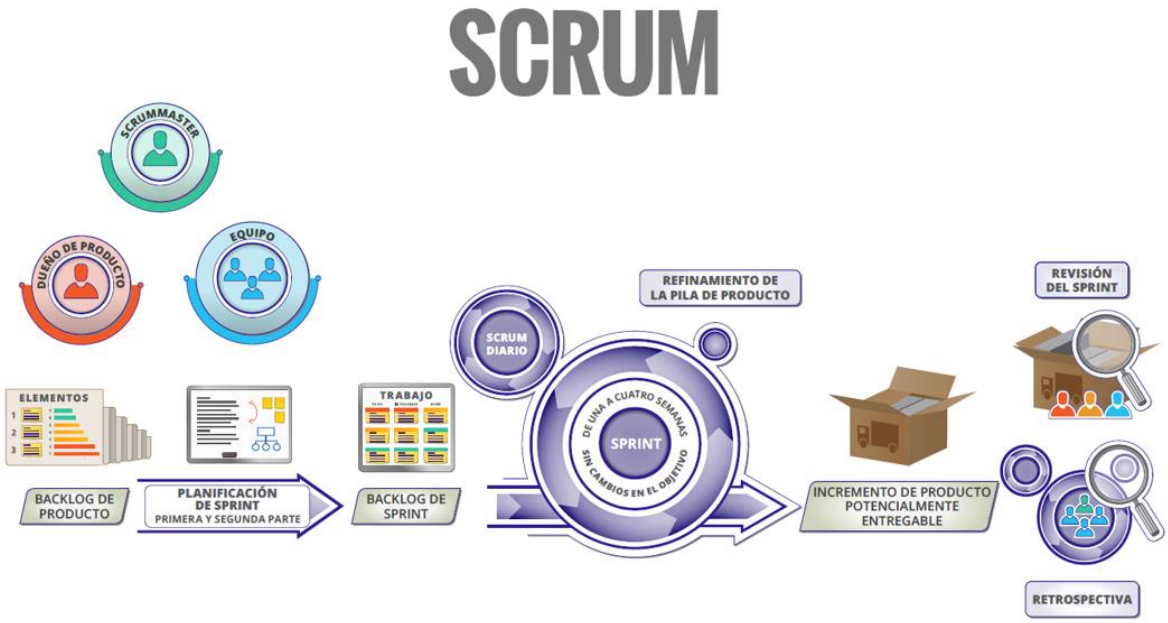

Figura 2-21. Esquema resumen de SCRUM.

Imagen tomada del material oficial del curso recibido por CertiProf (url de la web oficial: https://certiprof.com)

2.4.1.1 Principios de SCRUM

Los 6 principios de SCRUM son las pautas básicas para conseguir la plena cooperación, definidos por sus desarrolladores, Ken Schwaber y Jeff Sutherland, quienes proporcionaron la guía oficial de SCRUM en el año 2010 [21] y la mejoraron en 2013 [22]:

1. **Control del Proceso Empírico** → Supone la filosofía en torno a la que gira Scrum, basándose en las tres ideas de transparencia, inspección y adaptación.
2. **Auto-organización** → Se consigue un mayor compromiso y responsabilidad si los equipos son autónomos y se organizan internamente. De esta forma, los equipos añaden innovación y creación, propiciando un mayor crecimiento.
3. **Colaboración** → Principio centrado en las tres bases colaborativas: conciencia, articulación y apropiación. Además, apuesta por la visión del proyecto como un proceso creativo, fruto de la puesta en común de ideas con los equipos que trabajan en él.
4. **Priorización basada en el valor** → Principio básico de la metodología Ágil, que prioriza el valor sobre los medios en todo el ciclo de vida del proyecto.
5. **Time Box** → La limitación es el tiempo, por lo que cada evento o fase debe ceñirse a la estimación temporal, sin abarcar más.
6. **Desarrollo Iterativo** → Clave para gestionar los cambios y mejoras con el fin de generar productos que cubran y satisfagan las necesidades del cliente o negocio.

En base a los 6 principios, recogidos de forma visual en la *Figura 2-22*, SCRUM podría ser resumida como una metodología orientada al resultado, basada en el compromiso, que genera valor y promueve el equipo. La *Figura 2-23* ilustra esta definición.

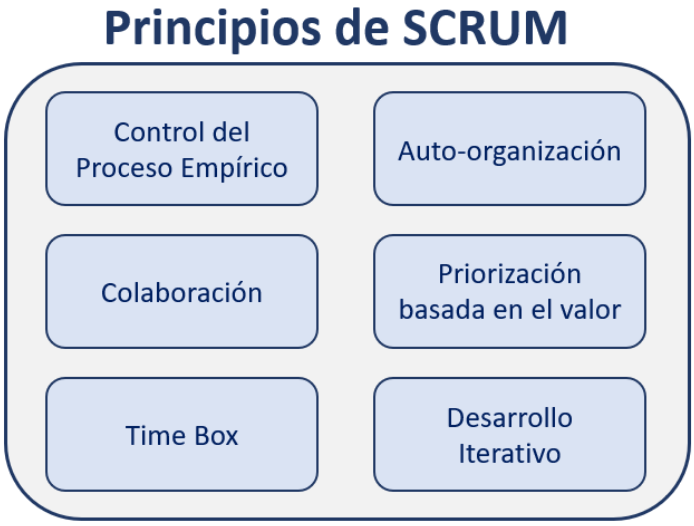

Figura 2-22. Principios de SCRUM.

Realización propia

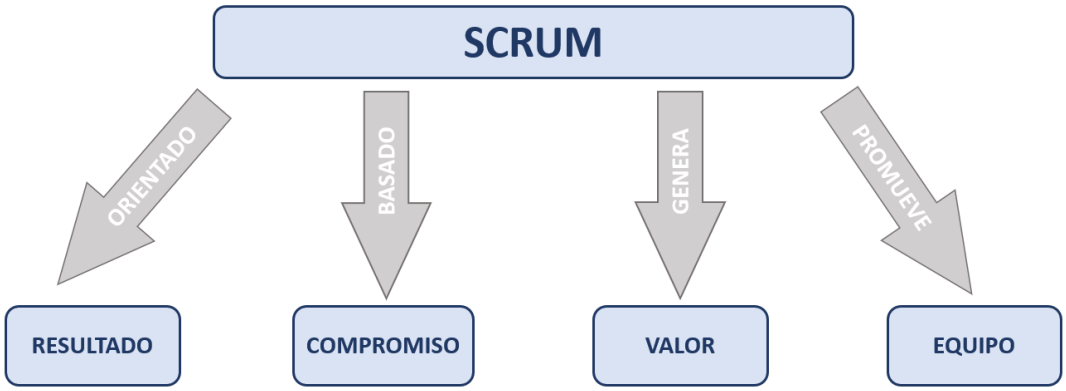

Figura 2-23. Definición de SCRUM basada en sus principios.

Realización propia

2.4.1.2 Roles en SCRUM

Los roles en SCRUM son los perfiles o dedicaciones de los miembros del equipo, para cumplir con los principios descritos. De esta forma se fomenta y facilita la auto-organización y se optimiza la productividad, creatividad y flexibilidad entregando productos iterativa e incrementalmente.

Los roles o elementos del Equipo Scrum, mostrados en la *Figura 2-24* son los siguientes:

1. *Product Owner* (Propietario del Producto)
2. *Development Team* (Equipo de Desarrollo)
3. *Scrum Master* (Facilitador de Proyecto)

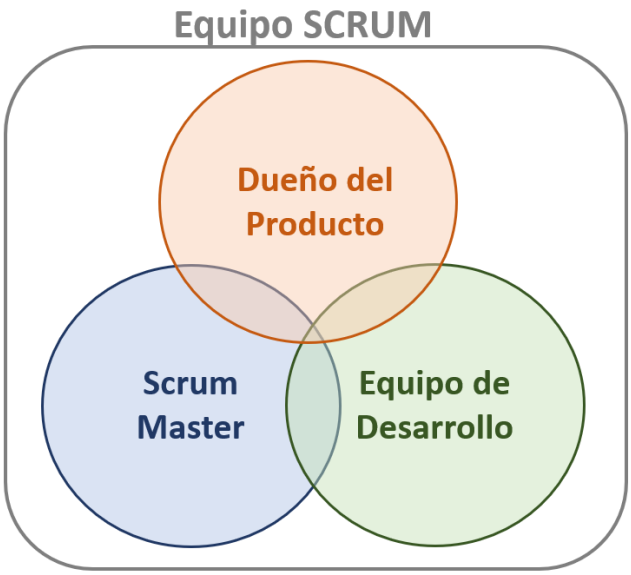

Figura 2-24. Roles en SCRUM.

Realización propia

A continuación, se detalla la funcionalidad de cada uno de ellos:

A. Product Owner

El *Product Owner* es, como dueño del producto, el responsable de optimizar el resultado y valor del producto, gestionando la lista de tareas o *Product Backlog* con el equipo de desarrollo.

Al ser el dueño del producto, debe servir de guía al equipo que está creando su producto, tomando decisiones argumentadas para optimizar y mejorar tanto el proceso de creación como, por consiguiente, el producto final.

Sus tareas podrían definirse de la siguiente forma:

- Definir y ordenar la pila del producto (*Product Backlog*) o lista de tareas para conseguir los objetivos de la forma más óptima y eficiente.
- Transferir los requerimientos al Equipo de Desarrollo de forma clara y transparente, asegurándose de que todo el equipo lo entiende para poder trabajar de forma autónoma, añadiéndole valor personal al producto.
- Se trata de una sola persona, no de un comité. Si, por la parte del cliente, un comité decide cambiar requerimientos o su prioridad, debe hacerlo a través de *Product Owner*, que lo transmitirá al resto del Equipo Scrum.

B. Development Team

El *Development Team* o equipo de desarrollo es el conjunto de profesionales que trabajan para entregar los incrementos "terminados" del producto, y que pondrán en producción el producto final. Los incrementos los crea el equipo de desarrollo, pues es el que tiene mayor capacidad para estimar los avances, ya que implementa y crea cada uno de los requerimientos del *Product Backlog* definido por el *Product Owner*.

En cuanto al tamaño del equipo de desarrollo, debe ser lo suficientemente grande para cubrir la cantidad de trabajo, pero lo suficientemente pequeño para permitir la comunicación y transparencia (mantenerse *Ágil*). Este aspecto lo va a definir el propio equipo, pues sabe cuán voluminoso o complejo es el trabajo que

hay que realizar.

Las características del equipo de desarrollo serían las siguientes:

- Auto-organización: el equipo define los incrementos del *Product Backlog*, sin recibir indicaciones ni decisiones de ningún otro rol. Pueden ser aconsejados o pedir ayuda para aclaración, pero jamás se le impondrá una decisión para acometer el desarrollo del producto la definición de los incrementos.
- Multifuncionalidad: posee las habilidades necesarias para llevar a cabo los incrementos del producto.
- Unidad y homogeneidad: el equipo no se divide en sub-equipos ni hay títulos o rangos, sino que se considera un todo o caja negra. Obviamente, habrá miembros especializados y responsables de ciertas áreas o detalles, pero la responsabilidad reca en el equipo de desarrollo como un todo.

C. Scrum Master

El *Scrum Master* es el responsable o líder que asegura la correcta aplicación de SCRUM, basándose en su guía, aportando al resto de roles el conocimiento sobre prácticas, reglas y valores.

Presta servicio, haciendo de intermediario, al resto de componentes del Equipo Scrum y a la Organización de la siguiente forma:

- Al *Product Owner*: hacer entender las técnicas y prácticas de SCRUM para gestionar correctamente el *Product Backlog*, definiendo los requerimientos y facilitando los *eventos* de forma clara y concisa. Esto conlleva a una mejora de la planificación dentro de un marco de trabajo empírico.
- Al *Developer Team*: guiar al equipo para su auto-organización y multifuncionalidad para crear productos con un alto valor. Es necesario, para cumplir estos objetivos, que el *Scrum Master* ofrezca los *eventos* necesarios y elimine impedimentos.
- A la *Organización*: guiar y ayudar a todos los empleados para adoptar SCRUM de forma correcta, planificando su implementación, motivando cambios para incrementar la productividad del Equipo Scrum, cooperando con otros *Scrum Masters*.

2.4.1.3 Eventos en SCRUM

Los eventos en SCRUM son periodos o compartimientos de tiempo limitado o *time-boxes*, limitados por una duración máxima. Su finalidad es regularizar y minimizar las necesidades de reuniones en un proceso común de gestión y desarrollo de un proyecto. Así se consigue cumplir con los principios del SCRUM de inspección y transparencia.

Hay un evento (*sprint*) que contiene al resto. El conjunto y sincronización de todos hace posible la adaptación e iteración, claves en nuestro marco de trabajo.

Los diferentes tipos de eventos son los siguientes:

- *Sprint*
- *Sprint Planning* (Planificación del Sprint)
- *Daily Scrum* (Scrum Diario)
- *Sprint Review* (Revisión del Sprint)
- *Sprint Retrospective* (Retrospectiva del Sprint)

A. Sprint

El *sprint* es el pilar básico o corazón del SCRUM. Es un periodo de tiempo de, aproximadamente un mes, durante el que se elabora un incremento "Terminado" del producto. En este momento es necesario hacer un inciso para definir el atributo "Terminado": es un estado subjetivo, impuesto por el equipo Scrum, basado en el cumplimiento de una serie de elementos del *Product Backlog*, que permite crear un bloque independiente, ejecutable y que puede ser entregado o puesto en producción. Por este motivo, el incremento elaborado en el Sprint debe ser utilizable y potencialmente desplegable.

El *Sprint* contiene la planificación del mismo (*Sprint Planning*), los Scrums Diarios (*Daily Scrums*), el trabajo del equipo de desarrollo, la Revisión del Sprint (*Sprint Review*) y su Retrospectiva (*Sprint Retrospective*).

Conociendo los elementos contenidos en un Sprint, hay que aclarar que un nuevo Sprint comienza de forma inmediata tras la finalización del anterior. Es lo que hace posible la adaptación y entregas iterativas en SCRUM, ya que cada *sprint* puede considerarse como un mini-proyecto, con los requerimientos y el tiempo definidos y acotados, motivos por los cuales, dentro de un Sprint, no se pueden realizar modificaciones que afecten al objetivo ni disminuir la calidad, aunque se pueda negociar un cambio de alcance con el *Product Owner*. Esta misma persona también es la única que podría, en caso de necesidad y aconsejado por el resto del equipo Scrum, cancelar un Sprint. Los motivos que pueden conllevar a una cancelación van relacionados a la posibilidad de que el producto quede obsoleto, y tendría como consecuencia una replanificación del Sprint, así como un análisis del *Product Backlog*. Consumen muchos recursos, pero son muy poco comunes, ya que precisamente SCRUM apuesta por un trabajo empírico iterativo para adaptarse a las necesidades continuamente.

B. Sprint Planning

El *Sprint Planning* es el proceso para plasmar y recoger la planificación del *sprint*, a través del trabajo colaborativo de la totalidad del Equipo Scrum. Se trata de una reunión con una duración de no más de 8 horas para *sprints* de un mes.

En este evento, el *Scrum Maste*r debe cerciorar la viabilidad del trabajo planificado, así como su ejecución y entendimiento por parte de todo el Equipo. El desafío de un *Sprint Planning* es que, una vez terminado, el Equipo de Desarrollo sea capaz de explicarle al *Product Owner* y al *Scrum Master* el objetivo del *Sprint* y cómo van a trabajar de forma auto-organizada para lograr un *Incremento* "Terminado", definiendo un *Spring Backlog* (pila del *Sprint*).

La planificación del *Sprint* debe realizarse con una meta u objetivo (*Sprint Goal*), contestando a dos preguntas claves:

a. ¿Qué puede hacerse y entregarse en el Sprint?
b. ¿Cómo se conseguirá realizar el trabajo necesario para la entrega del Incremento?

Con respecto a la pregunta a., el *Product Owner* discute con el Equipo de Desarrollo qué elementos del *Product Backlog* se pueden realizar para que la meta (*Sprint Goal*) sea un entregable o incremento "Terminado". Los elementos de entrada al *Sprint Planning* serían el *Product Backlog*, el último incremento y el rendimiento del Equipo de Desarrollo en el pasado *Sprint*. Como consecuencia, el Equipo Scrum define el *Sprint Goal* con un *Sprint Backlog*, basado en los elementos de entrada, y una guía de cómo abordarlo.

Para contestar la pregunta b., una vez establecido el objetivo y acotados los elementos del *Product Backlog* que se van a realizar en el *sprint*, el Equipo de Desarrollo se auto-organiza y proyecta para lograr el objetivo. Tanto el *Product Owner* como el *Scrum Master* pueden servir de consulta (el primero para especificaciones del producto, y el segundo como guía para la aplicación de SCRUM).

Por último, como consecuencia a la adaptación y al aspecto iterativo del marco de trabajo usado, he de

comentar que el *Sprint Goal* puede, a medida que el Equipo de Desarrollo avanza en la implementación del incremento, variar lo suficiente como para negociar el alcance del *Sprint Backlog* o Pila del *sprint* con el *Product Owner*.

C. Daily Scrum

La *Daily Scrum* es una reunión diaria (de 15 minutos) de inspección y adaptación, en la que el equipo de desarrollo planifica el trabajo a realizar en la jornada que empieza. Con este evento se optimiza el desempeño a realizar, evaluando el progreso hacia el *Sprint Goal*, contenido en el *Sprint Backlog*. Se analiza el trabajo realizado desde la última *Daily Scrum* para plasmar el progreso y priorizar las tareas pendientes para el incremento.

En este evento, es clave realizar las siguientes preguntas:

1. ¿Qué avanzamos ayer?
2. ¿Cuánto vamos a avanzar hoy?
3. ¿Hay algún obstáculo?

La reunión es interna, del equipo de desarrollo, siendo el *Scrum Master* el responsable de que ésta se celebre todos los días y cumpla con el tiempo establecido.

Una buena práctica es hacer la *Daily Scrum* todos los días a la misma hora y en el mismo lugar.

Las *Daily Scrums* son un elemento muy importante para mantener la comunicación interna, promover la toma rápida de decisiones y mejorar el nivel de conocimiento, posibilitando la adaptabilidad y mejora del producto. Evitan las innumerables y, a veces, innecesarias reuniones que consumen tiempo y recursos.

D. Sprint Review

La *Sprint Review* o revisión del *sprint* es una reunión que se celebra una vez finalizado el *sprint* para realizar una inspección y revisión del incremento y, si se considera necesario, hacer una adaptación del *Product Backlog*.

Encabezada por el *Scrum Master*, que asegura el correcto desarrollo y duración de la reunión, se trata de una reunión informal (no de seguimiento) de todo el Equipo Scrum para revisar y optimizar, si fuera posible, el valor del Incremento mediante una retroalimentación. El resultado debe ser un *Product Backlog* revisado de cara al siguiente *Sprint*. Tiene una duración máxima de cuatro horas para un *Sprint* de un mes.

Las características y función de cada rol en de este evento son las siguientes:

- Asistentes: totalidad del Equipo Scrum. Además, el *Product Owner* puede invitar a otros interesados para ser asesorado.
- *Product Owner*: decide qué elementos del *Product Backlog* se dan por "Terminados" o no. Hace una revisión del estado hasta el momento del *Product Backlog* y orienta al equipo, proyectando los objetivos posibles y el tiempo.
- Equipo de desarrollo: muestra el Incremento, realizando una prueba del trabajo "Terminado", contesta preguntas del *Product Owner* y hace balance de los obstáculos y resolución de los mismos.
- *Scrum Manager*: asegura el evento, haciendo de guía para asegurar la metodología, y controla los tiempos.
- Totalidad del equipo scrum: colaboran para planificar el próximo *Sprint*, recopilando información valiosa en función del *Incremento* y el análisis de pros y contras detectados en el *Sprint* finalizado.

E. Sprint Retrospective

La *Sprint Retrospective* es una reunión de análisis retrospectivo, celebrada una vez se realiza la *Sprint* Review, y antes del *Sprint Planning* siguiente. Consiste en un auto-análisis o inspección interna con el fin de elaborar un plan de mejoras a llevar a cabo en el próximo *sprint*.

Debe ser una reunión de motivación y colaboración, apartándose de discusiones o culpabilidades. No hay que olvidar el principal potencial de este marco de trabajo, la iteración para la adaptabilidad y mejora. Para asegurar los valores de este evento, como en el resto, está la figura del *Scrum Master*.

Su duración no va más allá de tres horas para *sprints* de un mes.

El resultado debe ser la identificación de las mejoras sobre lo ya "terminado" a llevar a cabo en próximo *sprint*. No sólo en el incremento realizado, sino en el proceso y prácticas que conllevó. Es por ello por lo que, a veces, es necesario redefinir el atributo "terminado" para ciertos elementos del entregable.

En resumen, el objetivo de la *Sprint Retrospective* sería:

- Análisis del último Sprint, basándose en las personas, relaciones procesos y herramientas (principios ágiles).
- Análisis, identificación y ordenación por prioridad de las mejoras.
- Elaboración del plan de mejoras para su inserción en la Planificación del próximo Sprint.

En la *Figura 2-25* se muestra un diagrama que resume todos los eventos que conforman el *sprint*:

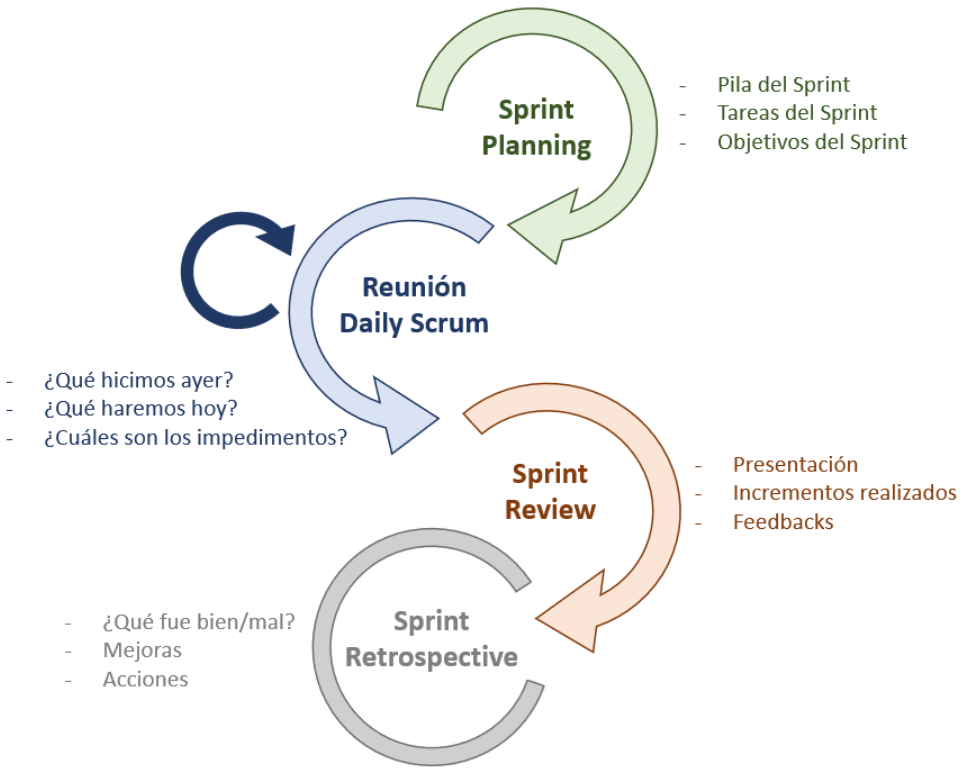

Figura 2-25. Eventos del Sprint en SCRUM.

Realización propia

2.4.1.4 Artefactos en SCRUM

Los artefactos en SCRUM son la materialización del valor del producto, ya que unos representan el trabajo a realizar y otro, el trabajo realizado.

Hasta ahora, se ha mencionado varios elementos clave en SCRUM: Pila del Producto o *Product Backlog*, Pila del *Sprint* o *Sprint Backlog*, e incremento. Son los llamados "Artefactos" en *SCRUM* y representan el trabajo en sí. En la *Figura 2-26* se recogen y representan los tres artefactos.

La función de los Artefactos es la de ofrecer de forma totalmente transparente la información clave del trabajo. Las decisiones para la optimización del valor del producto serán más certeras cuanto mayor sea la transparencia de los artefactos [23].

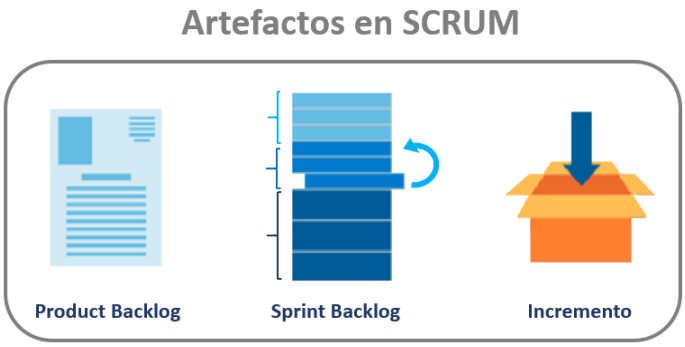

Figura 2-26. Artefactos en SCRUM.

Realización propia

A continuación, se define detalladamente cada uno de estos elementos:

A. Product Backlog

Es una lista o "pila" dinámica de los requisitos o especificaciones del producto, creada por el *Product Owner*, responsable de su contenido, disponibilidad y ordenación.

El *Product Backlog* no se entrega cerrado o completo, de ahí que sea una lista dinámica. Se crea con los requisitos conocidos hasta el momento, y va creciendo, modificándose y adaptándose a medida que el producto se va creando. Es decir, el *Product Backlog* "vivirá" mientras lo hago el desarrollo del producto, y viceversa.

Por lo tanto, el *Product Backlog* será una lista de requisitos, características, detalles, funcionalidades, correcciones y mejoras del producto, que termina siendo más larga y exhaustiva de lo que era al inicio del proyecto. A este proceso de modificación y adición de detalles se le conoce como refinamiento y es imprescindible en la creación de un producto de alto valor, adaptado y optimizado. El refinamiento se lleva a cabo por el *Developer Team* y el *Product Owner* mediante los eventos de *Sprint Review* y *Sprint Retrospective*.

Puesto que el *Product Backlog* representa el trabajo a través de los requerimientos, se puede analizar tanto el trabajo realizado como el restante. El *Product Owner* es el encargado, en cada *Sprint Review*, de evaluar el progreso y estimar el trabajo restante, pero sólo para tener una aproximación, pues *SCRUM* es un proceso empírico. Lo hace mediante prácticas de predicción basadas en la proyección de tendencias, como:

- *Burn Down*: predicción del trabajo pendiente (*Figura 2-27*)

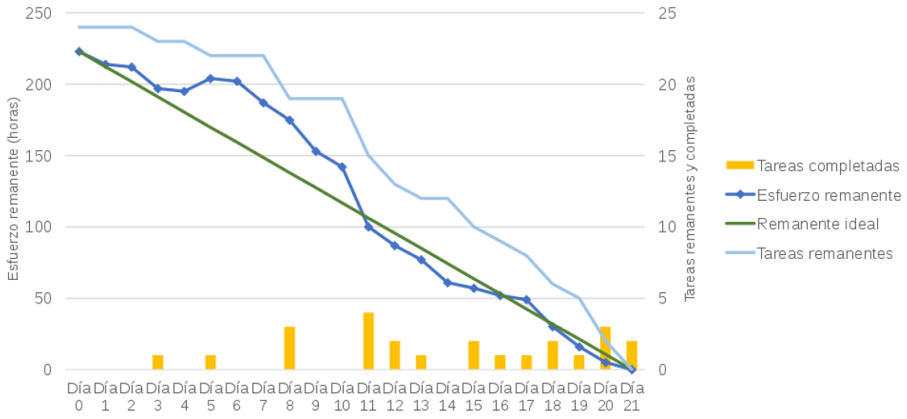

Figura 2-27. Ejemplo de Diagrama Burn Down.

Imagen tomada de la url: http://useragiledevelopment.blogspot.com/2011/05/burndown-chart.html

- *Burn Up*: trabajo completado (*Figura 2-28*)

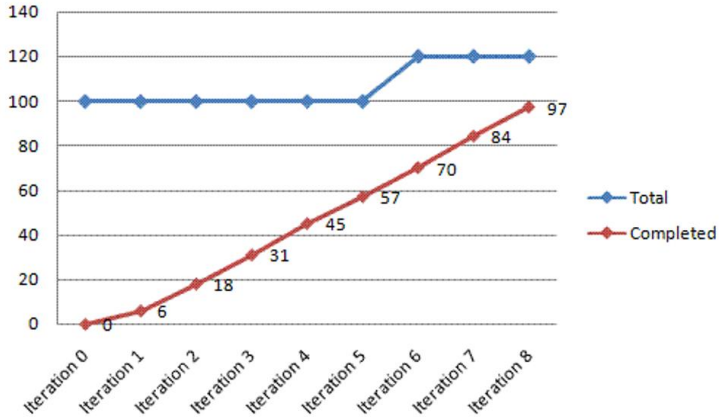

Figura 2-28. Ejemplo de Diagrama Burn Up.

Imagen tomada de la url: https://www.modernanalyst.com/Careers/InterviewQuestions/tabid/128/ID/3433/What-is-a-Burn-Up-Chart-and-how-does-it-differ-from-a-Burn-Down-Chart.aspx

- *Cumulative Flow*: flujo acumulado (*Figura 2-29*)

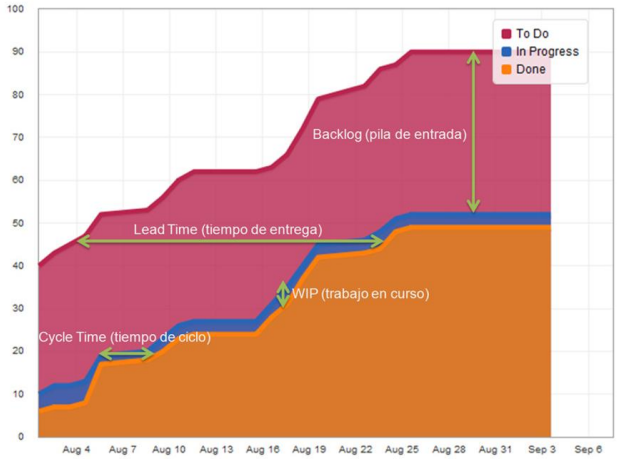

Figura 2-29. Ejemplo de Diagrama Cumulative Flow.

Imagen tomada de la url: https://scrum.menzinsky.com/2016/09/como-leer-un-cfd-un-diagrama-de-flujo.html

B. Sprint Backlog

Es el conjunto o "pila" de elementos del *Product Backlog* que se van a realizar o cubrir durante el *sprint* para cumplir el objetivo del mismo, que es la entrega de un incremento.

Se puede ver como el plan detallado del *Developer Team* para lograr un incremento de valor, de forma que es este equipo el único que puede modificarlo, eliminando elementos innecesarios o añadiéndolos si, durante la ejecución del trabajo en el transcurso del *Sprint*, lo ven necesario para lograr el objetivo. La revisión del *Sprint Backlog* tendría lugar en las *Daily Scrum*.

Por lo tanto, al igual que el *Product Backlog* (que sería la unión de todos los *Sprint Backlogs*), es una lista dinámica que parte de una estimación inicial y va creciendo y adaptándose a las necesidades que van surgiendo.

Remarcar que sólo el *Developer Team* gestiona el *Product Backlog*, pues es su plan diario de trabajo para lograr como objetivo un incremento válido y entregable al final del *sprint*. De la misma forma que el *Product Owner* analiza el progreso del proyecto en la Revisión del *Sprint* basándose en el *Product Backlog*, el *Developer Team* hace su seguimiento del *Sprint* diariamente (en la *Daily Scrum*) basándose en el *Sprint Backlog*.

C. Incremento

El incremento se define como la suma de los elementos del *Product Backlog* que han sido "terminados" durante el *sprint* y el resto de incrementos de los *sprints* previos.

Por lo tanto, un incremento es una parte entregable del total del producto, que puede ser utilizada y que funciona, sea o no liberada por el *Product Owner*. Que sea entregable y ejecutable no quiere decir que no se pueda modificar, pues, como comentamos en un punto anterior, el concepto de "terminado" se va evaluando en cada revisión a medida que aumenta el incremento.

En la *Figura 2-30* se puede visualizar cómo va creciendo el incremento en cada *sprint*, de forma que, en el último, ya se tiene el producto completo.

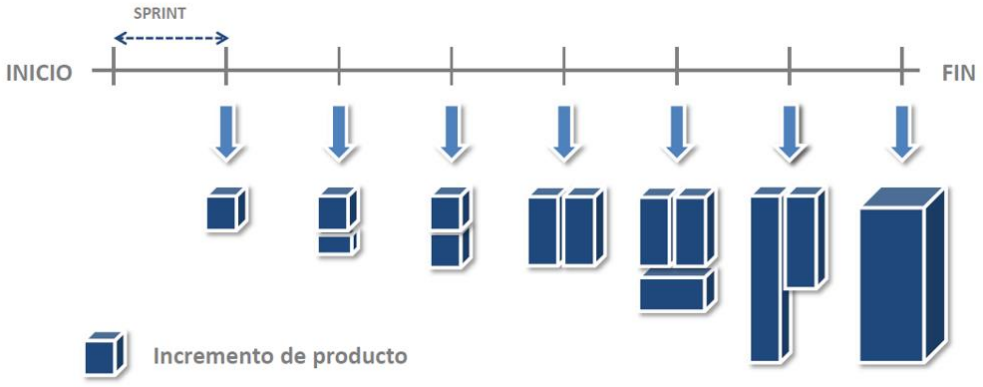

Figura 2-30. Incremento en cada Sprint de un proyecto basado en SCRUM.

Imagen tomada del material del curso recibido por CertiProf

Una vez definidos los **eventos** y los **artefactos** de Scrum, podemos entender todo el proceso visualizando el flujo representado en la *Figura 2-31*.

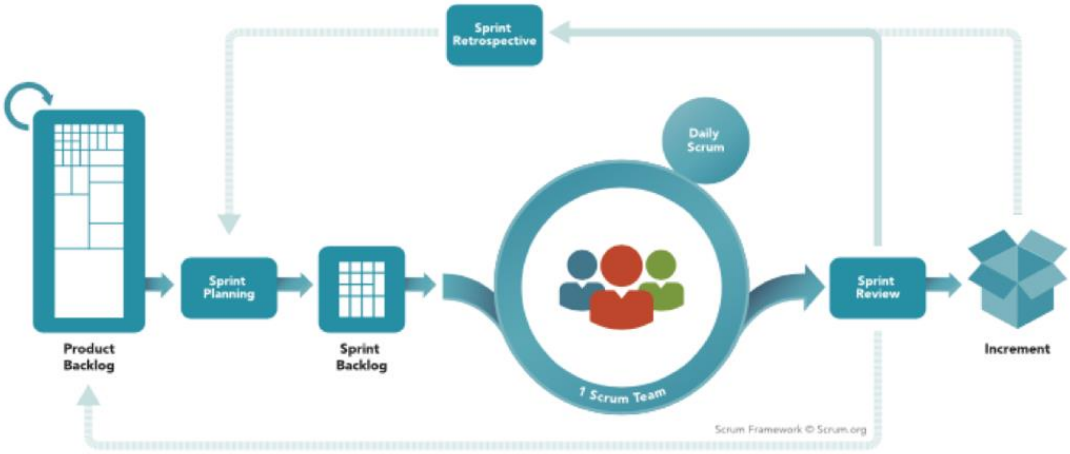

Figura 2-31. Diagrama de metodología SCRUM.

Imagen tomada de la web oficial de la guía (www.scrum.org)

A modo de resumen:

- El *Product Backlog* se va dividiendo en pilas más pequeñas (*Sprint Backlog*), definidas en cada *Sprint Planning*, antes de comenzar el *sprint*.
- La entrada de cada *sprint* es un *Sprint Backlog*, que se analiza diariamente, en la *Scrum Daily*, para conseguir el objetivo del *sprint*, que es un incremento de valor.
- Finalizado el *Sprint*, se analiza el incremento en la *Sprint Reiview* para considerarlo "terminado" o no y, además, se puede modificar o reordenar el *Product Backlog* en función de los pros y contras del *sprint*.
- Las mejoras y deficiencias se evalúan y recogen en la *Sprint Retrospective* para tenerlas en cuenta en el siguiente *Sprint Planning*, donde se vuelve a crear un *Sprint Backlog* que toma los siguientes elementos del *Product Backlog* más las mejoras o modificaciones aplicables al último incremento.

2.4.2 XP (eXtreme Programming)

XP es una metodología formulada por Kent Beck en 1999 [24] y aplicada por la empresa Chrysler, que promueve el trabajo en equipo, basándose en la realimentación continua del cliente con el equipo de desarrollo, poniendo énfasis en los valores de comunicación, simplicidad y coraje: comunicación plena de todos los involucrados, simplicidad en la implementación, y coraje para afrontar los cambios.

Como metodología ágil, se centra en potenciar las relaciones interpersonales.

Parte de la idea de implementar un producto sencillo que, si se debe mejorar, se refactoriza posteriormente. Es decir, se basa en versiones del producto.

El eje en torno al cual gira XP son las *historias de usuario*, una técnica para recoger, especificar y priorizar los requisitos. Son tablas en las que el cliente hace una descripción de las características o elementos del producto y su funcionalidad. La prioridad o valor la establece el cliente y el equipo de desarrollo le asigna un coste a la mismas, en función de las semanas de trabajo. Las historias de usuario no son estáticas, sino que se moldean y adaptan a medida que avanza el desarrollo, añadiendo nuevas historias de usuario, modificándolas, reemplazándolas o eliminándolas. Se utiliza el término *velocidad* como unidad de medida, definido como el número de historias realizadas por versión.

A continuación, se definen las 4 actividades de las que consta el ciclo de XP, sus 5 valores, roles y 13 buenas prácticas.

A. Actividades del ciclo de XP

1. **Planificación** → cliente y desarrolladores, tras entender cuál es el propósito y el objetivo del producto, escribe y priorizan las historias de usuarios a realizar para la próxima versión del producto.
2. **Diseño** → a partir de las historias de usuario, se realiza el diseño de la versión a implementar, siguiendo la práctica de diseño simple, usando las llamadas *Tarjetas CRC* (Class-Responsibility-Collaborator). Se realiza tanto antes como después de la codificación, realizándose una refactorización en función de las necesidades o problemas que surgen durante la codificación.
3. **Codificación** → el equipo de desarrollo, previo a la implementación, elaboran una serie de pruebas unitarias aplicadas a cada historia de usuario. Esto permite comenzar el desarrollo de forma más eficiente, teniendo un mayor conocimiento y eficacia a la hora de comenzar a desarrollar. En esta actividad, además, se lleva a cabo la refactorización o mejora en la estructura del código para mantener su sencillez y eficiencia, sin cambiar el comportamiento del mismo.
4. **Test** → pruebas definidas por el cliente para corroborar y aceptar la funcionalidad y características del producto. Además, se elaboran y ejecutan unas pruebas adicionales, llamadas de regresión, para comprobar que no se han producido errores al versionar el producto.

En la *Figura 2-32* se muestra cómo interaccionan las distintas actividades en el ciclo de desarrollo de XP.

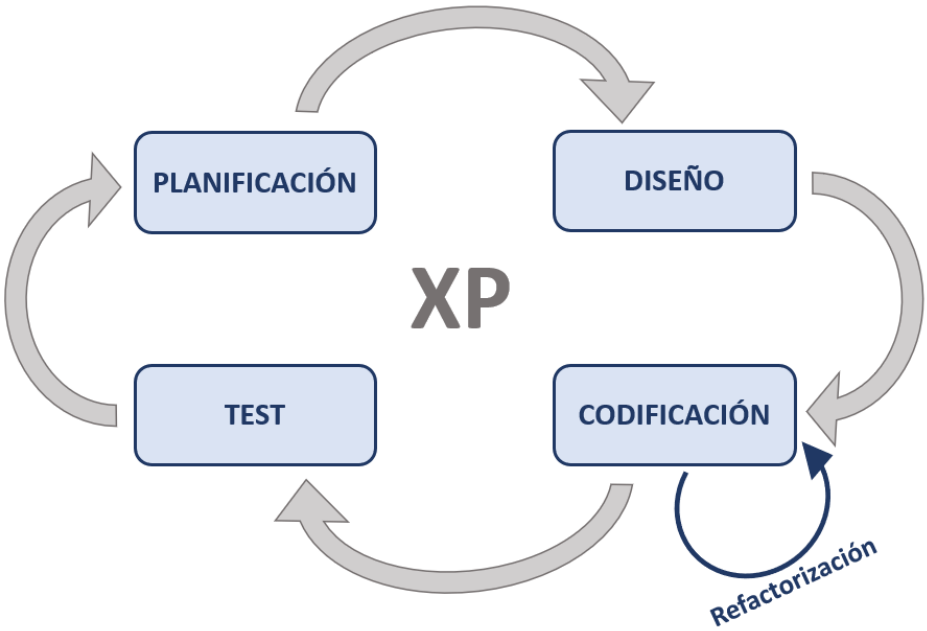

Figura 2-32. Actividades en el ciclo de desarrollo de XP.

Realización propia

B. Valores de XP

1. **Simplicidad** → prioridad a realizar algo más simple que funcione a algo más complejo que no funcione o no se use.
2. **Comunicación** → la metodología conlleva y aboga por una comunicación plena y continua.

3. **Realimentación** → cliente, equipo y usuarios finales, a través de la comunicación, retroalimentan el proceso y otorgan eficiencia y mayor valor al producto.
4. **Coraje** → capacidad y voluntad para afrontar cambios, así como correcta cooperación.
5. **Respeto** → la naturaleza iterativa de la metodología conlleva una convivencia continua con compañeros, cliente y usuarios, por lo que tiene que prevalecer el respeto hacia el trabajo y la opinión de los demás.

C. Roles de XP

- **Cliente**: crea las historias de usuarios, les da prioridad y decide cuáles se implementan en cada una de las iteraciones. También diseña las pruebas funcionales para la validación y aceptación de la entrega.
- **Programador**: realiza el desarrollo (código) y lleva a cabo las pruebas unitarias, previas a las de usuario.
- **Tester**: ejecuta las pruebas definidas por el cliente, y asesora a éste para definirlas. A través de las herramientas de soporte a pruebas, comunica los resultados al equipo.
- **Tracker**: realiza el seguimiento del proyecto, controlando el tiempo y las estimaciones. Además, propone cambios en base a la posibilidad de conseguir los objetivos en función del tiempo y los recursos disponibles.
- **Entrenador**: sirve de guía para asegurar la correcta aplicación de XP por parte del equipo.
- **Consultor**: experto en un área o tecnología que ayuda al equipo, aportando soluciones o enfoques.
- **Gestor**: coordina clientes y programadores.

D. Buenas prácticas en XP

1. **Equipo completo** → el equipo está formado por todas las personas involucradas y que tengan que ver con el proyecto, ya sea de la parte del cliente como de los proveedores.
2. **Planificación** → las historias de usuario se ejecutan y se realizan las miniversiones en base a una planificación, que se revisa de forma continua.
3. **Test del cliente** → el cliente diseña sus propias pruebas para validar las miniversiones, siempre apoyado en los desarrolladores.
4. **Versiones simplificadas** → se trata de las miniversiones, que deben ser lo suficientemente pequeñas para que haya entregas cada poco tiempo, pero a su vez deben ser independientes y ofrecer funcionalidad a nivel usuario.
5. **Integración continua** → hay un ejecutable sobre el que se van añadiendo las versiones, dotándolo cada vez de nuevas funcionalidades y haciéndolo más complejo. Cada vez que se "mejora" la integración, se realizan pruebas.
6. **Propiedad colectiva** → el desarrollo es transparente y accesible, de modo que cualquiera pueda verlo y probarlo. Así se permiten las pruebas automáticas.
7. **Codificación estandarizada** → el código debe ser homogéneo, para lo que se establece una guía de estilo, que sirve de referencia a las distintas personas que lo desarrollan.
8. **Metáforas** → para ayudar a los propios programadores y al cliente a la hora de acceder al código, se debe comentar de forma certera, usando frases y nombres que definan los distintos módulos y funcionalidades. De esta forma, se puede identificar y ubicar rápidamente cada desarrollo en el código, ya sea para versionarlo o para su mantenimiento o resolución de errores.

9. **Ritmo sostenible** → el objetivo es trabajar con un ritmo medio y regular, evitando días sin hacer nada y días de trabajo en exceso.
10. **Diseño simple** → optar por la forma más eficiente y fácil a la hora de desarrollar el código.
11. **Programación en parejas** → una buena práctica es que los programadores trabajen por parejas, cambiándose a menudo de compañero/a. De esta forma, todo el equipo de desarrollo está al tanto de todo.
12. **Desarrollo guiado por las pruebas automáticas** → ejecución frecuente de pruebas automatizadas.
13. **Refactorización** → mejorar el código, no su funcionalidad o comportamiento, a nivel estructural en cada versión para mantenerlo sencillo y eficiente.

Junto a las buenas prácticas, no está de más mencionar los 14 **principios**, que suponen el puente entre los valores y las prácticas: humanidad, beneficio mutuo, economía, autosimilitud, reflexión, mejora, diversidad, flujo, oportunidad, redundancia, calidad, fallo, pasos de bebé y aceptación de responsabilidad.

De forma resumida y para concluir, en la *Figura 2-33* se muestra el ciclo completo, con los elementos y las acciones llevadas a cabo en cada actividad:

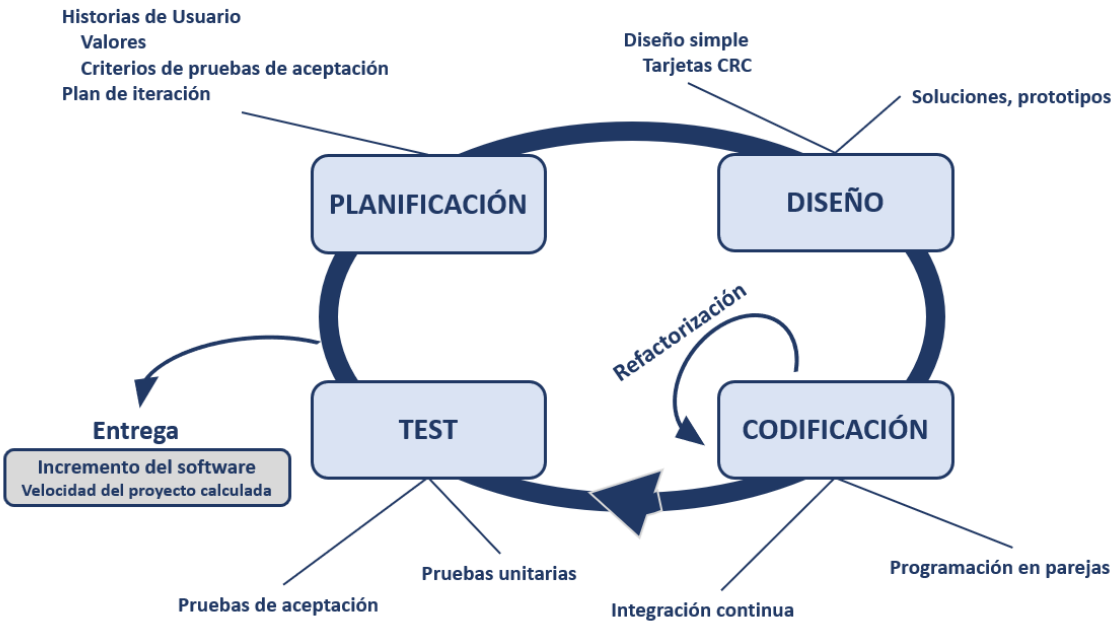

Figura 2-33. Ciclo de desarrollo de XP con acciones y buenas prácticas.

Realización propia

2.4.3 Crystal

Crystal es un conjunto de metodologías de peso liviano o *ágiles*, implementadas en los 90's por uno de los autores del Manifiesto Ágil, Alistair Cockburn [25]. Las nombró de esto modo, haciendo alusión a los minerales, que se caracterizan por dos factores: dureza y color. De forma análoga, los proyectos e Ingeniería de Software también se pueden caracterizar por dos factores:

1. **Dimensión** o **tamaño**: número de personas que conforman el proyecto.

   En función del número de integrantes del equipo, se hace una clasificación, asignando un color (uno de los factores de los minerales):

   - *Crystal Clear* → equipo de hasta 8 personas
   - *Crystal Yellow* → equipo entre 10 y 20 personas
   - *Crystal Orange* → equipo entre 20 y 50 personas
   - *Crystal Red* → equipo entre 50 y 100 personas
   - *Crystal Maroon* → equipo entre 100 y 200 personas
   - *Crystal Diamond* → equipo entre 200 y 500 personas
   - *Crystal Saphire* → equipo de más de 500 personas

2. **Criticidad**: medida de las consecuencias de los errores.

   Se mide de menos o más grave, es decir, se mide en base a la pérdida de usabilidad o confort, pérdidas económicas y pérdidas de recursos humanos.

   - **C** → pérdida de confort por fallo del sistema
   - **D** → pérdida de dinero discrecional, que es el nuestro
   - **E** → pérdida de dinero esencial, que es del que no podemos disponer libremente
   - **L** → pérdida de vidas por fallos del sistema

En la *Tabla 2-4* se recoge la clasificación y nombramiento de las distintas metodologías Crystal en base a los dos factores comentados:

		METODOLOGÍAS CRYSTAL					
		Clear	**Yellow**	**Orange**	**Red**	**Maroon**	**Diamond**
Criticidad	L	L6	L20	L40	L80	L200	L500
	E	E6	E20	E40	E80	E200	E500
	D	D6	D20	D40	D80	D200	D500
	C	C6	C20	C40	C80	C200	C500
		1-6	7-20	21-40	41-80	81-200	201-500
		Nº de personas en el equipo					

Tabla 2-4. Metodologías Crystal.

Realización propia

El objetivo de su creador es romper con la idea tradicional en la gestión de proyectos de que hay unas metodologías mejores que otras, y aplicables en todos los casos. Con las metodologías Crystal expone que no hay una metodología única aplicable, sino que, en función del tipo de proyecto, habrá siempre una metodología óptima que aporte maniobrabilidad a éste. Como se puede intuir por la tabla anterior, las parámetros o factores para elegir una metodología, según esta publicación, son:

- Tamaño del equipo

- Distribución del equipo
- Criticidad del proyecto
- Prioridades

Por este motivo, se definen varias metodologías, que tienen una base común. Cada una de las variantes tiene definidos sus roles, modelos de procesos, productos de trabajo y buenas prácticas.

En resumen, podemos definir Crystal como un conjunto de metodologías o procesos ágiles, empíricamente aplicables a distintos tipos de proyectos, clasificados en función de los parámetros anteriormente mostrados. Todas las variantes tienen una base común, la de la familia Crystal.

Esta base se apoya en 7 **principios** o estamentos:

1. **Frecuencia en las entregas** → puesto que se trata de una familia de metodologías ágiles, iterativas e incrementales, debe haber entregas cada cierto tiempo, que no está fijado (como en Scrum), pues dependerá del tipo de proyecto.
2. **Mejora reflexiva** → mejora continua, apoyada en las iteraciones, que permiten ajustar y refinar el proyecto.
3. **Comunicación osmótica** → ubicación física del equipo de forma conjunta para favorecer y permitir la comunicación directa y de forma fluida.
4. **Seguridad personal** → consideración de la opinión de todos los involucrados, invitando a éstos a no tener ningún tipo de obstáculo para expresarse.
5. **Fácil acceso al cliente** → a diferencia de SCRUM, el cliente no tiene que estar siempre con el resto del equipo, sino que se convocan reuniones semanales para tener un contacto periódico.
6. **Entorno técnico** → integración continua, gestión de la configuración y realización de pruebas automatizadas.

Las fases que tienen en común todas las metodologías de la familia Crystal son las siguientes:

- **Puesta en escena** → planificación para el siguiente entregable o incremento, seleccionando los requerimientos a abarcar por parte del equipo.
- **Revisiones** → un mismo incremento o entregable tiene varias iteraciones, que requieren de actividades, demostraciones y resumen de objetivos de éste.
- **Monitoreo** → medición y seguimiento de los progresos en las distintas entregas, marcando objetivo o hitos a cubrir en cada fase.
- **Paralelismo y flujo** → estabilización del monitoreo, que permite pasar a la siguiente fase.
- **Estrategia de diversidad holística** → división de los grandes equipos funciones en equipos multifuncionales.
- **Técnica de puesta a punto de la metodología** → fijación o modificación del proceso de desarrollo, elaborando una metodología específica, apoyada en formaciones, talleres y entrevistas.
- **Puntos de vista de usuario** → depende del "color" de la metodología Crystal elegida, pero se suele establecer en la opinión de 2 usuarios por versión y 3 revisiones del cliente por cada iteración.

Con respecto a los roles, éstos dependen fuertemente del color de la metodología, pues marcan el número de integrantes del proyecto y, en función del tamaño y la complejidad de éste, serán necesarios o no ciertos roles. Los más comunes son:

- **Patrocinador Ejecutivo**: provee del dinero para empezar con el proyecto.

- **Diseñador Principal**: encabeza el equipo de desarrollo y sirve de referencia a los demás, modelando el resultado.
- **Usuario Experto**: usuario que prueba los incrementos y al que consulta el equipo de desarrollo
- **Diseñador Programador**: produce el código, de acuerdo con el diseñador principal
- **Coordinador**: controla el avance del proyecto, verificando su estado, manteniendo el orden y facilitando la comunicación y dudas.
- **Experto en Negocios**: verifica el estado del negocio, definiendo las estrategias y estando en contacto continuo con el equipo de desarrollo para hacer seguimiento de la ejecución.
- **Verificador**: figura que se alterna en los distintos miembros del equipo, que verifica y pone a prueba el sistema, realizando reportes del estado del proyecto.
- **Escritor**: alternado entre los distintos miembros del equipo, redacta el manual de usuario.

2.4.4 AM (Agile Modeling)

AM es una metodología publicada inicialmente en el año 2000 por Scott Ambler [26], y enfocada en el modelado de sistemas y la realización de documentación de proyectos de software, basándose en un conjunto de buenas prácticas.

Se puede definir como un conjunto de recomendaciones, no como un proceso prescriptivo, pues no recoge la definición detallada de los procesos concretos para implementar un modelo. Más bien es un conjunto de valores, principios y buenas prácticas a aplicar.

AM no comprende todas las fases del proyecto, sino que, como se intuye de su definición, pone el foco en el modelado y documentación. Deja fuera la gestión de la totalidad de un proyecto, la programación de las actividades, las pruebas y resto de fases, como implementación o ejecución.

Se usa como complemento a otras metodologías de peso liviano o ágiles, como las anteriores vistas, SCRUM o XP.

Es importante nombrar y describir AM, pero no se va a detallar al nivel de las anteriores, pues, teniendo como referencia el informe de "State of Agile" del año 2020 usado al inicio de la sección, no es usada ni por el 1% de los proyectos tecnológicos.

Esta metodología se basa en los mismos 5 **valores** que XP porque, como se ha comentado, complementa a ésta:

1. Comunicación
2. Simplicidad
3. Realimentación
4. Coraje
5. Humildad o respeto

Los 11 **principios** de AM son los siguientes:

1. Modelado en base a un propósito
2. Maximización del entorno de la inversión (ROI) de los involucrados
3. Viaje ligero
4. Múltiples modelos
5. Realimentación rápida

6. Asumir la simplicidad
7. Abrazar el cambio
8. Incrementos pequeños
9. Trabajo de calidad
10. Hacer software como primer objetivo
11. Tener en mente el siguiente paso a realizar como segundo objetivo

Con respecto a sus **buenas prácticas**, se muestran a continuación, a través de la *Figura 2-34*, obtenida de la web oficial [49] y diseñada por su creador, Scott Ambler:

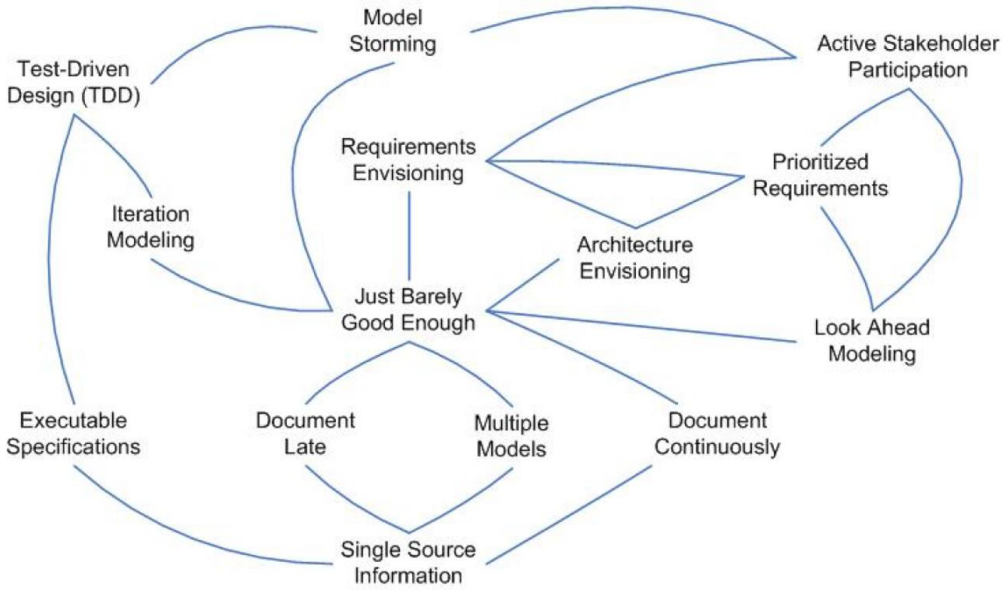

Figura 2-34. Esquema con las buenas prácticas de AM.

Imagen tomada de la url: http://www.agilemodeling.com/essays/introductionToAM.htm

2.4.5 ASD (Adaptative Software Development)

ASD es una metodología que tiene sus orígenes en 1990, planteada y propuesta por Jim Highmith y Sam Bayer, que fue publicada posteriormente, en el año 2000 [27].

ASD plantea una filosofía en la que predomina la colaboración entre personas y la auto-organización de los equipos sobre cualquier proceso o práctica. Asimila y entiende el cambio como una forma de mejora y adaptación en vez de luchar contra él y evitarlo.

Por tanto, podemos definirlo como una metodología que tiene su base en la adaptación continua para proyectos con circunstancias cambiantes.

Su ciclo de vida se compone de 3 **fases**:

1. **Especulación** → fase inicial, que marca el inicio del proyecto, en la que se lleva a cabo la definición y establecimiento de los objetivos, las limitaciones y los riesgos. En esta fase, se realiza una estimación temporal del proyecto, definiendo el número de iteraciones y la duración de éstas.

   A su vez, cada iteración requiere de una definición de sus objetivos y funcionalidades. En función del resultado de cada iteración, se realiza un análisis y se vuelve a estimar y recalcular tanto el tiempo como los objetivos y riesgos.

2. **Colaboración** → es la fase en la que se desarrolla y se gestiona el producto. Se puede considerar la etapa de ejecución del proyecto. Además, se revisan en detalle los requisitos y se establece la forma y organización del equipo, teniendo en cuenta las habilidades de cada miembro, para llevar a cabo el desarrollo.

3. **Aprendizaje** → fase clave, al tratarse de la revisión continua del proyecto. Determina y pone en valor la eficacia del equipo, contrastando con el resultado obtenido hasta el momento. Es el núcleo de la filosofía de *ASD*, pues representa el aprendizaje continuo, la naturaleza iterativa de la metodología.

   En esta fase se hace una evaluación y revisión de:

   - Calidad del producto:
     - Por parte del equipo de desarrollo
     - Por parte del cliente
   - Gestión del rendimiento y evaluación del aprendizaje
   - Situación del proyecto y planificación de la siguiente iteración

Para representar el ciclo de vida de ASD y entender su funcionamiento se ha diseñado la *Figura 2-35*.

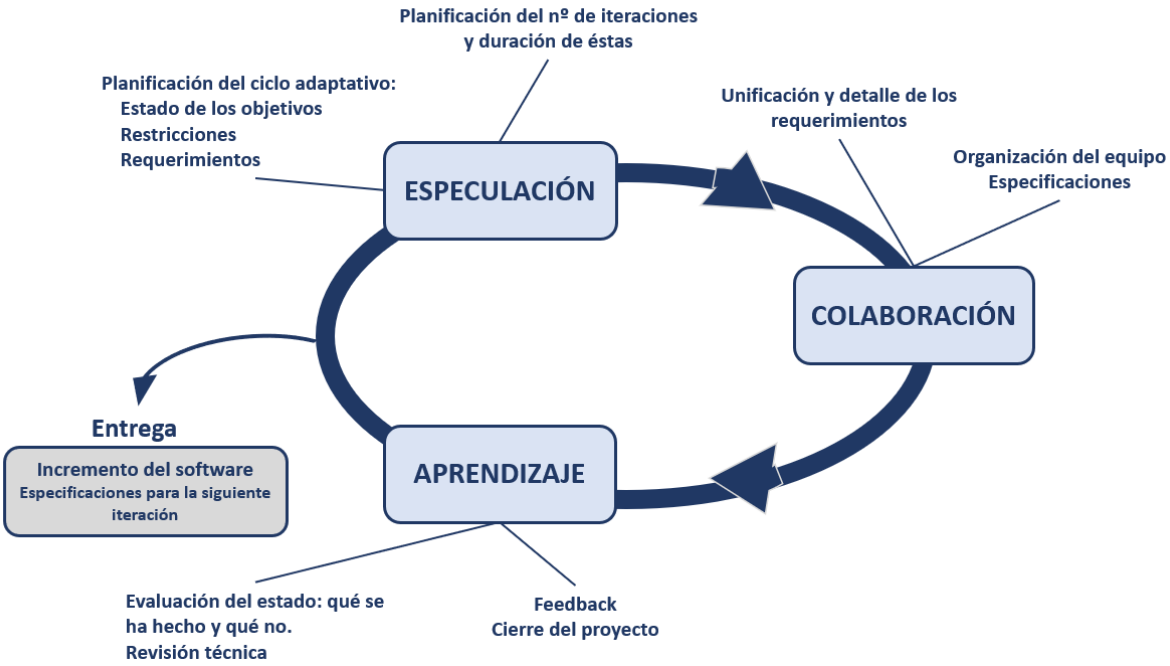

Figura 2-35. Ciclo de vida de ASD.

Realización propia

2.4.6 AUP (Agile Unified Process)

AUP es una metodología que se basa, haciendo una simplificación, en una metodología implementada previamente por IBM, llamada *Rational Unified Process* o *RUP*. Toma de ésta el uso de técnicas ágiles, como por ejemplo *TDD* (Test Driven Development), el modelado ágil, gestión de cambios y la refactorización en el desarrollo de software.

AUP fue implementada y publicada en 2002 por Scott Ambler (quien también creó AM) [28] y su última versión se lanzó en 2012, con el nombre de *Disciplined Agile Delivery* o *DAD*.

AUP se basa en 6 principios, 7 disciplinas, 4 fases y 2 tipos de iteraciones, que dan forma y base al ciclo de vida de esta metodología.

Los 6 **principios** de AUP son:

1. Los empleados saben lo que están haciendo
2. Simplicidad
3. Agilidad
4. Priorizar las actividades de alto valor
5. Independencia de las herramientas
6. Adaptación de la metodología a las necesidades del proyecto

Con estos principios como base y filosofía, AUP ofrece un ciclo de vida iterativo en el que se distinguen 2 tipos de **iteraciones**:

1. Iteraciones de **desarrollo** → incrementos de menor tamaño que van al área de pruebas.
2. Iteraciones de versiones de **producción** → incrementos probados y validados de mayor tamaño, que van al área de producción.

Las 7 **disciplinas** que conforman el ciclo de vida de AUP son:

1. **Modelado** → organización con el negocio, objetivos y problemas del proyecto, identificación y propuesta de una solución válida.
2. **Implementación** → transformación del modelo en código ejecutable y realización de pruebas.
3. **Test** → evaluación del estado sobre los objetivos, con el fin de garantizar la calidad. Se realiza una búsqueda de errores, se verifica que el sistema funciona como se diseñó, y se hace una verificación del cumplimiento de los requerimientos.
4. **Despliegue** → planificación de la entrega del producto y del despliegue en producción.
5. **Gestión de la configuración** → control de las versiones o entregables del proyecto y administración de cambios en éstos.
6. **Gestión del proyecto** → dirección de las actividades y recursos del proyecto, como es la gestión del riesgo, de los recursos humanos, la coordinación con todos los interesados del proyecto, la comunicación entre equipos.
7. **Entorno** → acceso por parte del equipo a la documentación, estándares, guías, herramientas y manuales.

Estas 7 disciplinas se realizan en las 4 **fases** que se ejecutan en serie durante el ciclo de vida del proyecto, como se muestran en la *Figura 2-36* y se describen a continuación:

1. **Inicio** → definición de requerimientos, alcance del proyecto, infraestructuras, financiación y aceptación de todos los involucrados.
2. **Elaboración** → prueba de la arquitectura del sistema.
3. **Construcción** → elaboración incremental del software, que cubra los requerimientos, y que funcione en cada entrega.
4. **Transición** → validación del sistema y despliegue en producción.

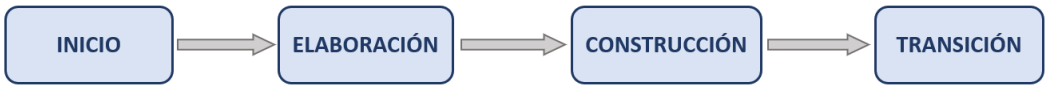

Figura 2-36. Fases del ciclo de vida de AUP.

Realización propia

Por último, para entender la relación entre disciplinas y fases, se muestra la *Figura 2-37*, tomada de la web oficial de la metodología, que se usa para indicar el esfuerzo en el tiempo. Las disciplinas se ejecutan de forma iterativa, definiendo las actividades, y las disciplinas se implementan de forma serial. Es decir, las disciplinas o tareas se organizan en fases e iteraciones.

En las cuatro fases se realizan iteraciones con un número variable según el proyecto. Se puede observar en la *Figura 2-37* cómo va variando el esfuerzo de cada disciplina en función de la fase en la que se encuentra el proyecto. Además, se puede comprobar cómo las primeras iteraciones se centran en el modelado y el entorno, la central en implementación y test, y la final en el despliegue. La gestión va teniendo lugar en todas las fases.

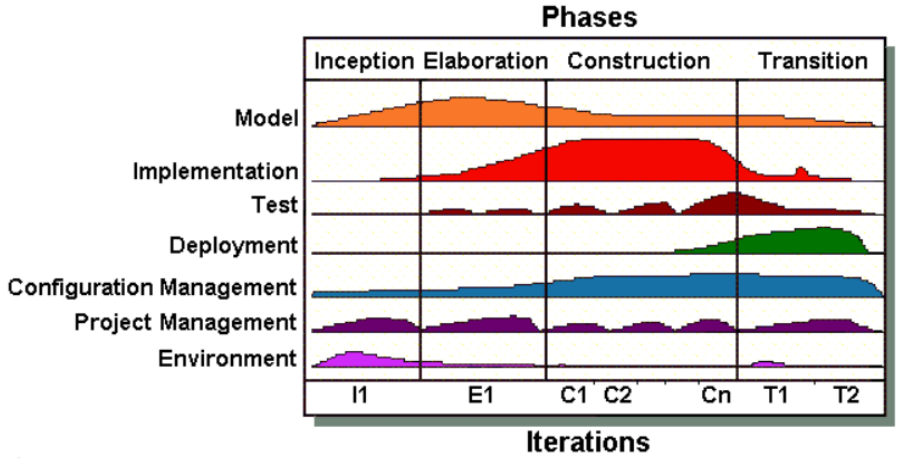

Figura 2-37. Diagrama de esfuerzo en el tiempo en AUP, en función de las fases y disciplinas.

Imagen tomada de la web oficial de AUP, url: http://www.ambysoft.com/unifiedprocess/agileUP.html

2.4.7 DSDM (Dynamic Systems Development Method)

DSDM es un método de desarrollo, creado por el Consorcio *DSDM Consortium* y publicado por primera vez en 1995. Es importante hacer una parada y definir "consorcio", pues es el único método del TFG definido por este organismo público.

Un consorcio se define como organización no lucrativa y proveedora independiente que posee, administra, gestiona y publica de forma gratuita un servicio. El objetivo del consorcio con DSDM era crear y poner al servicio del dominio público una metodología independiente de las herramientas. Así nace DSDM.

La versión más actual fue publicada en 2014 y se titula "DSDM Agile Project Framework" [29]. Es integrable con otras metodologías, tanto ágiles como tradicionales, como XP, PMBOK y PRINCE2, entre otras, complementando a éstas en lo referido a la comunicación con el cliente.

Podemos definir DSDM como un método que ofrece un *framework* o entorno genérico de escritura de código, para el desarrollo ágil en Ingeniería de Software. Su foco es la implicación del usuario de forma continua, permitiendo la adaptación continua a los cambios de requerimientos, ofreciendo un desarrollo de forma iterativa y creciente. De esta forma, se consigue disminuir el impacto y los costes ante cambios en necesidades o plazos.

Aunque DSDM esté dentro de las metodologías ágiles por ser iterativa y apoyarse en la implicación continua con el cliente, se aproxima mucho a las tradicionales. Por este motivo, se usa para complementar a las tradicionales, añadiéndole el dinamismo que ofrecen las ágiles.

Como prácticamente todas las metodologías, se basa en una serie de principios, su ciclo de vida se compone de varias fases, y el equipo se organiza en roles. A continuación, se enumeran y definen estos elementos de la metodología DSDM.

Los 9 **principios** en los que se apoya DSDM son los siguientes:

1. Involucración del cliente de forma activa.
2. Capacidad y poder del equipo para tomar decisiones.
3. Entregas frecuentes.
4. Funcionalidades críticas sobre las accesorias; es decir, el criterio principal de aceptación de un entregable es la satisfacción de las necesidades actuales del cliente.
5. Desarrollo iterativo e incremental.
6. Posibilidad de que los cambios en el desarrollo sean reversibles.
7. Establecimiento y fijación como línea base de los requerimientos y el alcance a alto nivel, antes de comenzar el desarrollo del proyecto. Este principio es el que aproxima DSDM a las tradicionales.
8. Realización de pruebas durante todo el ciclo de vida del proyecto.
9. Comunicación y cooperación de todos los interesados e involucrados en el proyecto.

Con estos principios como base, DSDM presenta 3 **fases**:

1. **Pre-proyecto** → identificación de los proyectos factibles y viables, obtención de la financiación para iniciarlos y aseguramiento del compromiso. Se trata del anteproyecto, necesario para asegurar su ejecución y para evitar errores o problemas en fases posteriores. Es la fase que acerca DSDM a las metodologías tradicionales, pues realiza una planificación y aseguramiento robustos antes de empezar con el ciclo de vida de desarrollo del proyecto.
2. **Ciclo de vida del proyecto** → es la fase de ejecución del proyecto como tal, compuesta por 5 etapas, de las cuales las 2 primeras son secuenciales, y las 3 últimas son iterativas e incrementales:
    I. **Estudio de la viabilidad** → análisis de la capacidad de adaptación de la metodología al proyecto.

II. **Estudio del negocio** → comunicación e involucración del cliente para asentar los objetivos y definir los requisitos funcionales, antes de comenzar con el desarrollo.

III. **Iteración del modelo funcional** → producción de prototipos incrementales y evolutivos para poder mostrar al cliente las funcionalidades a cubrir, de forma que se puedan especificar requisitos más concretos y realizar y documentar el análisis completo.

IV. **Diseño e iteración de la estructura** → complementa a la anterior, revisando los prototipos funcionales y diseñando el sistema operacionalmente.

V. **Implantación** → validación del sistema por parte del cliente y despliegue en producción.

3. **Post-proyecto** → aseguramiento de la eficiencia del sistema implantando, a través de su mantenimiento, correcciones y mejoras. Es una etapa que envía de nuevo al inicio del ciclo, ya sea para mantenimiento o mejoras, pues se trataría de un desarrollo continuo. Se podría considerar la etapa previa al inicio de otro desarrollo. No se debe interpretar como una etapa de cierre, pues estamos en un modelo incremental e iterativo. Sería más bien, una nueva etapa de análisis.

A través de la *Figura 2-38*, podemos entender cómo se producen las iteraciones en DSDM:

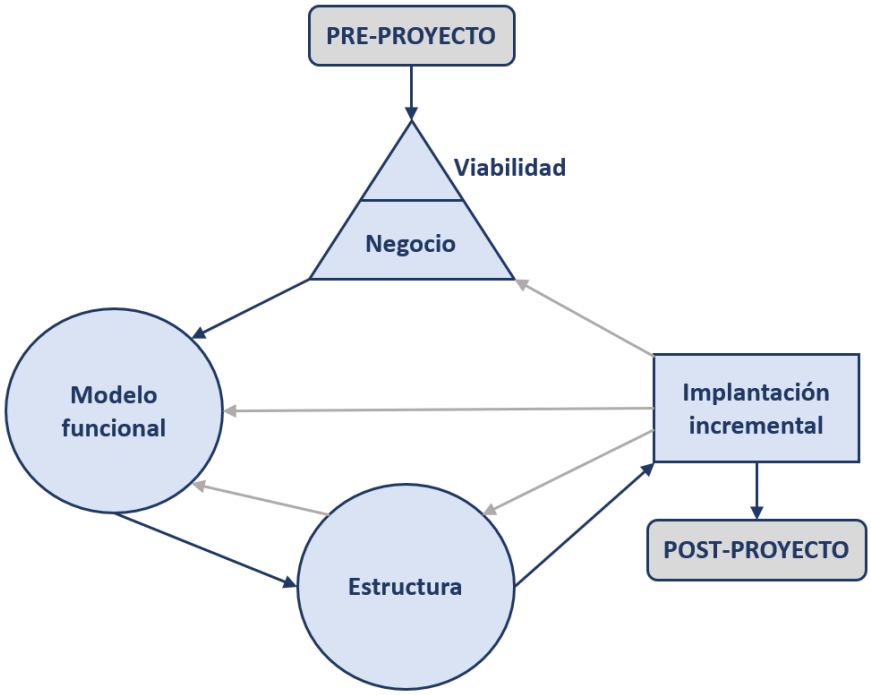

Figura 2-38. Ciclo de vida de DSDM.

Realización propia

Con respecto a los **roles**, DSDM propone varios, en función de las características del proyecto y el tamaño del equipo, cada uno de ellos con una responsabilidad propia. Se recogen en la figura *Figura 2-39*.

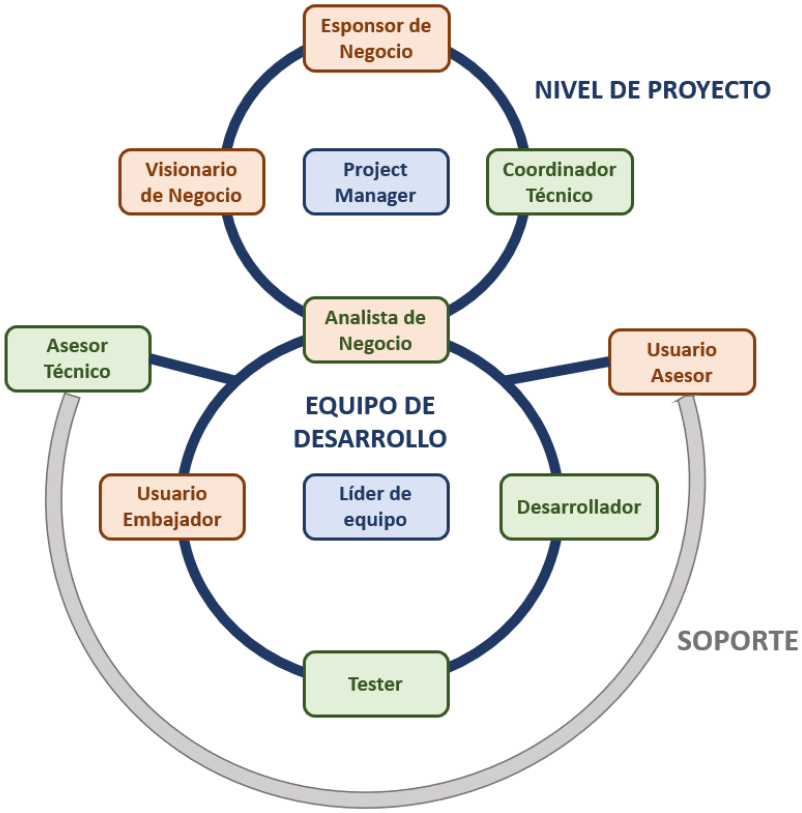

Figura 2-39. Roles en DSDM.

Realización propia (basada en la propuesta de DSDM Consortium)

2.4.8 FDD (Feature Drive Development)

FDD es una metodología implementada en 1999 en el libro "Java Modeling in Color with UML" de Peter Coad y Jeff de Luca. Como puede anticipar el título del libro, se presentó como un método basado en procesos para software orientado a objetos.

Tres años más tarde, en 2002, se realizó la publicación de su mejora y ampliación en el libro "A Practical Guide to Feature Driven Development", con la autoría de Stephen Palmer y John Felsing [29]. Esta publicación recoge la descripción de procesos ágiles con capacidad de adaptación para su aplicación a proyectos de tamaños medios y grandes.

Compartiendo filosofía con el resto de las metodologías ágiles, FDD se fundamenta en lo siguiente:

- Priorizar la comunicación entre todos los involucrados del proyecto.
- Desarrollo incremental para la gestión de problemas y complejidades del proyecto, basándose en descomposición de funcionalidades.
- Informar de los detalles técnicos a través de la continua comunicación interpersonal, gráficos y documentación.

FDD introduce y se apoya en la definición de "característica", que es entendida como "función que aporta valor al cliente y que puede ser implementada en dos semanas o menos" [30].

Con este concepto y elemento clave, FDD ofrece los siguientes **beneficios** y mejoras:

- Entendimiento y comprensión de cada característica por parte de los usuarios, pues son pequeños

bloques o funcionalidades concretas entregables. Permite una descripción de cada una con mayor facilidad y sencillez, entendiendo la relación e integración entre ellas, evitando ambigüedades y errores.

- Las características se pueden organizar de forma jerárquica en grupos relacionados directamente con el cliente.
- Puesto que en FDD nos referimos a un incremento como "característica", el equipo va a desarrollar características ejecutables y operativas cada intervalo de tiempo estimado, normalmente dos semanas.
- El uso de características como funcionalidades pequeñas permite revisar e inspeccionar se diseño y código de forma más eficiente (se divide algo complejo en pequeños bloques más sencillos).
- Como contra o desventaja, que cada característica supone una iteración en el ciclo de desarrollo incremental, lo que conlleva costes en la integración de cada incremento (análisis, rediseño, refactorización, pruebas, despliegue).

FDD tiene un ciclo de desarrollo incremental de 5 etapas o fases, en el que cada iteración, a su vez, consta de otras 2 fases: diseño y construcción.

Las 5 **etapas** de esta metodología son las siguientes:

1. **Desarrollo del modelo global** → A partir de las necesidades planteadas por el cliente, se crea un modelo previo que recoge de manera funcional y más generalizada los requisitos, objetivos, contexto y visión del proyecto. Este modelo se subdivide en áreas, que son analizadas de forma detallada. Con todas las áreas se monta un diagrama de clases, que relaciona y ubica éstas dentro del modelo.
2. **Construcción de la lista de características** → Creación de una lista, que es evaluada y aprobada por el cliente, con todas las funcionalidades del sistema. A su vez, cada funcionalidad se divide en funcionalidades más pequeñas, que permiten su entendimiento y detalle.
3. **Planificación** → Una vez elaborada y aprobada la lista de funcionalidades, se ordenan en función de su prioridad, dependencias e importancia, y se asignan a los responsables de programación.
4. **Diseño** → A partir de un conjunto de funcionalidades de la lista, se inicia un proceso iterativo en el que se realiza una inspección de diseño, codificación, pruebas e integración de la funcionalidad a implementar. En esta etapa se decide qué funcionalidad se implementa en cada iteración y qué conjunto de características abarca cada una.
5. **Construcción** → Desarrollo o implementación del diseño.

Se facilita la *Figura 2-40Figura 2-40. Ciclo de desarrollo de FDD.* para entender cómo interactúan las distintas etapas del ciclo de desarrollo en FDD.

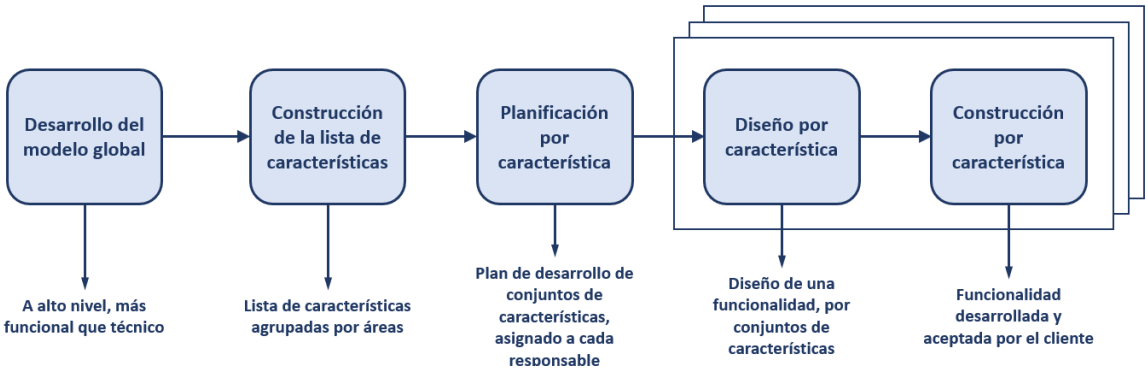

Figura 2-40. Ciclo de desarrollo de FDD.

Realización propia

Por último, hacer hincapié en la focalización de FDD en el aseguramiento de la calidad a través de prácticas en su ciclo iterativo, como son las inspecciones y auditorías de código, análisis basados en métricas y patrones, diseño y construcción.

2.4.9 LSD (Lean Software Development)

LSD no es una metodología como tal, sino que es una forma de pensar. Su publicación se centra en su filosofía o principios y ofrece, además, herramientas, instrumentos y comparativas con otras metodologías ágiles.

Por este motivo, vamos a describir la filosofía de esta propuesta, que tienen como objetivo ofrecer una síntesis de principios en la construcción de sistemas software. Prioriza los principios sobre las prácticas como forma de mejora en la calidad y desarrollo del producto.

El origen de esta metodología es el libro "Lean Software Development", publicado en 2003 por Mary y Tom Poppendieck [31], que propone LSD como una corriente de pensamiento, no como una herramienta en sí. Esta corriente se basa en los principios de fabricación de Toyota (TPS) propuestos en 1956 por Taiichi Ohno. Dichos principios se dieron a conocer como principios *Lean*.

Podemos definir LSD a partir de sus características fundamentales:

- Es una filosofía y una forma de pensar.
- Elimina los desperdicios o todo aquello que no sea de valor para el cliente, tras el análisis de los procesos de producción.
- Es la agrupación conceptos y técnicas para aumentar la productividad y mejorar la calidad.
- Los principios de las metodologías ágiles están basados en *Lean*.

Por lo tanto, se trata de la base del pensamiento ágil, que sirvió de referencia para definir los principios de cada metodología. A partir de estos principios, cada metodología construye una serie de herramientas o buenas prácticas para alcanzarlos.

Los 7 **principios** en los que se basa LSD son los siguientes:

1. **Eliminar desperdicios** → Desperdicio es todo aquello que no produce valor en el cliente. Como el objetivo es mejorar la calidad, todo desperdicio debe ser eliminado. Dicho de otra forma, sólo se realiza lo necesario y de forma eficiente. Por ejemplo, no se van a añadir funcionalidades extras que no cubran requisitos concretos, no se van a aplazar entregas, no se van a aceptar requerimientos ambiguos, se va a simplificar la burocracia en los procesos de comunicación.

   En concreto, se distinguen 8 tipos de desperdicios:

   - Defectos de producción: errores en el desarrollo y el servicio, debidos a una mala supervisión, baja calidad, formación del desarrollador.
   - Sobreproducción: producción no ajustada a la demanda, que desarrolla más de lo necesario.
   - Exceso de inventario: material, documentación o desarrollos no utilizados.
   - Esperas: tiempos muertos debido a una mala planificación o comunicación.
   - Transporte: traslado o intercambio de documentación, materiales o personas.

- Movimientos innecesarios: relacionado con el anterior, hace referencia al desplazamiento de personas o comunicaciones innecesarias que restan tiempo.
- Sobreprocesos: redundancia en procesos, por falta de optimización, control y verificaciones.
- Desaprovechamiento del talento humano: impedir que los trabajadores aporten conocimiento y tomen decisiones en base a su formación y experiencia.

2. **Crear conocimiento** → dedicar tiempo y esfuerzo a la comprensión de las necesidades del cliente como aspecto principal, pues es lo que aporta valor y eficacia al desarrollo. De lo contrario, podría implementarse un producto inútil, sin valor. Por ello, hay que considerar el proceso de desarrollo como una fase de aprendizaje, en la que hay que trabajar juntamente con el cliente, entendiendo sus peticiones y aportando valor para mejorarlas. Es necesaria y vital la comunicación iterativa incremental con el cliente.
3. **Diferir el compromiso** → retrasar y asegurar la aceptación de los requerimientos lo que sea necesario, hasta que se elimine toda incertidumbre y se tengan completamente claros todos los objetivos. Las decisiones deben basarse en hechos y no en suposiciones. Un buen análisis y verificación de requerimientos abre camino a un desarrollo efectivo. No se debe comenzar un desarrollo con requerimientos incompletos o ambiguos.
4. **Entregar rápidamente** → entrega de código ejecutable cada corto periodo de tiempo, con la calidad suficiente. Es básico en el ciclo iterativo e incremental del proyecto, pues se trabaja sobre un primer entregable, al que se le añade contenido y valor, contrastado con el cliente.
5. **Potenciar el equipo** → tener en consideración el valor humano y profesional del equipo a la hora de tomar decisiones. Se propone un cambio en la posición del equipo a la hora de decidir: los directivos deben escuchar y considerar a los desarrolladores para poder tomar una decisión de forma más certera. Además, hay que eliminar la idea de que una persona es un recurso que recibe órdenes y ejecuta. El equipo necesita recibir motivación y responsabilidad, con unos objetivos marcados para que pueda seleccionar sus compromisos y prioridades.
6. **Construir con calidad** → añadir calidad no sólo al producto, sino a todo el proceso del proyecto. Para ello, es necesaria la transparencia y claridad a la hora de trabajar, teniendo clara la función y rol a desempeñar, los objetivos y alcance del proyecto. Es esencial un control de la calidad, basado en pruebas automatizadas y la refactorización del código, eliminando redundancias y haciéndolo más eficiente.
7. **Optimizar el todo** → considerar el proyecto como un todo en lugar de dar valor a cada etapa de forma independiente. En todo momento, conocer dónde estamos y hacia dónde queremos ir, haciendo una optimización global, en lugar de optimizar cada etapa por separado, pues podría provocar trabajos "parcialmente terminados" o elementos no integrables con el resto.

2.4.10 Kanban y Scrumban

Kanban es un método creado y propuesto en 2008 por David J. Anderson, basado en la metodología de producción industrial del mismo nombre, e inspirado en los sistemas de Toyota (TPS) y *Lean*.

Se trata de un proceso gradual y evolutivo, totalmente dinámico y abierto al cambio, en el que el trabajo se divide en partes. El trabajo se plasma en un tablero, en el que hay tantas columnas como estados o partes del proyecto. Se usa una tarjeta o *posit* asociada a cada tarea, en la que se recoge cierta información. Cada tarjeta debe, por lo tanto, pasar por todas las columnas del tablero o pizarra, pues así se plasma su recorrido por el ciclo de desarrollo del proyecto.

Kanban se puede considerar una herramienta o técnica más que una metodología en sí [32].

Sirve para hacer una asignación clara de tareas y monitorizar el avance de éstas, priorizando. Su objetivo es tener una foto viva y actualizada, en todo momento, del estado del proyecto en función a las tareas.

Se usan distintos colores, por ejemplo, para distintos usuarios, prioridades o bloques.

Un ejemplo sería la *Figura 2-41*, en la que cada columna es un estado del proyecto y cada *posit* una tarea.

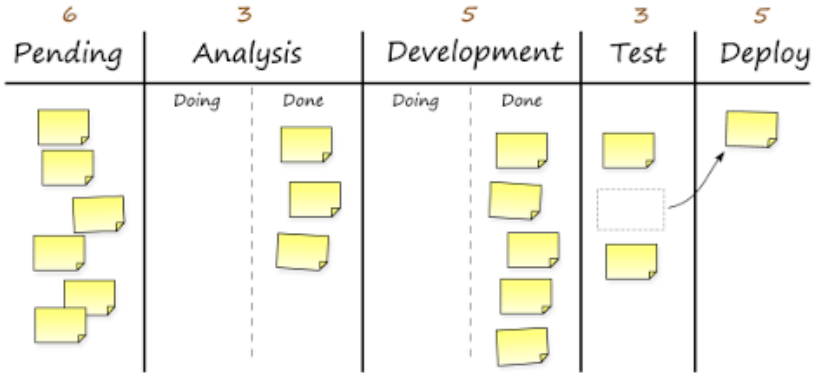

Figura 2-41. Tablero Kanban.

Imagen tomada de la url: https://qanewsblog.com/2016/05/11/kanban-o-como-ser-flexibles-pero-poniendo-limites/

Una peculiaridad de Kanban es que no considera roles y mantiene a todo el equipo en el mismo nivel jerárquico para potenciar la capacidad de cambio y evitar bloqueos propios de la burocracia y asignaciones.

Tampoco requiere reuniones diarias de seguimiento, ya que una vez que cada miembro del equipo asume una tarea, la va actualizando sobre el propio tablero o pizarra.

Por estas características, es considerado una técnica o herramienta de señalización y monitorización, no una metodología. Es el motivo por el que se usa como complemento o herramienta en metodologías ágiles como SCRUM para monitorizar el proyecto a través de la visualización del estado de las distintas tareas.

Así nace la metodología Scrumban, que mezcla y toma ciertas características de SCRUM y de Kanban, dando lugar a una metodología de fusión o combinada. Hay aspectos de ambas que no permiten una combinación eficaz, ya que SCRUM es muy estricta en cuanto a estructuración, todo lo contrario a Kanban.

De *Kanban* toma las siguientes características:

- Flexibilidad en la planificación y ejecución de tareas (se hace lo necesario cuando sea necesario).
- Flujo visual de tareas.
- Limitación de la cantidad de trabajo. Aquí se usa el término *WIK* o *Work In Process*.

De *SCRUM* toma los siguientes elementos y características:

- Reunión diaria.
- Roles: equipo y otros sin especificar.
- No se añaden *sprints*, pero se añade a la pizarra la columna *To Do*, que recoge lo siguiente a realizar.

Es decir, se le añade a *Kanban* lo necesario para ser considerada más que una herramienta de visualización y pueda ser aplicada como un método.

Su principal aplicación es en proyectos de mantenimiento, pues en este tipo se producen variaciones frecuentes de las tareas, aparición de nuevas tareas, cambios en la prioridad y errores inesperados. En este tipo de situaciones, Scrumban presenta mayor dinamismo que SCRUM.

La *Figura 2-42* es una foto tomada a una pizarra de oficina en la que cada columna es un estado del proyecto y cada *posit* una tarea (los colores indican distintas prioridades).

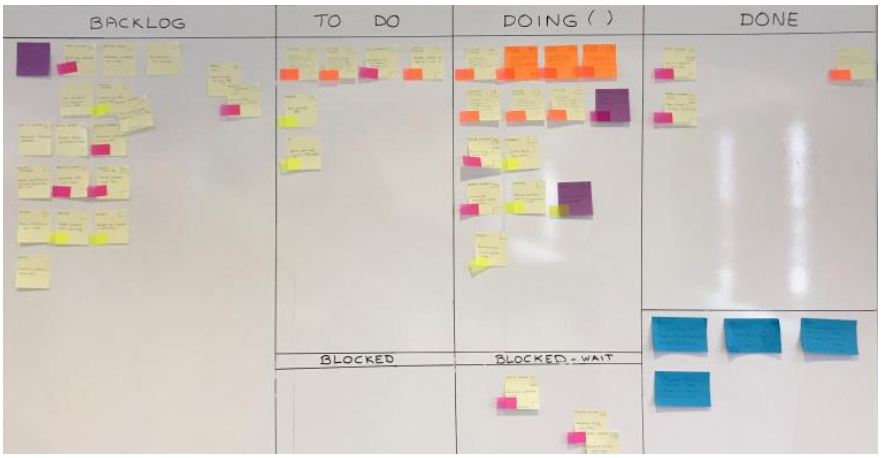

Figura 2-42. Tablero Scrumban real.

Imagen tomada de la url: https://itnove.com/blog/kanban/equipos/significado-y-beneficios-limitar-wip-en-kanban/

3 COMPARATIVA DE METODOLOGÍAS Y ELECCIÓN

Realizado el recorrido y la descripción de las metodologías más usadas y popularizas, tanto tradicionales como ágiles, en este capítulo se va a llevar a cabo un estudio comparativo de éstas para poder identificar las ventajas e inconvenientes de cada una en función de las características o factores del proyecto.

En primer lugar, será necesario hacer una comparativa entre los 2 bloques de metodologías, pues se trata del factor identificativo. Tras este primer criterio, se hará una comparación y establecimiento de criterios para elegir una metodología dentro de un bloque o familia.

La comparativa, obviamente, se basará en características objetivas, definidas en cada una de las metodologías. Sin embargo, el criterio de elección será mayormente subjetivo, puesto que la aplicación de un método puede depender de muchos factores, como política de empresa, petición del cliente, presupuesto, equipo, Evidentemente, hay motivos por los que la aplicación de otra metodología puede resultar ventajosa con respecto a la que se esté aplicando en ese momento, pero puede deberse a políticas de la propia empresa, del cliente o del responsable de equipo.

Cerraremos este capítulo con una pequeña conclusión, una vez hecha la comparativa y establecidos unos criterios o, más bien, sugerencias de elección.

Es importante hacer referencia al gran esfuerzo de investigación y análisis para poder hacer una comparativa objetiva entre metodologías y tener una base sólida para proponer unos criterios de elección de metodología, que no se basen sólo en la experiencia [33][34][35][36][37].

3.1 Tradicionales vs Ágiles

Para poder superponer ambos bloques e identificar los factores claves e identificativos de ambas familias, es necesario recoger una lista con las ventajas e inconvenientes de cada una. Me basaré en las características comunes que hacen que una metodología se considere tradicional o ágil.

Aunque se realice una descripción de los elementos más importantes, se llevará a cabo un estudio comparativo de factores más concretos a través de una tabla, pues es la forma más rápida y clara de contraponer los dos bloques.

De forma resumida, podemos hacer una distinción inicial entre ambos bloques, haciendo alusión a su propia

naturaleza o modelo genérico en el que se basan: las tradicionales se basan en un modelo secuencial predictivo, mientras que las ágiles se basan en un modelo iterativo, adaptativo y empírico. Por lo tanto, como diferencia principal para iniciar la tabla de características, tenemos el gran dinamismo que presentan las metodologías ágiles.

En la *Figura 3-1* se visualiza, de forma genérica, el modelo de ciclo de vida de las metodologías tradicionales y las ágiles.

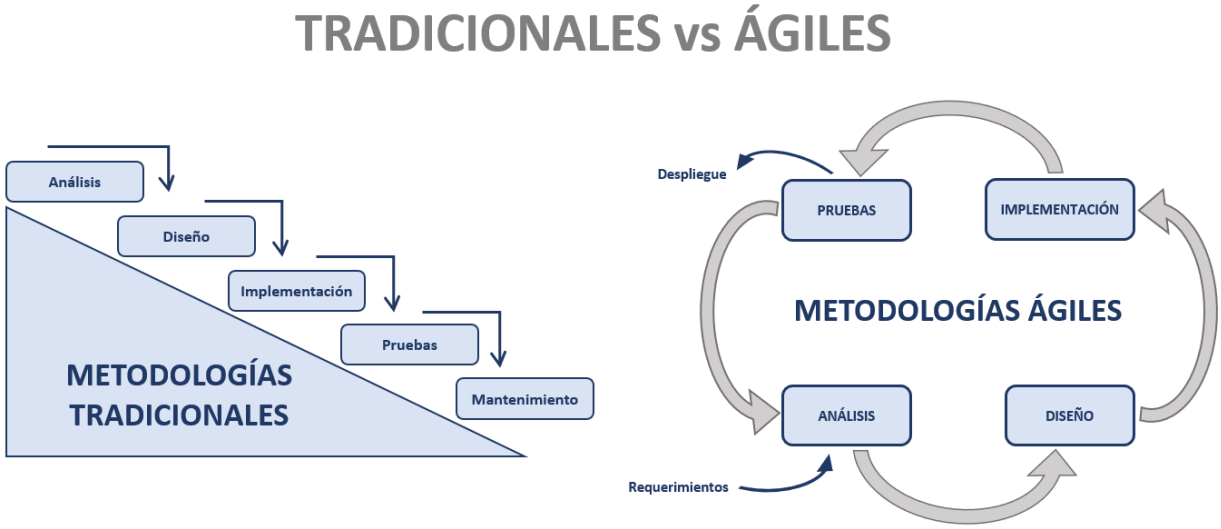

Figura 3-1. Tradicionales vs Ágiles: modelo.

Realización propia

Por lo tanto, podemos hacer una distinción básica entre los dos bloques: modelo **secuencial** de las Tradicionales y modelo **iterativo** de las Ágiles.

Otros dos aspectos claves, previo al detalle de factores, son:

- *Planificación robusta* de las Tradicionales contra la adaptabilidad y *mejora continua* de las Ágiles.
- *Gran impacto y pérdidas económicas* de las Tradicionales ante *errores y cambios* frente al tratamiento de éstos como un elemento de entrada y *mejora del producto* de las Ágiles.
- Implicación del cliente: *Integrado en el equipo* en las Ágiles, y al *principio y final* de cada fase o del proyecto en las Tradicionales.
- *Equipo Disperso vs Equipo Unificado*: en la metodología Tradicional el equipo es disperso, con los roles diferenciados y las tareas son exclusivas de cada rol. En la Ágil, el equipo es un todo, cooperan y trabajan unidos. Es importante, para entender aún mejor la diferencia, la figura del *líder*. Podríamos definir a un líder como la persona que encabeza y dirige a un equipo, estando implicado/a como uno/a más, sin diferencias de posición. Sin embargo, el concepto de jefe va relacionado al poder o autoridad sobre un equipo, con una diferencia de posición en la jerarquía, dirigiendo y no encabezando al equipo, sin formar parte directa de él.
  En la *Figura 3-2* se puede observar la diferencia entre un líder de un equipo ágil, y un jefe de un equipo tradicional.

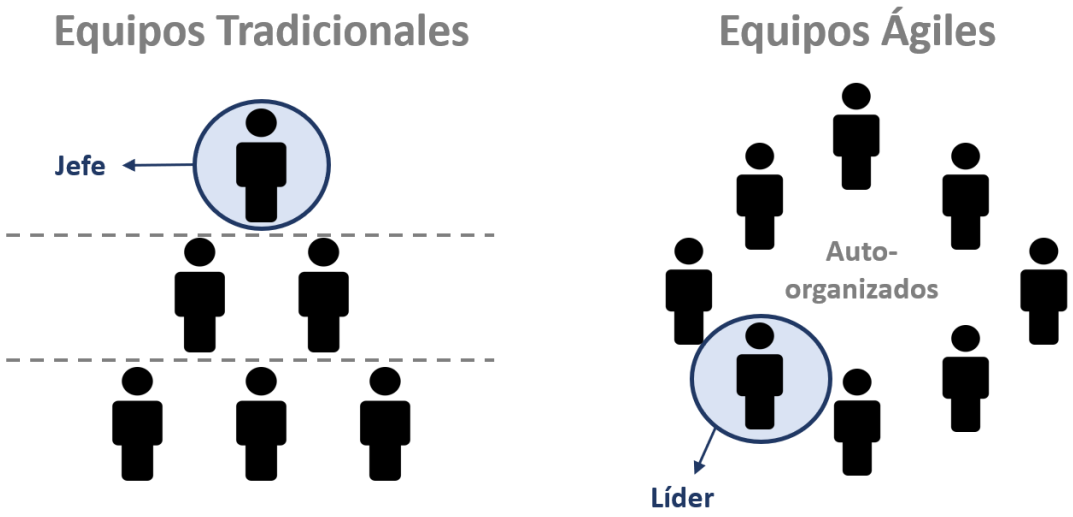

Figura 3-2. Tradicionales vs Ágiles: equipo.

Realización propia

Partiendo de estas diferencias, como elementos clave o básicos para su distinción, se muestra, a continuación, la *Tabla 3-1*, con todas las características y su valor para metodologías Tradiciones y Ágiles:

	METODOLOGÍAS	
Elementos/Características	**TRADICIONAL**	**ÁGIL**
Modelo	Secuencial	Iterativo
Valores prioritarios	Procesos	Personas
Retroalimentación o *Feedback*	Muy poco o nulo	Continuo
Tamaño del proyecto	Cualquier tamaño	Pequeño - Medio
Estimación del proyecto	Proyecto cerrado	Proyecto abierto a modificaciones
Duración del proyecto	Larga - Media	Pequeña
Detalle inicial de los requisitos	Muy alto	Bajo, se va realimentando
Priorización de requisitos	Hay que cubrirlos todos, no se prioriza en detalle	Por valor para obtener un incremento que funcione
Documentación	Cerrada, muy detallada y rigurosa	Abierta, se va mejorando
Gestión	Dirigida	Colaborativa

Desvío del coste	Muy bajo	Puede oscilar muchísimo
Impacto ante cambios	Muy alto	Muy bajo
Tamaño del equipo	Grande	Pequeño
Disposición del equipo	Disperso	Unido
Roles	Específicos y no intercambiables	Genéricos y flexibles
Cliente	Al principio del proyecto y al final de cada etapa	Integrado en el equipo
Entregas	Una entrega al final del proceso	Tantas como *Sprints* o iteraciones
Valor del producto	No se garantiza que sea óptimo	Óptimo y adaptado

Tabla 3-1. Tradicionales vs Ágiles: características.

Realización propia

La *Tabla 3-1* va a ser esencial para analizar los elementos y factores del proyecto a gestionar, permitiendo identificar qué familia de metodologías es la más apropiada en función de las características.

Con el objetivo de profundizar y posicionarnos en uno de los dos bloques metodológicos, en las siguientes subsecciones se van a describir las ventajas e inconvenientes de cada una de estas dos familias.

3.1.1 Ventajas e inconvenientes de las metodologías Tradicionales

Siendo redundante en este aspecto, vuelvo a hacer hincapié en que ninguna metodología, método o herramienta es mejor que otro, sino que dependerá del proyecto en sí y del contexto, empresa o grupo de trabajo en el que se va a llevar a cabo.

De forma objetiva, se van a enumerar y describir las ventajas e inconvenientes que presentan las metodologías Tradicionales, de forma genérica.

3.1.1.1 Ventajas de las metodologías Tradicionales

- Debido a la necesidad de cerrar cada hito o etapa, se realiza una planificación y diseño detallados, a un nivel de detalles muy amplio, que los hacen muy sencillos de entender. Cliente y desarrolladores no dejan ningún punto abierto y están de acuerdo y compenetrados al 100%.
- Como el alcance y las estimaciones se acotan al principio, el seguimiento del proyecto es mucho más fácil, pues se trata de un desarrollo secuencial definido, en el que sólo se puede avanzar.
- La obligatoriedad de cerrar cada etapa para comenzar con la siguiente permite al equipo trabajar en otros proyectos (que se encuentran en otras etapas). Por ejemplo, mientras los analistas acuerdan la documentación y requisitos con el cliente, los desarrolladores están trabajando en otro proyecto de la corporación, que se encuentra en la etapa de desarrollo. Una vez se liberen los analistas y quede cerrada la documentación, pueden empezar con otro proyecto mientras los desarrolladores trabajan y

los testers o equipo de pruebas comienza a plantear el plan de pruebas en función de los requisitos. Es decir, trabajar con varios proyectos en cascada permite sincronizarlos para que los diferentes perfiles siempre estén trabajando. Por este motivo, se puede descomponer un proyecto muy grande en proyectos en cascada más pequeños, que se pueden encajar para abordarse en paralelo.

- Una vez cerrados los requisitos, el cliente se libera y no es necesario estar en contacto con él durante el desarrollo.
- Se realiza una única entrega de un producto "completo" y no varias entregas de "trocitos".

3.1.1.2 Inconvenientes de las metodologías Tradicionales

- Como consecuencia al cierre predictivo de los requisitos al finalizar la etapa de análisis, es normal encontrar inconvenientes, dudas e incongruencias durante la etapa de desarrollo. Este problema es más frecuente en el desarrollo software, donde no se detectan detalles y elementos más específicos hasta que no se empieza a implementar. En este caso, el equipo de desarrollo consulta al cliente, pero siempre adoptando una solución que se ciña lo máximo posible al requerimiento inicial. Para el caso de un proyecto con un producto físico, una buena práctica antes de desarrollarlo es la realización de un maqueta o prototipo.
- También se puede sufrir el descontento del cliente con el producto entregado, ya que sólo participa en la documentación y no lo ve hasta el final del proceso.
- Poca flexibilidad en el desarrollo y un elevado coste para la modificación o adaptación de un producto, pues se trabaja con una documentación cerrada y se sigue avanzando hasta que se obtiene el producto completo, que se entrega al cliente.

3.1.2 Ventajas e inconvenientes de las metodologías Ágiles

De la misma forma que con las Tradicionales, se van a recoger un conjunto de ventajas e inconvenientes, de forma objetiva y genérica, de las metodologías Ágiles.

Nos servirá para, una vez identificada la naturaleza del método a usar, tener un segundo filtro para contrastar y antiparnos a los distintos factores que nos podemos encontrar durante el ciclo de vida de un proyecto. Podemos, por lo tanto, hacer un tratamiento de los *pros* y los *contras* como una especie de *checklist*, de manera que comprobemos si nuestro proyecto comparte estas características, y contempla posibles factores adversos.

3.1.2.1 Ventajas de las metodologías Ágiles

- Homogenización del equipo, en la que el propio cliente se siente parte de él, trabajando codo con codo con el resto del equipo.
- Anticipación al resultado del producto y entregas iteradas, lo que permite al cliente visualizar cómo se está trabajando, tomar decisiones y poder solicitar modificaciones.
- Posibilidad de entregar una versión terminada pero más básica del producto si el mercado lo requiere, mejorándose con un evolutivo con posteriores iteraciones.
- Sincronización entre el desarrollo y el usuario final.

3.1.2.2 Inconvenientes de las metodologías Ágiles

- Indisponibilidad o desinterés del cliente, poniendo en peligro la cooperación.
- La metodología es mucho óptima si todos los miembros del equipo están centrados y dedicados únicamente al proyecto en cuestión. Otro punto clave es la ubicación física de todo el equipo en un mismo lugar, que no siempre es posible.
- Entregas incompletas en algún *timebox*, lo que retrasaría el proyecto y supondría un incremento del coste. A veces, se producen por decisiones del cliente, que el equipo de desarrollo no contempló en la planificación de la iteración.
- Posible refactorización del producto debido a cambios de alcance que conllevarían reducción del valor o calidad.

3.2 Criterios de elección de metodología

La elección de la metodología "ideal" no existe, ni siquiera hay una guía o estándar para poder hacer una correcta selección. Como ya se ha remarcado, no sólo dependerá de las características del proyecto, sino que, en muchos casos, es algo que viene impuesto por la propia empresa u organismo.

Además, no todas las descritas en el capítulo anterior son metodologías completas para abarcar un proyecto, sino que algunas son conjuntos de principios, recomendaciones y buenas prácticas. Ciertas metodologías cubren todas las etapas del proyecto, pero otras se centran en un rol concreto. Hay que destacar también que algunas son de aplicación por parte de gobiernos de ciertos países.

Sin embargo, trataremos de hacer un análisis o guía "subjetiva" para poder identificar una metodología acorde a las especificaciones de nuestro proyecto. Será mucho más fácil su entendimiento cuando, en la siguiente sección del TFG, se proponga una necesidad real de un cliente, un equipo por parte del proveedor y una solución, factores que podremos encajar en la guía que definiremos a continuación.

El primer punto, para el cual hemos realizado la comparativa entre las dos grandes familias de metodologías, es hacer una elección entre Tradicional y Ágil.

Para ello, podemos basarnos en las siguientes preguntas, que se responderán con las características del proyecto a gestionar:

1. ¿Cuál es la prioridad del negocio o cliente?
2. ¿Cuál es la implicación y cómo es la interacción con el cliente?
3. ¿Cuánto tiempo de duración tiene el proyecto?
4. ¿Cuál es el tamaño del equipo?
5. ¿La planificación es rigurosa, cerrada y controlada, o abierta y contempla incertidumbre?
6. ¿Cómo es la jerarquía de roles y comunicación?
7. ¿Se trabaja sobre una solución cerrada o modificable?
8. ¿Se realiza una única entrega, o entregas frecuentes?
9. ¿Los requisitos están cerrados rigurosamente o a alto nivel y con posibilidad de inserción, eliminación y/o modificación?
10. ¿Cómo es el riesgo e impacto ante errores o cambios?
11. ¿Cuál es el nivel de documentación?
12. ¿Las funcionalidades en el equipo pueden cambiar y ser intercambiables?

Obviamente, para hacer una identificación, será necesario dar una respuesta a estas preguntas, identificando

el valor de éstas para las 2 familias de metodologías. En la *Tabla 3-2* se da respuesta, en los dos extremos, para poder tener una estimación de la naturaleza de la metodología a implementar. Es una aproximación que facilitará el descarte de uno de los 2 bloques. En función del valor (más o menos la respuesta), se recomienda usar una metodología dentro de la familia, aproximación que se hará posteriormente:

Pregunta	METODOLOGÍAS	
	+ -	- +
	TRADICIONAL	ÁGIL
Prioridad del Negocio	Cumplimiento de la planificación	Valor del producto
Implicación e interacción con el cliente	Baja	Muy alta
Duración del proyecto	Larga	Corta
Tamaño del equipo	Grande	Pequeño
Planificación	Rigurosa y cerrada	Abierta y adaptable
Jerarquía de roles y comunicación	Cerrada, organizativa	Dinámica, colaborativa
Solución	Cerrada	Adaptada
Entrega	Única	Frecuente
Requisitos	Rigurosamente claros y cerrados	Ambiguos y abiertos
Riesgo e impacto ante errores o cambios	Alto	Bajo
Documentación	Alta	Baja
Funcionalidades del equipo	Estáticas	Flexibles

Tabla 3-2. Tradicionales vs Ágiles: elección.

Realización propia

Si las respuestas, en su mayoría, caen claramente en una columna concreta, no habría ningún tipo de dudas para elegir una metodología Tradicional o Ágil.

Por el contrario, si hay algún tipo de incertidumbre o duda, quizás los factores más importantes para guiar nuestra elección sean los siguientes:

I. **¿Cómo impactarían los cambios durante el transcurso del proyecto?**
   a. Poco impacto, pues no se ciñe a una planificación estricta y está abierto a modificaciones para aumentar el valor del producto → Ágil
   b. Gran impacto, pues incumple la planificación, ya que supondría volver a una etapa que se había dado por cerrada (como puede ser el análisis, la replanificación). Supondría pérdidas irrecuperables → Tradicional

II. **¿Se quiere una o pocas entregas con el producto final, o se van a realizar "entregables" que se van a validar y analizar?**
   a. Entrega con solución final → Tradicional
   b. Entregables que se validan y mejoran → Ágil

III. ¿Cómo sería la implicación del cliente?

a. Cierre de requerimientos, plazos y aceptación del producto → Tradicional

b. Interacción constante, forma parte del equipo → Ágil

Se proponen los criterios de la tabla y las últimas preguntas como una forma de identificación, basándome en los aspectos más notorios de ambos tipos de metodologías.

El siguiente paso, y mucho más complejo, es elegir una metodología dentro de la familia seleccionada. No es nada trivial ni obvio, pues cada una de las descritas en la sección anterior tiene una serie de particularidades e incluso permite su fusión con otras.

Una práctica aconsejable es acudir directamente a un estudio oficial de mercado, donde podremos ver la popularidad y nivel de satisfacción con las distintas metodologías. No hay mejor baremo que una valoración basada en la experiencia.

Hay 2 factores internos para tener en cuenta a la hora de poder elegir o no de forma certera:

- Política de gestión de la empresa → en muchos casos, la empresa tiene como política y práctica habitual el uso de unos métodos y herramientas concretas, ya sea por la formación de los empleados o por acuerdos internos con organismos.
- Conocimiento y adaptabilidad de la metodología por el equipo → en función de las exigencias del proyecto, a veces es más viable seguir con la metodología actualmente aplicada que formar al equipo desde cero para introducir una nueva. Esto es trabajo interno de la empresa, que debe montar un plan de formación y capacitación del equipo a través de cursos, formaciones y certificaciones.

Debido a estos factores, hay metodologías más popularizadas que se podrían considerar como dueñas del monopolio en la gestión del proyecto. Es lo que permite trabajar en un marco común y fiable, en el que se afianzan las tareas tanto en proveedores como clientes, haciendo posible la especialización y la mejora profesional por experiencia.

Acudiendo al *CHAOS Report*, que se trata de un análisis y estudio del éxito de los proyectos, llevado a cabo cada año por *The Standish Group* (organización asesora de investigación primaria fundada en 1985 y centrada en el rendimiento del desarrollo de software), podemos visualizar, en la *Tabla 3-3*, los resultados de usar un tipo de metodología u otro, y cuál ha sido la más empleada.

En concreto, vamos a mostrar los resultados obtenidos en 2019, pues los de 2020 aún no han sido publicados. Se trata de una media global, sin hacer diferenciación en el tamaño de proyectos (a mayor tamaño, sube la efectividad de los tradicionales):

MÉTODO	ÉXITO	CAMBIO	FALLO
ÁGIL	42%	50%	8%
TRADICIONAL	26%	53%	21%

Tabla 3-3. CHAOS report 2019.

Datos obtenidos de la publicación oficial (The Standish Group, 2019): https://www.standishgroup.com/

Este resultado nos puede orientar, pero no debemos tomarlo como una medición única y final. Añadir, por su utilidad, que las metodologías puras más usadas (sin fusionar ni alternar con otras) de cada una de las familias

han sido SCRUM para Ágiles, y PMBOK para tradicionales.

Aun así, se va a resumir cada una de las metodologías vistas en este TFG, dando información concreta que nos pueda orientar a la hora de elegir una.

A. METODOLOGÍAS TRADICIONALES

1. PMBOK → Cubre todas las fases del ciclo de vida del proyecto. Consta de 47 procesos, agrupados en 5 conjuntos y 10 áreas de conocimiento.
2. PRINCE2 → Para ambientes controlados, se centra más en la dirección. Cubre organización, gestión y control. Consta de 8 procesos y 8 áreas de conocimiento.
3. ICB → Centrado en la dirección y en la estandarización y optimización en el cierre del proyecto. Consta de 46 elementos organizados en 3 grupos.
4. SWEBOK → Es un cuerpo de conocimiento que provee de buenas prácticas para todas las etapas, divido en 15 áreas de conocimiento.
5. SSADM → Usado por el Gobierno de Reino Unido, se basa en una rigurosa documentación para el diseño y planificación. Es de aproximación, completamente estática.
6. MÉTRICA → Usada por el Gobierno de España, cubre las etapas de planificación, desarrollo y mantenimiento.
7. MERISE → Usada por el Gobierno de Francia, abarca las primeras cuatro etapas: viabilidad, análisis, diseño e implementación. Deja fuera la fase de mantenimiento.

B. METODOLOGÍAS ÁGILES

1. SCRUM → Metodología ágil más popularizada y utilizada. Requiere de una participación plena del cliente, que es otro miembro más del equipo. Se basa en continuas reuniones y entregas. Es sencilla de entender y de aplicar.
2. XP → Centrada en las relaciones interpersonales, también exige una participación activa del cliente, pero menos intensa que en *SCRUM*. El número y frecuencia de entregas también baja con respecto a la anterior.
3. Crystal → Conjunto de metodologías adaptadas al tamaño del equipo y al nivel de criticidad. Conlleva muchas entregas. Está más enfocada a la implementación que al mantenimiento.
4. AM → Sirve como complemento a otras, pues no es una metodología en sí, sino un conjunto de recomendaciones enfocadas en el modelado y documentación.
5. ASD → Metodología basada en la adaptación continua. Más independiente del cliente, se apoya en la especulación, colaboración y aprendizaje del equipo, que es auto-organizado.
6. AUP → Metodología muy completa, pero a su vez un poco más compleja de aplicar. Prioriza de forma especial el valor, pero a su vez exige 2 tipos distintos de iteraciones, lo que hace más complicado su uso. Requiere conocimientos robustos de la metodología.
7. DSDM → Dentro de las ágiles, es la más cercana a las tradicionales. Tiene un mayor peso en la documentación y prototipado de la solución. Apoyado en una planificación más estricta, cubre todas las etapas de un proyecto.
8. FDD → Gira entorno a las características o funcionalidades del proyecto, que no son más que los requerimientos. Permite un desglose en ramas, a distintos niveles de profundidad, de cada requerimiento o función. Para su seguimiento, se apoya en la frecuente entrega de informes.

9. LSD → Filosofía o forma de pensar, que recoge principios de buenas prácticas para crear un contexto de trabajo óptimo, en el que poder aplicar una metodología ágil con éxito y eficacia.
10. Scrumban → Con menos robustez que SCRUM, permite gestionar proyectos cambiantes constantemente. De aplicación en proyectos con mayor incertidumbre y con requerimientos abiertos, es una buena elección para mantenimiento y estabilización de soluciones, pues permite la aparición de nuevas tareas, asignación y monitorización actualizada en todo momento.

3.3 Conclusiones

La metodología ideal o perfecta no existe, sino que ésta tomará más o menos lógica y efectividad en función de las características del proyecto. No obstante, sí que se puede realizar un análisis de contextualización para tener una aproximación a una metodología u otra.

En el terreno profesional, normalmente cada empresa tiene sus métodos de gestión de proyectos, por acuerdos y sincronización con el cliente, adaptación y homogenización del mercado laboral, o formación y preparación interna del equipo. Sin embargo, sí que es posible la aplicación de metodologías distintas dentro de equipos concretos, que conllevaría a la formación de todos los integrantes para poder aplicar un nuevo método.

A pesar de abogar por la correcta aplicación de la metodología, en muchos casos hay que salirse un poco del guion establecido, pues a veces surgen factores, necesidades y exigencias que rompen con lo planificado o estimado, no sólo con las metodologías tradicionales. ¿Quién iba prever, por ejemplo, la situación de alerta a nivel mundial, que ha obligado a dinamizar y modificar por completo el modelo y ritmo de trabajo? Otro factor que empuja a una improvisación del guion es el cliente en sí. Hay, aún, muchas grandes empresas reacias a la adaptación y cambios metodológicos, que deben ser "educadas" para la aceptación y aplicación completa de nuevas filosofías de trabajo. Hay veces en las que, por intereses, pactos o imposiciones del negocio, se ordena desde un nivel superior a la gestión de un proyecto el cese de la dinámica que se estaba llevando a cabo, para cubrir de forma masiva otros frentes o, lo que vulgarmente se conoce como "apagar fuego". No obstante, esta situación no deja de ser fruto de una planificación o gestión poco efectiva pues, más arriba de la gestión de un proyecto, hay una gestión de gestiones o del negocio (terreno que no cubrimos en este TFG).

Sin embargo, en esta sección se han establecido una serie de criterios o *checklist* para, al menos, conocer cuándo es posible aplicar una metodología Ágil. Además, se ofrece un mini-resumen de las aplicaciones o características claves de las metodologías descritas en el presente documento. El detalle, procesos y herramientas de cada una se recoge en la segunda sección del documento y, también se facilitan las fuentes y referencias para poder hacer una consulta más exhaustiva.

De esta forma y con una carga de subjetividad, podríamos hacer un resumen del método sugerido para elegir un tipo de metodología:

- Usar la *Tabla 3-2* para dar respuesta y posicionar los factores característicos del proyecto en una familia de metodologías.
- Se ha hecho un estudio de *pros* y *contras* de cada tipo de metodología, que pueden servir de guía para comprobar la compatibilidad de la metodología con nuestro proyecto.
- Tener en cuenta los elementos más característicos y diferenciadores:
    - Prioridad.
    - Implicación del cliente.
    - Respuesta ante cambios y errores. Planificación y requerimientos (estáticos o dinámicos).
    - Frecuencia y número de entregas.

Elegido el tipo de metodología, el método concreto dependerá de características y niveles de éstas. Es la

parte más compleja, pues hay un inmenso abanico de posibilidades. Una sugerencia realizada es consultar el ranking de uso y éxito, publicado por organizaciones oficiales. Como es de esperar, también dependerá de la formación del propio equipo. Se ha recogido, de forma breve, el objetivo y elementos identificativos de cada metodología.

En el mundo laboral, una práctica común es fusionar distintas metodologías, tomando de cada una lo interesante, de acuerdo con la naturaleza del proyecto.

Podemos cerrar este punto con las siguientes conclusiones:

- Hay que entender las metodologías como un conjunto de herramientas y buenas prácticas para abordar proyectos de distintas características y con un ciclo de vida concreto.
- Ninguna metodología es ideal ni mejor que otra. Hay que verlas relativamente y siempre en un contexto de aplicación.
- No hay una definición clara ni una delimitación exacta de las fronteras entre las metodologías. Es decir, no son excluyentes ni se indica de forma concreta qué tipo de proyectos abarcaría. Hay metodologías que tienen muchos elementos en común, otras que sirven de complemento, incluso otras que aportan ideales y formas de pensar. Incluso metodologías clasificadas en distintas familias comparten características.
- No hay un criterio absoluto para la elección de una metodología. La mayor dificultad sufrida en el trabajo de investigación realizado ha sido intentar hacer una comparativa, cuando se trata de algo muy complejo, que ni siquiera organizaciones oficiales han publicado. No es nada trivial, sino que habría que basarse en la documentación oficial, pero, sobre todo, en la experiencia, resultados y el ensayo prueba-error.
- Aunque, quizás se note en el documento un poco más mi inclinación hacia las metodologías ágiles (por mi experiencia con ellas, mi formación y el caso práctico que voy a describir en el siguiente punto), ninguna familia de metodologías es superior a la otra. Simplemente, parten de prioridades y procedimientos distintos: las tradicionales tienen como base un análisis y planificación riguroso para asegurar la implementación e ir cerrando etapas, y las ágiles usan la incertidumbre, cambios y errores durante la implementación para iterar, replanificando y adaptando el resultado.
- En la práctica, la mejor opción es intentar adoptar la metodología que cuadre con las características de nuestro proyecto como base, y combinarla con otras, que complementen y nos ofrezcan soluciones eficientes y eficaces. Es decir, debemos romper con la idea de exclusividad, abrirnos al uso de nuevas herramientas y, sobre todo, aprender y formarnos para dotar de calidad tanto nuestro trabajo, como el de nuestros/as compañeros/as.

4 CASO PRÁCTICO

Ser un Gerente de Proyectos es como ser un artista,
tienes los flujos de procesos de diferentes colores
combinados en una obra de arte.

- Greg Cimmarrusti -

En este capítulo vamos a llevar todo el trabajo de investigación teórica a la práctica, planteando un proyecto real, basado en un caso gestionado personalmente en mi empresa actual. Para proteger la información y propiedad intelectual, no se van a dar datos reales del cliente, de la empresa proveedora ni detalles del producto que comercializan.

Haciendo un breve resumen, hemos recogido las metodologías de gestión de proyectos más popularizadas y utilizadas, clasificadas en las dos grandes familias, tradicionales y ágiles, dando una especie de "guía" o consejos para poder identificar la metodología más adecuada a nuestro proyecto, en función de sus características.

Mi mayor aporte en este TFG es llevar toda esta teoría a la práctica, con un proyecto ya gestionado, recogiendo tanto la elección de la metodología como toda la ejecución del mismo. De esta forma podremos poner a prueba los criterios de elección de metodología y sus resultados o consecuencias, a medida que el proyecto avanza y tienen lugar ciertos acontecimientos o aparición de otros factores.

En primer lugar, se recogerán las necesidades del cliente y las características del proyecto para poder hacer una elección de metodología. A continuación, en función de la metodología elegida, se narrará todo el desarrollo del proyecto, de forma que se pueda comprender cómo se aplica un método a un caso real, y los factores que condicionan el avance de un proyecto en el marco laboral.

4.1 Necesidad del cliente

Nuestra empresa, especializada en soluciones de software e infraestructuras para pequeñas y medianas empresas (PYMEs), recibe la siguiente necesidad de un cliente que se dedica a vender ciertos productos:

1. PROBLEMA

- Actualmente tienen una web en la que se registran y gestionan clientes para realizar los pedidos, pero que es incapaz de dar un servicio de atención al cliente efectivo.
- Los diferentes canales o vía de comunicación de los clientes con el equipo de atención son: un correo electrónico de atención, un número de teléfono, perfil empresarial en Facebook e Instagram. También tienen un apartado de preguntas frecuentes en la web con un formulario que se envía al correo

electrónico de atención.

- Estas vías de comunicación no son eficientes ni permiten, en una única herramienta, tener las consultas e incidencias relacionadas con clientes y pedidos para, de un simple vistazo, ver la información relacionada y poder dar una solución de calidad a los clientes.
- Además, no permiten realizar informes de forma fiable para analizar la calidad del servicio. Tampoco ofrece un servicio de notificación automático al cliente, ni envío de formularios.
- Todo el servicio de atención al cliente es manual, lo que hace perder mucho tiempo a los trabajadores.
- No hay una asignación automatizada de trabajo a los agentes, lo que hace que se solapen y se pierda tiempo.

2. NECESIDADES

- Tener una herramienta de atención al cliente en la nube, o lo que es lo mismo, un *Service Cloud* que amplíe las posibilidades de comunicación del cliente con la empresa proveedora, así como hacerlos converger en un único lugar para tener el control y trazabilidad de todas las interacciones. Esta herramienta estará sincronizada con la BBDD (base de datos) de la web original para tener toda la información de clientes, pedidos y productos, sobre los que se crearán casos (consultas e incidencias).
- Los canales de comunicación a recoger serían:
    - Un chat con un bot en la web actual, que responda a consultas y cree incidencias automáticamente en la nueva herramienta.
    - Un formulario en la web actual, que crea casos automáticamente en la nueva herramienta.
    - El correo de atención se sincronizará con la nueva herramienta, creándose un caso automáticamente.
    - La línea de WhatsApp estará sincronizada con la nueva herramienta.
    - Las cuentas de Facebook e Instagram están vinculadas a la nueva herramienta para que se creen casos cada vez que haya alguna interacción con las cuentas de la empresa.
- Se quiere monitorizar la actividad y resultados a través de informes y gráficas, para poder analizar las consultas e incidencias más frecuentes y, así, mejorar el servicio.
- Notificación automática al cliente o interesado/a, para transmitirle confianza.
- Envío de encuestas de satisfacción para mejorar el servicio.
- Automatización de todos los pasos previos posibles, para facilitar y optimizar el trabajo de los agentes: hay pasos que se repiten siempre, como pedir datos personales y del pedido; hay preguntas frecuentes que se contestan de la misma forma.
- Asignación de casos a los distintos agentes, para no solaparse entre ellos y optimizar los tiempos.

En resumen, se requiere una herramienta de atención al cliente en la nube, que esté sincronizada con la BBDD de la web de pedidos para optimizar las consultas e incidencias de los clientes, haciendo converger todos los canales de comunicación de éstos con el negocio, automatizando procesos de resolución, optimizando la calidad del servicio con asignación y reparto de trabajo, bots, respuestas automáticas y encuestas, y que permita monitorizar la información para hacer un análisis de calidad a través de informes y gráficas.

Por último, como dato vital, el cliente necesita una solución lo más rápida posible, solicitando un *MVP* o "producto mínimo viable" cuanto antes, para lo que da total disponibilidad. El producto final se irá detallando y adaptando a medida que se desarrolla y prueba, recogiendo la impresión de la experiencia de los agentes y la opinión de los clientes.

Podemos identificar 2 elementos claves en este dato:

- Implicación total del cliente.
- Varias entregas, como mínimo 2:
    - MVP: herramienta sincronizada con la BBDD de la web de pedidos, que cree caso de forma automática a través del correo y formulario, y permita abrir chat de respuestas automáticas. Además, se envían alertas de email al cliente para informar de la apertura y cierre del caso, enviando una encuesta de satisfacción.
    - Producto final, refinado y con el resto de las funcionalidades: bot entrenado para chat y WhatsApp, vinculación y detección de consultas e incidencias en Facebook e Instagram, optimización de asignación y colas de tareas, informes y gráficas de análisis.

4.2 Elección de metodología

Antes de comenzar el desarrollo del proyecto, es necesario elegir la metodología que permita una gestión óptima, en base a las características del proyecto.

Para ello, nos basaremos en la *Tabla 3-2*, que recoge las principales diferencias para elegir una metodología tradicional o ágil.

Primero, justificaremos la elección de la respuesta a cada pregunta de la tabla para, finalmente, plasmar ésta con los resultados y la elección.

Los elementos característicos del proyecto serían los siguientes:

- Prioridad del Negocio → puesto que se trata de un servicio de atención al cliente que optimice y dote de mayor calidad la experiencia del usuario, así como el trabajo de los agentes, la prioridad del negocio sería el **valor del producto**.
- Implicación del cliente → por explícita petición, el cliente da una disponibilidad **total**, pues la solución urge y se va a terminar de definir en base a la experiencia y el propio desarrollo.
- Duración del proyecto → el negocio necesita empezar a utilizar la herramienta lo antes posible, pues empieza la temporada de Navidad y la propuesta ha sido lanzada a finales de septiembre. Por lo tanto, la duración debe ser lo más **corta** posible.
- Tamaño del equipo → se trata de un proyecto relativamente pequeño y que necesita más la implicación de **pocas personas** completamente sumergidas e implicadas en la solución, que de un equipo excesivamente desalineado.
- Planificación → las necesidades básicas están definidas, pero los detalles y funcionalidades más concretas están **abiertas** a la **adaptación** y experiencia durante el desarrollo.
- Jerarquía de roles y comunicación → como ya se ha comentado, se tiene disponibilidad completa del cliente y se necesita pocas personas, pero que estén completamente mimetizadas con el proyecto, tomando decisiones y añadiendo valor. Por lo tanto, se necesita **dinamismo y colaboración** más que una jerarquía burocrática para agilizar el proyecto.
- Solución → el negocio da unas necesidades básicas de las que partir, pero necesita una solución **adaptada** a la experiencia y validaciones tanto de los agentes como de los clientes, optimizando el producto.
- Entrega → se solicita, como mínimo, una entrega del *MVP* y otra con el producto final. Así que, como

mínimo va a haber **2 entregas** en un periodo corto de tiempo.

- Requisitos → se dan los requisitos básicos, pero de forma **generalizada** y se deja la lista **abierta** para irlos recogiendo y desarrollando en paralelo al desarrollo.
- Impacto ante errores o cambios → **bajo**, de hecho, se basa en el dinamismo y los cambios para llegar a una solución de valor y calidad.
- Documentación → **poca**, la justa para comenzar. Sí que se entregará una documentación técnico-funcional al finalizar el proyecto, con la solución detallada.
- Funcionalidades del equipo → el equipo de desarrollo va a realizar tanto labores más funcionales, como técnicas, incluso va a tomar decisiones de cara a ofrecer una mejor solución. Se necesitará un equipo **flexible**.

Seleccionando estos valores en la *Tabla 3-2*, propuesta en el capítulo anterior como *checklist* para identificar el tipo de metodología, obtendríamos la *Tabla 4-1*.

Pregunta	METODOLOGÍAS	
	+ -	- +
	TRADICIONAL	ÁGIL
Prioridad del Negocio	Cumplimiento de la planificación	Valor del producto
Implicación e interacción con el cliente	Baja	Muy alta
Duración del proyecto	Larga	Corta
Tamaño del equipo	Grande	Pequeño
Planificación	Rigurosa y cerrada	Abierta y adaptable
Jerarquía de roles y comunicación	Cerrada, organizativa	Dinámica, colaborativa
Solución	Cerrada	Adaptada
Entrega	Única	Frecuente
Requisitos	Rigurosamente claros y cerrados	Ambiguos y abiertos
Riesgo e impacto ante errores o cambios	Alto	Bajo
Documentación	Alta	Baja
Funcionalidades del equipo	Estáticas	Flexibles

Tabla 4-1. Caso práctico: elección de tipo de metodología.

Elaboración propia

Claramente, la metodología a usar cuadra perfectamente con las características y objetivos de una metodología Ágil.

Como hemos visto, hay un gran abanico de posibilidades dentro de esta familia, pero en este caso tenemos dos elementos diferenciadores, claves de una metodología en concreto:

- Participación plena del cliente.
- Comunicación periódica y más de una entrega.

Aunque, como se indicó en el capítulo anterior, ninguna metodología es mejor que otra, y un mismo

problema puede resolverse con distintas metodologías, incluso de diferente familia, en este caso nos interesa utilizar SCRUM. Se debe a los 2 rasgos característicos anterior, y al conocimiento y experiencia del equipo y del negocio con esta metodología.

4.3 Gestión del Proyecto

Elegida la metodología y aceptada la propuesta, se le propone formalmente una oferta al cliente, en la que se recogen las necesidades a cubrir y se propone un coste de los recursos necesarios para llevar a cabo el proyecto.

En este caso en concreto, el propio negocio ha propuesto fechas para la implantación, tanto del *MVP*, como del producto completo, que debe ser en 2 meses, durante octubre y noviembre, de modo que esté totalmente operativo en diciembre.

Para la estimación del tiempo, toma de requerimientos y organización del proyecto, se elabora el *Kick-off* o inicio, que es un documento-presentación formalizado en el que se recoge el detalle del equipo, metodología a realizar, y el alcance a alto nivel (por funcionalidades).

Con el objetivo de entender el propósito del *Kick-off*, se dedica un subapartado a este elemento.

Indica los primeros pasos en el *Kick-off* (podría considerarse como fase inicial o sprint 0), nos sumergiremos en la metodología *SCRUM*, describiendo los roles, artefactos y eventos, y cómo se llevaron a cabo.

4.3.1 Fase inicial: *Kick-off*

Las necesidades planteadas por el negocio describen lo que sería el alcance a alto nivel. Basándonos en ellas, se elabora un documento, con el que se presenta al cliente las funcionalidades que se van a cubrir, junto a los recursos necesarios, agenda de sesiones (*sprint planning*) de análisis para definir el *product backlog*, y división del proyecto en *sprints*, de modo que se ajuste a los tiempos solicitados y se cubran los requisitos.

Podríamos ver esta fase inicial en la *Figura 4-1*.

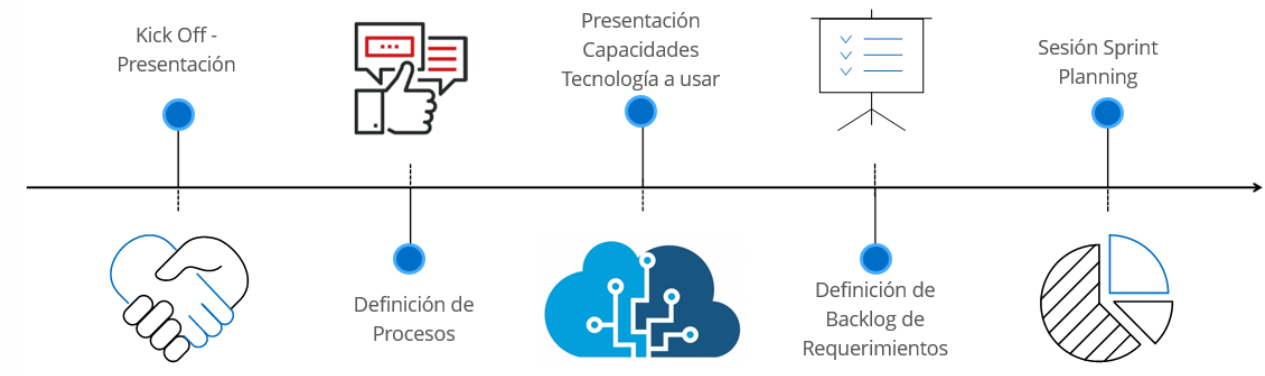

Figura 4-1. Fase inicial del proyecto.

Realización propia

- En la presentación se hace una referencia a las habilidades y terrenos que abarca la empresa, así como el posicionamiento en la industria.
- En la definición de procesos se recogen las necesidades planteadas por el cliente, que serán cubiertas en el proyecto.
- Se presenta la tecnología a usar, mostrando las capacidades y opciones que ofrece.

- Se proponen sesiones conjuntas con el negocio para definir el *Product Backlog* del que se partirá, que se irá ordenando y modificando en las distintas iteraciones, para darle mayor valor al producto.
- Se agenda la sesión de *Sprint Planning* para, una vez definida la pila de producto, se decida qué requerimientos se van a cubrir en el primer *sprint*.

Para las necesidades propuestas, se decide formar el siguiente equipo, mostrado en la *Figura 4-2*:

- *Scrum Master* → se encargará de asegurar la correcta aplicación de la metodología *SCRUM*, estando de coordinador y consultor tanto del cliente como del proveedor.
- *Product Owner* → miembro del equipo que cerrará los requerimientos con el cliente, que además va a desempeñar el rol de *Consultor funcional*.
- *Equipo Scrum* → además del *Product Owner*, que también va a trabajar como *Consultor funcional*, el equipo contará con la presencia de un *Consultor Técnico* y un *Desarrollador Senior* de la tecnología a usar. El motivo por el que hay 2 consultores y 1 único desarrollador especializado es que la tecnología de implantación permite desarrollo "sin código" para dar una solución estándar a la parte funcional, mientras que el *desarrollador senior* añadirá la parte personalizada y más compleja, que se escapa de la solución estándar.
- *Representante de negocio* → responsable de la parte del cliente que trabajará de forma conjunta con el resto del equipo, aclarando dudas, validando dudas y tomando decisiones.

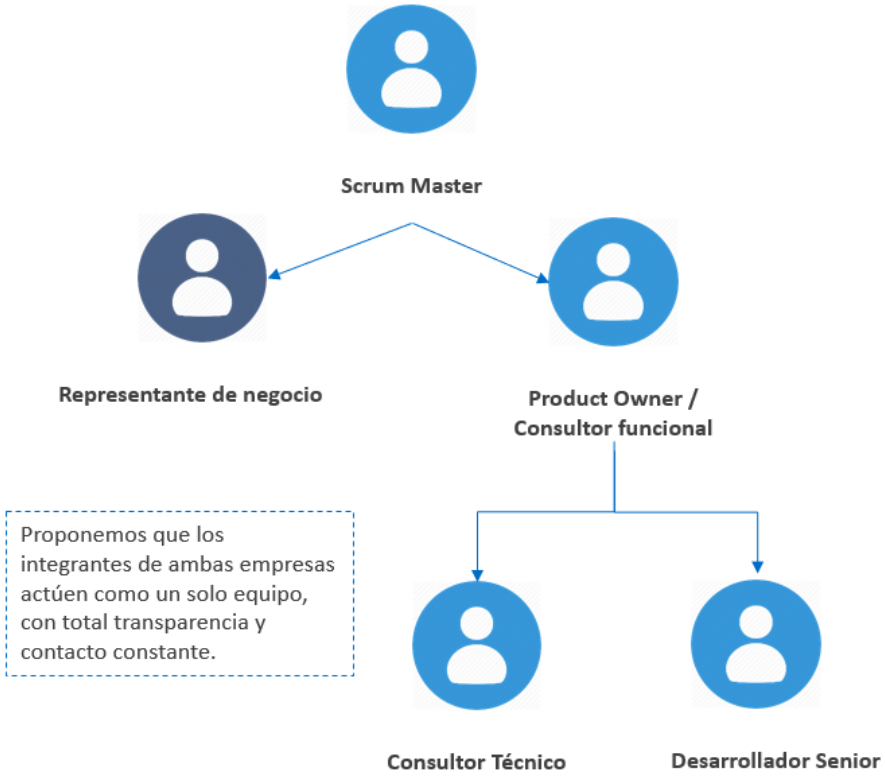

Figura 4-2. Equipo Scrum propuesto.

Realización propia

En cuanto a las herramientas y tecnologías que se utilizarán, se hace la siguiente aclaración en el *Kick-off*:

- Metodología: SCRUM, mediante backlogs y *sprints*.
- Herramienta de gestión: JIRA y Teams (ahora haremos una definición de estas herramientas).
- Documentación: entrega y actualización de la documentación funcional y técnica de cada desarrollo realizado.
- Entregas: cada 2 semanas se realizarán sesiones de seguimiento y validación.

Antes de avanzar, hacemos un pequeño inciso para definir las herramientas de gestión que se usan en el proyecto:

- **JIRA**: software de gestión de proyecto creada por Atlassian, que ofrece un tablero de visualización de las tareas a realizar, pudiendo recoger todo el *Product Backlog* para dividirlo y asignarlo entre los diferentes *sprints*, así como poder indicar el estado, responsable, horas y descripción de cada tarea o requerimiento. Promueve y ayuda al desarrollo y entrega iterativa e incremental, permitiendo realizar informes y visualizar las tareas en un tablero *Kanban*, con las siguientes columnas de estado, en las que se moverán las distintas tareas:
    - Pila → tareas que están aún sin iniciar.
    - En curso → tareas que están en desarrollo, ya empezadas, pero sin finalizar.
    - Hecho → tareas finalizadas.
    - Bloqueado → tareas bloqueadas, por distintos motivos (falta de información, falta de recursos, problemas técnicos externos, …) que impiden avanzar.

  Además, permite crear bloques en los que agrupar las tareas, por funcionalidades, como integración, migración, canales de entrada, funcional, documentación.

  Cuando lleguemos a la parte de requerimientos y desarrollo de cada *sprint*, se mostrará visualmente.

- **Teams**: espacio de trabajo ofrecido por Microsoft y basado en el chat de Office 365, que permite la comunicación y colaboración de los equipos de trabajo. Además de disponer de chat, se puede crear un equipo con carpetas de contenido compartido. Permitirá al equipo estar en constante comunicación y trabajar sobre documentos actualizados en todo momento.

  Se creará un equipo en el que estarán todas las personas involucradas: representante de negocio (cliente), *Scrum Master*, *Product Owner* y equipo *Scrum* (consultores y desarrollador). La *Figura 4-3* muestra el aspecto del equipo de Teams.

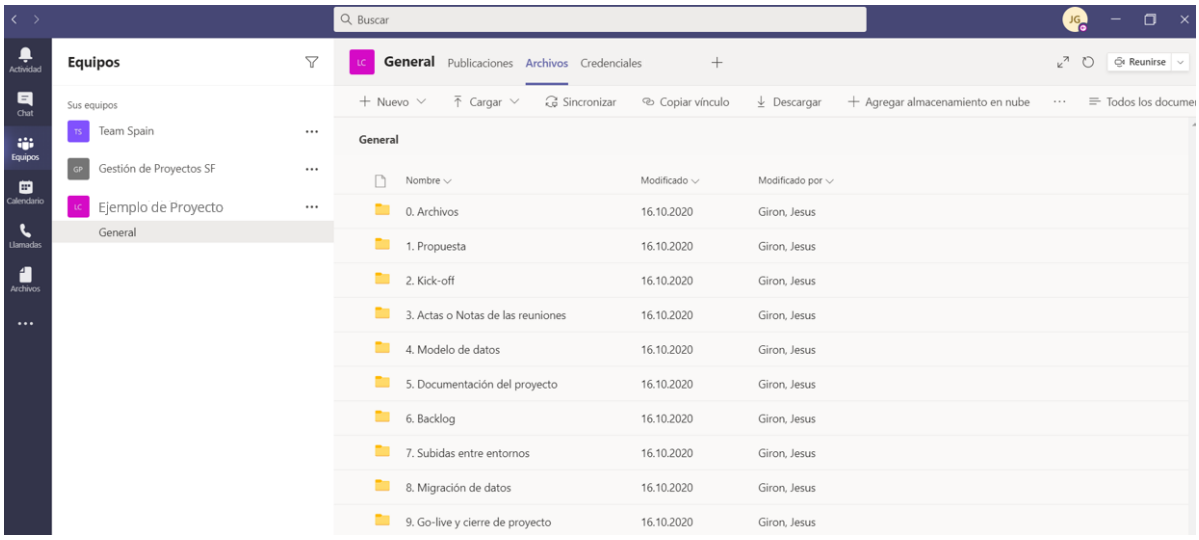

Figura 4-3. Equipo de Teams para la gestión y comunicación.

Realización propia

Aunque la metodología SCRUM sea de tipo ágil y se base en las distintas iteraciones para mejorar y entrar en detalles de los requerimientos, obviamente se necesita una planificación inicial, en la que se recojan los distintos *sprints* y las fechas de ejecución y entrega. Evidentemente, no se recoge al detalle cada tarea, pues el objetivo es cuadrar con el cliente los distintos *sprints*, entendiéndose que todos los días hay *daily scrum*, previo a cada *sprint* hay un *Sprint Planning* y, tras la entrega, un *Sprint Retrospective*.

Para ello, utilizo un diagrama de *Gantt* en el que se recoge la estimación temporal de las distintas entregas, indicados por semanas, mostrado en la *Tabla 4-2*.

	S 0	S 1		S 2		S 3		S 4		S 5		S 6		S 7		S 8		S 9	S 10
Reuniones de análisis inicial																			
Análisis																			
Sprint 1																			
Entrega 1																			
Sprint 2																			
Entrega 2										MVP									
Sprint 3																			
Entrega 3																			
Sprint 4																			
Entrega 4																			
Testing																			
Documentación y Formación																			
Soporte post-implantación																			

Tabla 4-2. Planificación inicial de sprints y entregas.

Elaboración propia

Como se puede observar, se ha dividido el proyecto en 4 *sprints*, siendo el segundo de ellos el *MVP*. Es decir, la entrega del segundo *sprint* conlleva despliegue e integración (los detalles los vemos en el *bakclog*).

Hay una primera semana para toma de requerimientos y análisis y, en cada entrega, se realiza una fase de pruebas, entrega de documentación técnico-funcional de lo realizado, y una formación al usuario final.

Planteada la planificación general, a alto nivel, del proyecto, se propone en el mismo *Kick-off* un calendario para las sesiones de análisis de la primera semana (con los temas a tratar, basados en las necesidades presentadas por el cliente), en la que se definirá el *Product Backlog*, a partir del cual se definen los *sprints*. Este calendario se muestra en la *Tabla 4-3*.

Tabla 4-3. Calendario de sesiones de análisis.

Realización propia

Por último, se añade el detalle de las sesiones de análisis propuestas, con la fecha y hora en la que se van a tratar los distintos elementos o funcionalidades, y la necesidad de asistencia de ciertas personas. Se muestra en la *Figura 4-4*.

1ª Sesión	2 de Octubre (10:00 – 11:30)	Requerido asistencia de
30 min	Modelo de datos: - Almacenamiento de la información y relación entre "entidades" - Flujo funcional	- Representante del cliente - Product Owner (y consultor funcional)
15 min	Usuarios, Roles y Perfiles: - Seguridad y visibilidad de la información	- Representante del cliente - Product Owner (y consultor funcional)
45 min	CASOS: - Tipificación - Proceso de Soporte (estados) - Canales de entrada: formulario, web, whatsapp, RRSS - Recepción: colas y usuarios - Notificaciones, auto-responses	- Representante del cliente - Product Owner (y consultor funcional) - Consultor técnico
2ª Sesión	**5 de Octubre (10:00 – 11:30)**	**Requerido asistencia de**
30 min	Chat: - Flujo de encaminamiento de respuesta	- Representante del cliente - Product Owner (y consultor funcional)
10 min	Encuestas y recuperación de datos	- Representante del cliente - Product Owner (y consultor funcional)
15 min	Productos y catálogo	- Representante del cliente - Product Owner (y consultor funcional)
35 min	Knowledge / FAQ	- Representante del cliente - Product Owner (y consultor funcional)
3ª Sesión	**7 de Octubre (10:00 – 11:30)**	**Requerido asistencia de**
30 min	Migración de datos: - Extracción de la información - Adecuación y carga en SF	- Representante del cliente - Product Owner (y consultor funcional) - Consultor técnico - Proveedor del cliente
40 min	Integraciones: - Correo - Prestashop	- Representante del cliente - Product Owner (y consultor funcional) - Consultor técnico - Proveedor del cliente
20 min	Informes y paneles	- Representante del cliente - Product Owner (y consultor funcional)

Figura 4-4. Detalle de las sesiones de análisis.

Realización propia

4.3.2 *Product Backlog*

Tras llevar a cabo las sesiones programadas con el cliente para el análisis funcional de las necesidades, se elaboró el *Product Backlog* o Pila del Producto, con todos los requerimientos necesarios para cubrir la propuesta.

Puesto que se trata de un proyecto de urgencia y que el cliente tiene bastante claro, se elaboró una muy buena aproximación de lista de requerimientos a alto nivel. Eso sí, los detalles quedaban completamente abiertos a las iteraciones, criterios del equipo y las pruebas.

De esta forma, se acordó el siguiente *Product Backlog*, mostrado en la *Tabla 4-4*, en el que cada requerimiento se identifica con un código, pertenece a un bloque (las tareas se identifican y asignan mejor en base a la necesidad técnica o funcional) y tiene unas horas estimadas para su dedicación (basadas en el análisis con el negocio):

Código de Req	Requerimiento	Bloque	Horas estimadas
PR-1	Modelo de Datos	Funcional	8
PR-2	Usuarios	Funcional	2
PR-3	Seguridad	Funcional	2
PR-4	Record Pages	Funcional	4

Código de Req	Requerimiento	Bloque	Horas estimadas
PR-5	Canal de entrada: Formulario Web	Canales de Entrada	4
PR-6	Fichas para objetos personalizados	Funcional	1
PR-7	Mapping de campos	Integración	8
PR-8	Carga de datos (Migración)	Migración	8
PR-9	Ruta para Pedidos	Funcional	1
PR-10	Horario de Oficina	Funcional	1
PR-11	Email-to-Case	Funcional	4
PR-12	Formatos Compactos	Funcional	2
PR-14	Tests Apex	Desarrollo	4
PR-15	Asignación de Casos a Usuarios	Desarrollo	4
PR-16	Auto-respuesta: Apertura de Caso	Desarrollo	3
PR-17	Auto-respuesta: Cierre de Caso	Desarrollo	3
PR-18	Encuestas	Desarrollo	4
PR-19	Omnichannel	Funcional	4
PR-20	Config Supervisor Omnichannel	Funcional	2
PR-21	Canal de entrada: Chat	Canales de Entrada	8
PR-22	Encaminamiento de respuesta del Bot	Desarrollo	8
PR-24	Canal de entrada: WhatsApp	Canales de Entrada	8
PR-25	Canal de entrada: RRSS	Canales de Entrada	8
PR-26	Informes y Paneles	Funcional	6
PR-27	Guía para muestra del Producto	Documentación	4
PR-28	Manual de Usuario	Documentación	24
PR-29	Documentación Técnico-Funcional	Documentación	24
PR-30	Integración: Cuentas	Integración	2
PR-31	Integración: Pedidos	Integración	2
PR-33	Integración: Direcciones	Integración	2
PR-34	Formación	-	8
PR-35	Despliegue	-	8
PR-36	Pruebas	-	16
PR-37	Ruta para Casos	Funcional	1

Tabla 4-4 Caso práctico: Product Backlog inicial.

Elaboración propia

El orden en el que se recogen los requerimientos no es indicativo, ya que fueron apuntados a medida que se iban cubriendo las necesidades que planteaba negocio.

La sesión de análisis fue un poco intensa, con una duración de 4 horas y media, repartidas equitativamente en 3 días, pues el cliente tenía bastante claros los objetivos e incluso tenía una base de conocimiento técnico suficiente de la tecnología.

El siguiente paso, para gestionar de forma ágil el desarrollo del proyecto y poder ser visualizado por todo el equipo (incluido el representante del cliente), fue crear un proyecto en *JIRA*, añadiendo todos los requerimientos. La herramienta tiene el aspecto mostrado en la *Figura 4-5*.

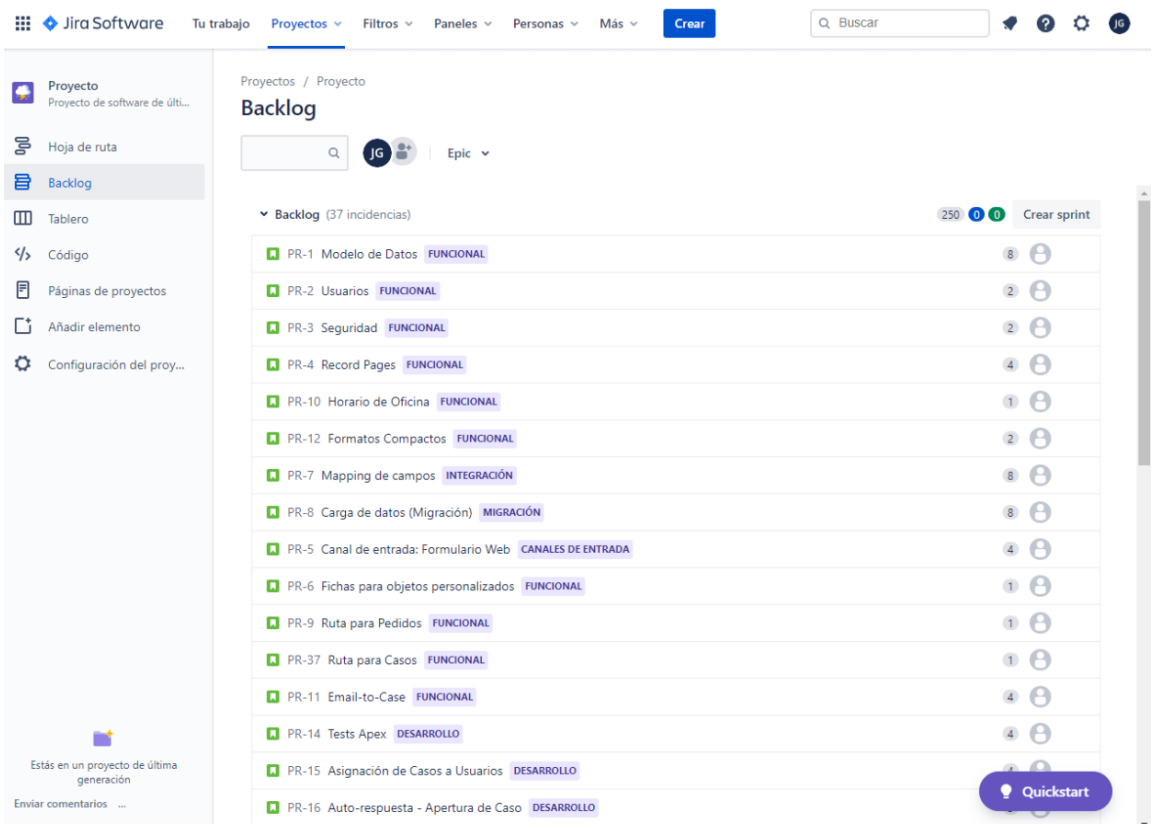

Figura 4-5. Backlog Inicial en JIRA.

Realización propia

Recordemos, como se vio en el *Kick*-off, que se decidió abordar el desarrollo del proyecto en 4 *sprints* de 2 semanas cada uno, dándole al desarrollo del proyecto un total de 2 meses. Se pretendió que en el segundo *sprint* se consiguiera el *MVP* o Producto Mínimo Viable, que tendría las funcionalidades básicas para poder utilizar la herramienta (se verá el reparto en el detalle de cada *sprint*). Adicionalmente, como cierre del proyecto, se dan 2 semanas de mantenimiento.

Con el *Product Backlog* recogido y acordado, ya se puede realizar el primer *Sprint Planning* para decidir qué requerimientos se abordarán en el primer *Sprint*.

4.3.3 *Sprint* 1

Cerrado el análisis y el *Product Backlog*, se convocó al equipo para realizar la Planificación del Sprint (reunión que tuvo una duración de 2 horas) y dar respuesta a las siguientes preguntas (mismo procedimiento en todos los *sprints*):

1. ¿Qué puede hacerse y entregarse en este *sprint*?
2. ¿Cómo se conseguirá realizar el trabajo necesario para la entrega del incremento?

En este primer *Sprint Planning*, el *Product Owner* (que en este caso también va a trabajar de *Consultor funcional*) junto con el resto del equipo, decidieron abordar los siguientes aspectos:

- Parte más visual y funcional de la herramienta: diseño de páginas, formatos, fichas, barras de estados.
- Estructura de los datos: modelo de datos, equivalencia de campos con el sistema origen.

- Configuración y seguridad de usuarios: jerarquía de roles, accesos, permisos y visibilidad.
- Carga de set de datos del origen: migración parcial de la BBDD de la web origen para poder visualizar la información en la nueva herramienta, y así poder comprobar la correcta implementación.
- Documentación de este primer incremento.

En *JIRA* se seleccionaron los requerimientos correspondientes para cubrir estas necesidades, agrupándolas en el *Sprint* 1, o lo que es lo mismo, definiendo el *Sprint Backlog 1*, quedando como se muestra en la *Figura 4-6*.

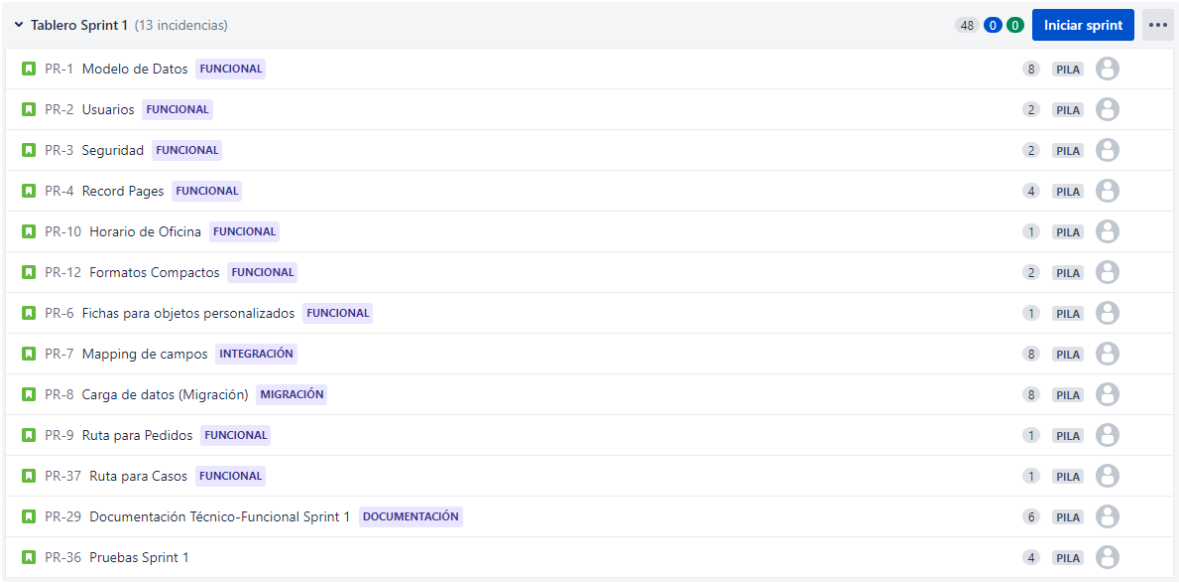

Figura 4-6. Sprint Backlog 1 en JIRA.

Realización propia

Desde el primer día de inicio del *sprint 1* (que abarcó 2 semanas) se celebró una *Daily Scrum* de 9:00 a 9:15 de la mañana para analizar el avance del día anterior, qué avance habrá el día actual y si hay algún bloqueo u obstáculo.

Durante la primera semana, se avanzó satisfactoriamente, como estaba previsto, pero al inicio de la segunda semana surgieron unos contratiempos:

- El modelo de datos planteado estaba incompleto: se iba a trabajar sobre *Clientes, Direcciones, Pedidos, Productos y Casos*, pero se vio necesaria una entidad intermedia entre *Pedidos y Productos*, que se llamó *Productos del Pedido*, y relacionaba cada uno de los ítems o líneas del pedido con un producto. Hubo que añadirla al modelo de datos (incrementando las horas de dedicación estimadas), además de añadir al *Product Backlog* la generación del servicio de integración con la web origen para cargar esta información.
- Como consecuencia del punto anterior, también hubo que realizar el *mapping* o equivalencia de campos entre el sistema origen y destino para contemplar la nueva entidad, y así poder extraerla para cargarla.

En este momento, el *Kanban* o tablero del estado del *sprint 1* presentaba el aspecto recogido en la *Figura 4-7*.

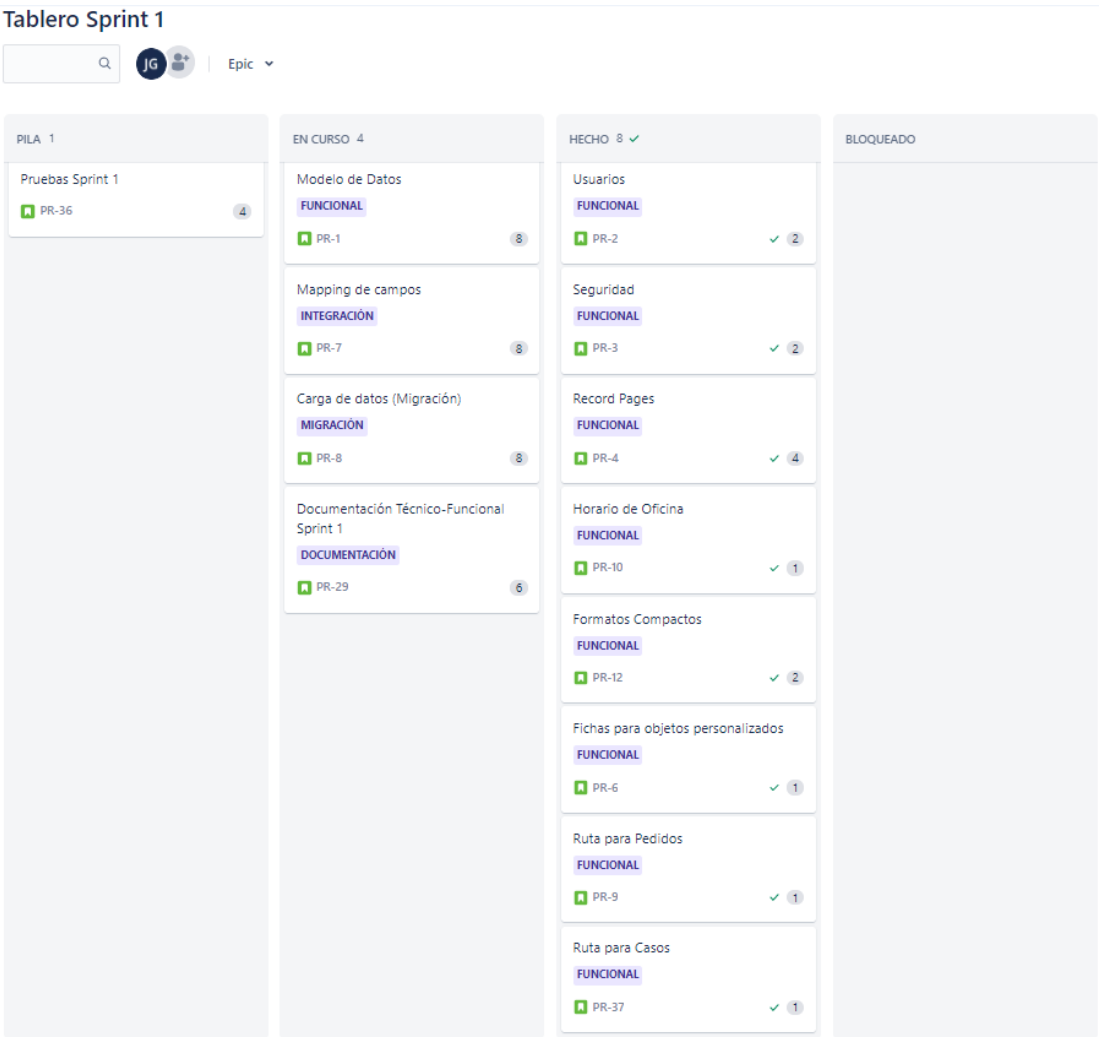

Figura 4-7. Kanban Sprint 1 en JIRA.

Realización propia

Puesto que este tipo de situaciones son las comunes en un proyecto basado en SCRUM y se cuenta con la participación continua del cliente, no supuso ningún problema, pues se solventó sobre la marcha, tomando las decisiones conjuntamente para que este primer *Incremento* cubriera lo ideado para el primer *sprint*.

El *sprint 1* se cerró en el tiempo estimado (2 semanas), cubriendo, además de enriquecer los puntos o requerimientos acordados. Además, se realizaron pruebas visuales tras la carga de información para validarlo, y se redactó la documentación de lo implementado en este *sprint*.

Para dar el "OK" al *sprint 1*, se celebró una reunión, el *Sprint Review*, donde se revisó y comprobó el incremento realizado. Tuvo una duración de 2 horas, haciendo las pruebas necesarias para considerar la implementación como "Terminada". Un punto clave e interesante de esta reunión fue añadir al *Product Backlog* el requerimiento del servicio de integración de la nueva entidad (producto del pedido).

El cierre del primer *sprint* tuvo lugar con el *Sprint Restrospective*, reunión de 1 hora con el equipo, en el que se comentan y recogen algunas mejoras a tener en cuenta, en base al *sprint* realizado. Como resultado, salieron 2 mejoras interesantes:

- Implicación de terceros: en este caso, para evitar dejar información relevante de la BBDD fuera, se propone al cliente que involucre al equipo técnico que les mantiene su web de pedidos.
- Un pequeño desarrollo para mostrar las imágenes de los productos en la herramienta, de modo que se

tenga una especie de vista de catálogo para facilitar el trabajo a los agentes que van a usar el servicio semi-automatizado de atención al cliente.

Se añade al *Product Backlog* este pequeño desarrollo.

4.3.4 *Sprint* 2

Se lleva a cabo de la misma forma que el primero:

- *Sprint Planning* para definir el *Sprint Backlog*. El tablero muestra las tareas a realizar, así como las ya cerradas en el *sprint* anterior, como se muestra en la *Figura 4-8*.

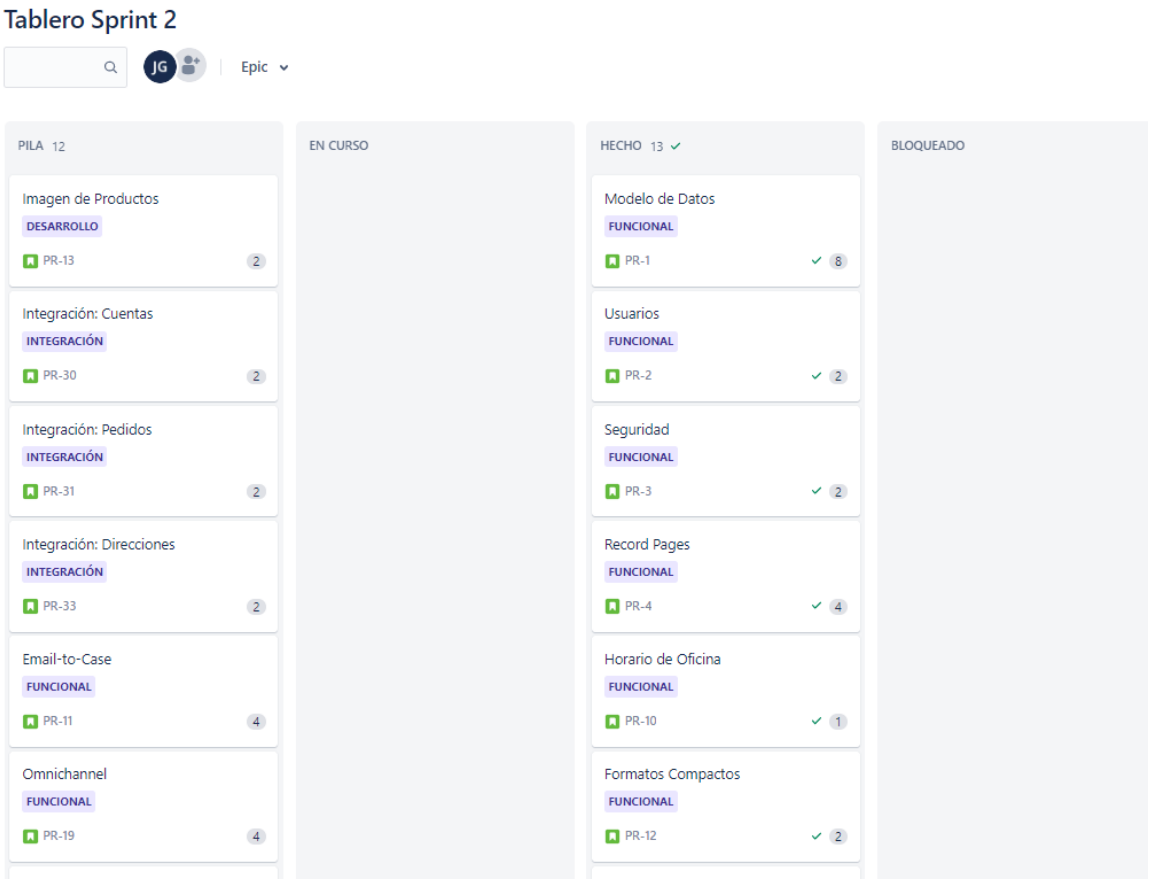

Figura 4-8. Kanban Sprint 2 en JIRA.

Realización propia

- *Daily Scrum*: todos los días a primera hora (de 9:00 a 9:15) durante las 2 semanas del *sprint*, como punto de control y actualización.
- Desarrollo del *sprint 2*: comenzó como estaba previsto, avanzando en las tareas funcionales y de desarrollo, de forma conjunta con el cliente, hasta que surgió el gran bloqueo del proyecto. Por la parte del cliente, su equipo técnico no estaba capacitado para desarrollar un servicio de integración como se acordó, en el que ellos nos enviaban la actualización de *Clientes, Direcciones, Pedidos, Productos del Pedido y Productos* de se BBDD. Deciden pedir presupuesto para subcontratar esta implementación, pero no le salía rentable. Por este momento, decidimo de forma conjunta cambiar el sentido de la

integración y desarrollarlo nosotros con las siguientes consecuencias:

- Aumentar el coste, pues requiere más horas de desarrollo.
- No se puede desplegar el *MVP* en este segundo *sprint*, pues la herramienta no tiene sentido si no está sincronizada con la BBDD de clientes y pedidos.

El estado que presentaba el *sprint 2* en este momento era el mostrado en la *Figura 4-9*.

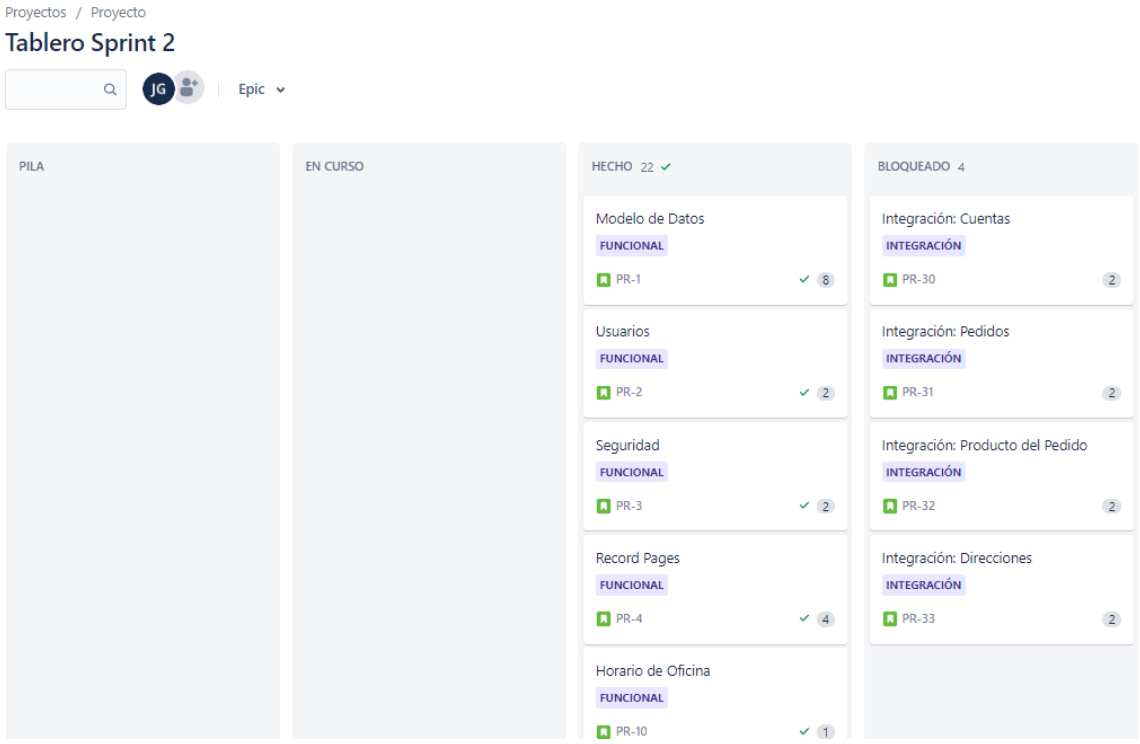

Figura 4-9. Kanban Sprint 2.2 en JIRA.

Realización propia

- *Sprint Review*: con una duración, esta vez de 3 horas, se revisó lo implementado y se añadieron al *Product Backlog* las tareas de integración con el nuevo sentido (implementamos nosotros, consultando su BBDD). El resto de los puntos sí se dan como "Terminados" y aprobados.
- *Sprint Retrospective*: duró 1 hora más que la primera, y se decidió tomar el mando de la integración nosotros solos, sin implicación de terceros, y sólo con el apoyo del cliente (sin otros proveedores). Se revisó la pila para añadir y dar más peso a la integración, de forma que para el siguiente *sprint* se pudiera tener el *MVP*.

4.3.5 *Sprint* 3

Como se ha indicado, se convirtió en el *sprint* de entrega del *MVP* por imposibilidad de hacerlo en el *sprint 2*.

En estas 2 semanas, se realizó la implementación de los servicios de integración para sincronizar la nueva herramienta con la web original del negocio. Además, se habilitó la gestión inteligente de los canales de entrada para los clientes, se refinó la lógica de asignación de casos a usuarios desarrollada en el *sprint 2* y se mostró una "demo" del producto, previa a las pruebas, despliegue y documentación.

De la misma forma, se gestionó el desarrollo de estas 2 semanas con los eventos y artefactos de SCRUM:

- *Sprint Planning* para definir el *Sprint Backlog.* Se seleccionaron las tareas de:
    - Integración con web.
    - Gestión inteligente de canales de entrada.
    - Envío de encuestas de satisfacción con el servicio a los clientes que usarán la herramienta.
    - Guía y demo para el negocio, previa al despliegue del *MVP*.
    - Pruebas del *MVP*.
    - Despliegue.
    - Documentación del *sprint*.
- *Daily Scrum*: todos los días a primera hora (de 9:00 a 9:15) durante las 2 semanas del *sprint*, como punto de control y actualización.
- Desarrollo del *sprint 3*: el gran peso lo tuvo la implementación de los servicios de integración que, por parte del negocio no se pudieron realizar y se re-ofertaron para desarrollarlos nosotros. Las siguientes tareas no presentaron imprevistos.

  Se hizo un aseguramiento del *MVP* a través de pruebas y una *demo* al cliente y, una vez verificado y aprobado, se realizó el despliegue y puesta en producción para su uso, sobrando incluso un par de días en los que estuvimos de soporte y formación a los usuarios, que ya se enfrentaban a un uso real.
- *Sprint Review*: debido al volumen de pruebas y necesidad de aprobación para el despliegue del *MVP*, esta reunión tuvo una duración de poco más de 1 hora. Era necesario dar el *incremento* como "terminado" para su puesta en producción, por lo que ya se había hecho previamente un seguimiento riguroso.

  Quiero destacar la gran satisfacción, tanto del negocio como de nosotros, del gran trabajo y gestión realizada durante el *Sprint 3*, consiguiéndose un *MVP* aceptable en producción.
- *Sprint Retrospective*: las 2 jornadas de trabajo conjunto con el usuario, que utilizaba la herramienta en producción, dieron para varias mejoras y propuestas a incluir en el siguiente y último *sprint*:
    - Al ser una herramienta de atención al cliente, se daba un patrón y redundancia en ciertas consultas e incidencias de los clientes. Por lo tanto, podría haber una forma de automatizar tanto el chat como WhatsApp para que diera respuestas sin necesidad de dedicación de una persona. Por lo tanto, se decidió añadir al siguiente *sprint*:
        - Un bot que tomara información a través de chat y WhatsApp y que pudiera dar respuesta a las preguntas frecuentes, bajando la capacidad de ocupación de los usuarios.
        - Una base de conocimiento, en la que se añadirían las preguntas frecuentes y casos redundantes, de la que se alimentaría el bot para "aprender" y mejorar tanto la calidad como la cantidad de sus respuestas.
    - Modificación en las encuestas enviadas para recoger información relevante que no se había considerados inicialmente (no sólo satisfacción con el servicio, sino también propuestas de mejora).

  Se añadieron las modificaciones al *Product Backlog* para el siguiente y último *sprint*.

  Esta sesión tuvo una duración de 2 horas.

4.3.6 *Sprint* 4

Llegamos al último *sprint* con la salida del *Sprint Retrospective* anterior, además de lo que quedaba pendiente:

- Reporting: Informes y paneles.
- Manual de Usuario del producto completo.
- Formación.
- Pruebas finales.
- Documentación Técnico-Funcional total.

Al igual que en los anteriores *sprints*, se llevó a cabo de la siguiente forma:

- *Daily Scrum*: todos los días a primera hora (de 9:00 a 9:15) durante las 2 semanas del *sprint*, como punto de control y actualización.
- Desarrollo del *Sprint 4*: Durante la primera semana se completaron las mejoras y nuevas implementaciones que quedaban pendientes, pudiendo dar inicio a las pruebas internas de la totalidad del producto.

  La segunda semana se invirtió en dar la formación a los nuevos usuarios finales (en 2 sesiones de 4 horas), realizar las pruebas con el usuario (tras su aprobación se desplegó la nueva versión), elaborar el manual completo y cerrar la documentación técnico-funcional del producto completo.
- Los siguientes eventos, *Sprint Reiview* y *Sprint Retrospective*, se convirtieron en una sesión para marcar el cierre del proyecto. En esta sesión se revisó nuevamente la pila de requerimientos, analizando las modificaciones y verificando que se cubría la totalidad de las necesidades del negocio. Además, se pusieron sobre la mesa las cuestiones que fueron más bloqueantes para hacer un análisis:
  - El punto más bloqueante fue el relacionado a los servicios de integración, que inicialmente los iba a implementar la parte de negocio pero que, por indisponibilidad de su equipo técnico, terminamos haciendo nosotros.

    Relamente, el único impacto fue no poder desplegar el *MVP*, saliendo perjudicado el propio cliente. No obstante, de cara a los objetivos y tiempos del proyecto, no hubo un impacto notorio, pues mientras estos puntos estaban bloqueados, pudimos avanzar con otros desarrollos, cambiando las prioridades de la pila.

Cerramos las iteraciones mostrando el estado final del *Product Backlog* en la *Tabla 4-5*, en el orden en el que se realizaron los requerimientos, anotando el *sprint* en el que se llevaron a cabo, las modificaciones y comentarios aclarativos:

Código Req	Requerimiento	Bloque	Sprint	Horas estimadas	Comentarios
PR-1	Modelo de Datos	Funcional	1	16	Inicialmente estimado en la mitad, pero hubo que añadir un nuevo objeto y enriquecer campos
PR-2	Usuarios	Funcional	1	2	
PR-3	Seguridad	Funcional	1	2	
PR-10	Horario de Oficina	Funcional	1	1	
PR-4	Record Pages	Funcional	1	4	

Código Req	Requerimiento	Bloque	Sprint	Horas estimadas	Comentarios
PR-6	Fichas para objetos personalizados	Funcional	1	1	
PR-9	Ruta para Pedidos	Funcional	1	1	
PR-37	Ruta para Casos	Funcional	1	1	
PR-12	Formatos Compactos	Funcional	1	2	
PR-7	Mapping de campos	Integración	1	8	
PR-8	Carga de datos (Migración)	Migración	1	8	
PR-36	Pruebas Sprint 1	-	1	4	
PR-29	Documentación Técnico-Funcional Sprint 1	Documentación	1	6	
PR-13	Imagen de Productos	Desarrollo	2	2	No se había recogido este pequeño desarrollo
PR-19	Omnichannel	Funcional	2	4	
PR-5	Canal de entrada: Formulario Web	Canales de Entrada	2	4	
PR-21	Canal de entrada: Chat	Canales de Entrada	2	8	
PR-11	Email-to-Case	Funcional	2	4	
PR-16	Auto-respuesta: Apertura de Caso	Desarrollo	2	3	
PR-17	Auto-respuesta: Cierre de Caso	Desarrollo	2	3	
PR-36	Pruebas Sprint 2	-	2	4	
PR-29	Documentación Técnico-Funcional Sprint 2	Documentación	2	6	
PR-15	Asignación de Casos a Usuarios	Desarrollo	2 y 3	4	Iniciado en el sprint 2, pero terminado en el 3
PR-20	Config Supervisor Omnichannel	Funcional	3	2	
PR-25	Canal de entrada: RRSS	Canales de Entrada	3	8	
PR-30	Integración: Cuentas	Integración	3	8	Planificados para el Sprint 2, pero finalmente se realizaron en el Sprint 3. La integración se basaba en un servicio de recepción de información. Sin embargo, por falta de conocimientos técnicos en la parte del usuario, y para aligerar lo máximo posible, se decidió y ofertó nuevamente para que la integración fuese un ataque nuestro, leyendo de su base de datos y trayendo a nuestro sistema.
PR-31	Integración: Pedidos	Integración	3	8	
PR-32	Integración: Direcciones	Integración	3	8	
PR-33	Integración: Productos del Pedido	Integración	3	8	No se había contemplado este objeto/tabla intermedia entre pedidos y productos
PR-27	Guía para muestra del Producto	Documentación	3	4	

Código Req	Requerimiento	Bloque	Sprint	Horas estimadas	Comentarios
PR-14	Tests Apex	Desarrollo	3	4	
PR-35	Despliegue *MVP*	-	3	8	
PR-36	Pruebas Sprint 3	-	3	4	
PR-29	Documentación Técnico-Funcional Sprint 3	Documentación	3	6	
PR-18	Encuestas	Desarrollo	3 y 4	8	
PR-24	Canal de entrada: WhatsApp	Canales de Entrada	3 y 4	16	No se había recogido inicialmente esa estimación, pero aumentó para dotar al Bot de inteligencia en las respuestas
PR-22	Encaminamiento de respuesta del Bot	Desarrollo	4	16	Inicialmente estimado en la mitad porque sólo se querían respuestas automáticas para pedir información y crear el caso. Finalmente se quiso dotar de inteligencia para dar respuestas frecuentes, basadas en una base de conocimiento, también añadida a la pila
PR-23	Base de conocimientos - Knowledge	Funcional	4	4	Hubo que añadir una base de concocimiento con las preguntas frecuentes para que el bot pudiera dar respuestas genéricas automáticas
PR-26	Informes y Paneles	Funcional	4	6	
PR-28	Manual de Usuario	Documentación	4	24	
PR-34	Formación	-	4	8	
PR-36	Pruebas Finales	-	4	4	
PR-29	Documentación Técnico-Funcional Total	Documentación	4	6	

Tabla 4-5. Caso práctico: Product Backlog tras último sprint.

Elaboración propia

4.3.7 Cierre del Proyecto y próximos pasos

Es interesante comentar cuáles fueron los últimos pasos tras completar el desarrollo del proyecto. En este caso, por la política y dinámica de gestión de proyectos llevadas a cabo en la empresa proveedora, una vez completado el último *sprint*, con el producto desplegado y en funcionamiento, tienen lugar los siguientes eventos:

- **Soporte post-implantación**: servicio de 2 semanas, en el que la empresa proveedora está al servicio del negocio para solventar cualquier duda o incidencia con el producto, además de aclarar cualquier aspecto del funcionamiento (el cliente también tiene el manual completo, entregado junto a la documentación).
- **Balance interno**: cada empresa y cada proyecto tiene su modo de hacer balance, dependiendo en gran medida de la metodología y las circunstancias del proyecto. En este caso, al ser un proyecto más bien pequeño y con unos objetivos muy claros por parte del cliente, se pudo hacer una buena estimación en

la fase de análisis y toma de requerimientos.

En concreto, la forma de contrastar los gastos y los beneficios es haciendo una comparativa entre las horas teóricas o estimadas (registradas en JIRA) con las horas reales de dedicación, recogidas mediante las imputaciones del equipo de trabajo de forma diaria.

- **Sesión de cierre de proyecto**: junto con el negocio y todos los involucrados, se convoca una sesión en la que se hace un balance de todo lo realizado y del ciclo de vida del proyecto, para cerrar formalmente y manera conjunta el proyecto, iniciándose el periodo de garantía del producto (en este caso, 6 meses).

  Antes de dar por concluido el proyecto, se sugiere al negocio un listado de evolutivos para una posible fase posterior, dejando abierta la relación con éste. Para ello, se explican los procedimientos, como la gestión por bolsas de horas, que consiste en un "pack" de horas, con diferentes precios en función de la cuantía.

Como se ha indicado, es un caso práctico real, en el que yo mismo trabajé, con unos métodos adaptados a las características y situación del proyecto, que me ha servido para ilustrar la aplicación de SCRUM.

No supone ni representa un uso ideal de la metodología y herramientas. Hay que añadir que no sólo cada empresa y cada equipo dentro de una empresa trabaja de forma diferente, sino que el cliente o negocio también aplica sus formas.

Es más, pasado un tiempo trabajando con los mismos gestores de proyectos y clientes, puedes llegar a identificar quién es el responsable de un proyecto pues, como toda elaboración y creación, lo interesante es darle tu propio sello.

REFERENCIAS

[1] Project Management Institute, *A guide to project management body of knowledge (PMBOK guide)*, 5th edition, 2013.

[2] Celia Desmond, *The ComSoc Guide to Managing Telecommunications Projects*, 2010.

[3] Estepa, R., *Evolución Histórica de las Telecomunicaciones*, 2004.

[4] Joskowicz, J., *Historia de las Telecomunicaciones*, 2012.

[5] Butler, A., *La administración de proyectos: sus funciones y sus errores*, 1990.

[6] UCPR, *Revista Académica e Institucional*, Nº 77, 2009.

[7] Gyepro, *Breve Reseña Teórica de la Gestión de Proyectos*, 2005.

[8] Conalep, *Definición y ejemplos de Proyecto*, 2013.

[9] Aaron J. Shenhar, Stevens y R. Max Wideman, *Toward a Fundamental Differentiation between Project Types*, PICMET'97 conference Innovation in Technology Management, 1997.

[10] Dot Tudor y George A. Walter, *Using an Agile Approach in a Large, Traditional Organization*, 2006.

[11] Marinka Varas Parra, *Examinando los procesos de la Dirección de proyectos*, IX Congreso de Ingeniería de Organización, 2005.

[12] Laboratorio Nacional de Calidad del Software, *INGENIERÍA DEL SOFTWARE: METODOLOGÍAS Y CICLOS DE VIDA*, 2009.

[13] The Stationery Office, *Managing Succefull Projects with PRINCE2*, ISBN: 978-0-11-331059-3, 2009

[14] International Project Management Association (IPMA), *IPMA Competence Baseline (ICB)*, version 3.0, ISBN: 0-9553213-0-1, 2006

[15] Gilles Caupin, Hans Knöpfel, Gerrit Koch, Klaus Pannenbäcker, Francisco Pérez-Polo, y Chris Seabury, *Comparison between ICB and other Project Management Standards*, IPMA, 2004.

[16] IEEE Computer Society, *Guide to the Software Engineering Body of Knowledge (SWEBOK)*, version 3.0, ISBN-13: 978-0-7695-5166-1 ,2014.

[17] SSADM foundation, *Business Systems Development with SSADM*, ISBN 0-11-330870-1, 2000.

[18] Ministerio de Administraciones Públicas del Gobierno de España, *MÉTRICA. Metodología de Planificación, Desarrollo y Mantenimiento de sistemas de información*, version 3, 2001.

[19] D. E. Avison, *MERISE: A European methodology for developing information systems*, European Journal of Information Systems, vol. 1, 1991.

[20] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland y Dave Thomas, *MANIFIESTO AGIL*, 2001

[21] Ken Schwaber y Jeff Sutherland, *Scrum Guide*, 2010.

[22] Ken Schwaber y Jeff Sutherland, *La Guía Definitiva de Scrum: Las Reglas del Juego*, 2013.

[23] Roman Pichler, *Agile Product Management with Scrum: Creating Products that Customers Love*, 2010.

[24] Kent Beck, *Extreme Programming Explained: Embrace Change*, Pearson Education, 1999.

[25] Alistair Cockburn, *Humans and Technology: Crystal Methodologies*, 2004.

[26] Ambler, S. W., Agile Modeling, 2001c, 2001f, 2002.

[27] Highsmith J., Orr K., *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*, 2000.

[28] Ambler, S. W., *Disciplined Agile Delivery*, 2002.

[29] DSDM Consortium, *DSDM Agile Project Framework*, 2014.

[30] Stephen Palmer y John Felsing, *A Practical Guide to Feature Driven Development*, 2002.

[31] Mary y Tom Poppendieck, *Lean Software Development*, 2003.

[32] Lucija Brezočnik y Črtomir Majer, *Comparison of agile methods: Scrum, Kanban, and Scrumban*, 2016.

[33] AENOR, *Directrices para la dirección y gestión de proyectos*, UNE-ISO 21500, 2013.

[34] Mike Cohn, *Agile estimating and planning*, ISBN-13: 978-0-131-47941-8, 2005.

[35] Jeffrey A. Livermore, *Factors that Impact Implementing an Agile Software Development Methodology*, 2007.

[36] RG Figueroa, CJ Solís, AA Cabrera, *METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES*, 2008.

[37] Ashley Aitken y Vishnu Ilango, *A Comparative Analyses of Traditional Software Engineering and Agile Software Developm*ent, 2013.

PÁGINAS WEB CONSULTADAS:

[38] ISO (International Organization for Standardization), url: http://www.iso.org, fecha de consulta: 03/10/2020

[39] IEEE Computer Society, url: http://www.computer.org, fecha de consulta: 04/10/2020

[40] IEEE (Institute of Electrical and Electronics Engineers), url: http://www.ieee.org, fecha de consulta: 04/10/2020

[41] PMI (Project Management Institute), url: http://www.pmi.org, fecha de consulta: 10/10/2020

[42] IPMA (International Project Management Association), url: http://www.ipma.org, fecha de consulta: 10/10/2020

[43] AENOR (Asociación Española de Normalización y Certificación), url: http://www.aenor.es, fecha de consulta: 24/10/2020

[44] Agile Spain, url: http://www.agile-spain.org, fecha de consulta: 07/11/2020

[45] Scrum Alliance, url: http://www.scrumalliance.org, fecha de consulta: 07/11/2020

[46] State of Agile, url: https://stateofagile.com, fecha de consulta: 07/11/2020

[47] The Standish Group, url: https://www.standishgroup.com, fecha de consulta: 14/11/2020

[48] Blog de trabajos de investigación de Ingeniería de Software de la Universidad "Unión Bolivariana", url: http://ingenieriadesoftware.mex.tl, fecha de consulta: 15/11/2020

[49] AM (Agile Modeling), url: http://www.agilemodeling.com, fecha de consulta: 21/11/2020

[50] Effective Practices for Software Solution Delivery, url: http://www.ambysoft.com, fecha de consulta: 21/11/2020

[51] DSDM (Dynamic System Development Method), url: http://www.dsdm.org, fecha de consulta: 21/11/2020

GLOSARIO

AACE: American Association of Cost Engineering

ACM: Association for Computing Machinery

AM: Agile Modeling

ANSI: American National Standards Institue

ASD: Adaptative Software Development

ASI: Análisis del Sistema de Información

AUP: Agile Unified Process

CCTA: Central Computer and Telecommunications Agency

CERN: Conseil Européen pour la Recherche Nucléaire

CMMI: Captability Maturity Model Integration

CP: Close Project (en PRINCE2)

CS: Control Stage (en PRINCE2)

CSI: Construcción del Sistema de Información

DAD: Disciplined Afile Delivery

DFD: Data Flow Diagram

DP: Direct Project (en PRINCE2)

DSDM: Dynamic Systems Development Method

DSI: Diseño del Sistema de Información

ECD: Entity Relationship Diagram

ELH: Entity Life Histoy

EVS: Estudio de la Viabilidad del Sistema

FDD: Feature Drive Development

GC: Gestión de la Configuración

GITT: Grado en Ingeniería de las Tecnologías de Telecomunicación

GP: Gestión de Proyectos

GSM: Global System for Mobile

IAS: Implantación y Aceptación del Sistema

ICB: IPMA Competence Basline

IEC: International Electrotechnical Commission

IEEE: Institute of Electrical and Electronics Engineers

IP: Internet Protocol

IP: Init Project (en PRINCE2)

IPMA: International Project Management Association

ISO: International Organization for Standardization

LDS: Logical Data Structure

LSD: Lean Software Development

MP: Magagement Product (en PRINCE2)

MSI: Mantenimiento del Sistema de Información

MVP: Minimum Viable Product

NGN: Next Generation Networking

PCM: Pulse Code Modulation

PERT: Program Evaluation and Review Techniques

PL: Planification (en PRINCE2)

PMBOK: Project Management Body of Knowledge

PMI: Project Management Institute

PSI: Planificación de Sistemas de Información

ROI: Return Of Investment

RUP: Rational Unified Process

SB: Stage Boundary (en PRINCE2)

SSADM: Structured Systems Analysis and Design Method

SU: Start Up (en PRINCE2)

TDD: Test Driven Development

TFG: Trabajo de Fin de Grado

TIC: Tecnologías de la Información y la Comunicación

VDSL: Very high-bit-rate Digital Subscriber Line

VoIP: Voice over IP

WBS: Work Breakdown Structure

WIK: Work In Process

XP: eXtreme Programming