Astic

Asociación Profesional de Cuerpos Superiores de Sistemas y Tecnologías de la Información de las Administraciones Públicas.

Temario para la preparación de la Oposición al Cuerpo Superior de Sistemas y Tecnologías de la Información de la Administración del Estado.

TEMAS ESPECÍFICOS – BLOQUE B
III. INGENIERÍA DE LOS SISTEMAS DE INFORMACIÓN

86. El ciclo de vida de los sistemas de información. Modelos del ciclo de vida.

AUTOR: Elisa Martín Ortega.
REVISOR: José Manuel Goberna Ruiz-Poveda.

Creación: septiembre 2023
Actualización:

ÍNDICE

INDICE DE ILUSTRACIONES

1 INTRODUCCION

En el presente documento se desarrolla el tema 086 del temario para la preparación de la Oposición al Cuerpo Superior de Sistemas y Tecnologías de la Información de la Administración del Estado y que pertenece al bloque B, parte III que tiene por título Ingeniería de los Sistemas de Información.

La interoperabilidad de los sistemas tiene un alcance suficiente para dedicar dos temas del temario. Esta primera parte estará enfocado al desarrollo de los siguientes apartados:

1. El ciclo de vida de los sistemas de información.
2. Modelos del ciclo de vida.

Este tema 086 tiene estrecha relación con un amplio conjunto de otros temas del bloque B parte III del temario.

2 EL CICLO DE VIDA DE LOS SISTEMAS DE INFORMACIÓN

2.1 Introducción

La definición de métodos, técnicas y procedimientos por parte de la ingeniería del software surge motivado por el desarrollo desordenado que las aplicaciones informáticas tuvieron en su origen, y que produjo lo que se denominó la **crisis del software** a comienzos de los años 70. Esta crisis se evidenciaba en aspectos como el incumplimiento de las expectativas, la falta de fiabilidad de las aplicaciones, el incumplimiento de los presupuestos y de la planificación temporal, la imposibilidad de evolucionar tecnológicamente las aplicaciones desarrollada, el alto coste de mantenimiento y en definitiva la ineficiencia respecto a todas las actividades alrededor de los desarrollos informáticos.

Para dar solución a estos problemas, surge la **Ingeniería del Software (SW)** que se define como el establecimiento y uso de principios, ya validados en otras disciplinas de la ingeniería, orientados a obtener de forma eficaz y eficiente, software fiable, portable y ajustado a las necesidades de las organizaciones. Es decir, para avanzar por las fases principales de definición, desarrollo y mantenimiento, y de apoyo en relación a la calidad, configuración, seguimiento, entre otras. Estos objetivos hacen necesario definir:

- Métodos que indiquen cómo construir técnicamente el SW.
- Herramientas que permitan el desarrollo del SW.
- Procedimientos que definan la secuencia en la se aplican los métodos y los controles, y que permitan el seguimiento, y en definitiva la gestión.

En consonancia con otras ingenierías, se asume la necesidad de definir los procesos, considerando los procesos de software como el conjunto de actividades que conducen a la creación de un producto software. Una vez definido el proceso de software, se define **ciclo de vida** como el marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso. Es decir, es el conjunto de etapas por las que atraviesa el sistema desde su concepción, hasta su retirada, pasando por su desarrollo y explotación.

El **ciclo de vida** o el **modelo de ciclo de vida (MCV)** de los sistemas es una representación simplificada y dada unas circunstancias concretas de un proceso de software que describe un conjunto de fases y las relaciones entre ellas. A este concepto se hace también referencia como ciclo de vida, modelo de desarrollo o modelo de proceso de software.

En el ciclo de vida de los sistemas de información se debe tener en cuenta los distintos enfoques actuales tanto del desarrollo como en la adquisición de software (SW):

- Desarrollo de SW utilizando metodologías clásicas y maduras, que pueden ayudarse de numerosos marcos, estándares y bibliografía.
- Desarrollo de SW utilizando metodologías ágiles, para las que no hay una publicación tan abundante de estándares y documentación.
- La adquisición de sistemas de información, teniendo en cuenta tanto los productos como los servicios.

En cualquier caso, todos los sistemas de información que sirvan de apoyo a las necesidades de las organizaciones para ofrecer sus productos y servicios, deberían tener definido su ciclo de vida ya que conocerlo permitirá maximizar la eficiencia y eficacia, además de reducir los riesgos.

En la definición del ciclo de vida deberá tenerse en cuenta todas las etapas que cubran desde la definición de requisitos hasta la retirada del sistema cuando este no sea útil para la organización, y por tanto constar de procesos para la adquisición, desarrollo y suministro de productos, proyectos y servicios del software, estableciendo pautas para su control y mantenimiento.

Por último, es necesario establecer formalmente una **metodología de desarrollo de software** que describe el trabajo a desarrollar en cada paso, los productos a obtener y las técnicas que se recomienda usar para generarlos. El uso de la metodología permite un trabajo repetible y por tanto permite hacer seguimiento y asegurar los objetivos planificados. En el caso de la Administración Pública la metodología oficial es Metrica v.3.

Este tema se centra en el concepto de ciclo de vida de los sistemas y sus distintos modelos.

2.2 OBJETIVOS

La definición de un ciclo de vida tiene como objetivo definir las actividades que se deben llevar a cabo por la organización, identificando, en este caso particular de los sistemas de información, las tareas para el logro eficaz y eficiente del desarrollo, explotación, mantenimiento, control de la calidad y de la gestión del riesgo, que proporcione la adecuada gestión del proyecto de forma que pueda asegurarse el cumplimiento de una planificación temporal y presupuestaria.

2.3 NORMATIVA Y MARCO JURÍDICO APLICABLE

Este apartado aplica a los dos epígrafes de este tema.

Aunque puede parecer que no existe normativa directamente relacionada con el ciclo de vida de los sistemas, atendiendo un marco más amplio, encontramos numerosas normas relacionadas:

Interoperabilidad

El Real Decreto 4/2010, de 8 de enero, por el que se regula el Esquema Nacional de Interoperabilidad en el ámbito de la Administración Electrónica, en adelante ENI, establece en su artículo 5 a la interoperabilidad como uno de sus principios básicos y cualidad integral, determinando que:

> *"La interoperabilidad se tendrá presente de forma integral desde la concepción de los servicios y sistemas y a lo largo de su ciclo de vida: planificación, diseño, adquisición, construcción, despliegue, explotación, publicación, conservación y acceso o interconexión con los mismos."*

En su artículo 26 de título Ciclo de vida de servicios y sistemas indica:

> *"La conformidad con el Esquema Nacional de Interoperabilidad se incluirá en el ciclo de vida de los servicios y sistemas, acompañada de los correspondientes procedimientos de control."*

Además, el ENI tiene a lo largo de su articulado tiene un enfoque muy importante en relación al ciclo de vida de los documentos y expedientes, así como de los datos.

En relación con el ENI, siempre es interesante conocer las recomendaciones recogidas en las Normas Técnicas de Interoperabilidad (NTI's).

El Real Decreto 203/2021, de 30 de marzo, por el que se aprueba el Reglamento de actuación y funcionamiento del sector público por medios electrónicos estableces en su artículo 17 de título "Directorios de aplicaciones reutilizables" que la Administración General del Estado mantendrá el Directorio general de aplicaciones para su libre reutilización.

Seguridad

En materia de seguridad se debe garantizar que los fabricantes mejoren la seguridad de los productos con elementos digitales desde la fase de diseño y desarrollo y durante todo el ciclo de vida, en este sentido el Real Decreto 311/2022, de 3 de mayo, por el que se regula el Esquema Nacional de Seguridad, en adelante ENS, incluye numerosas referencias a su aseguramiento de la seguridad, y en concreto identifica las medidas de seguridad relativas al desarrollo de las aplicaciones

mp.sw	Protección de las aplicaciones informáticas
mp.sw.1	Desarrollo de aplicaciones
mp.sw.2	Aceptación y puesta en servicio

Ilustración 1. Medida mp.sw de Protección de las aplicaciones informáticas del Anexo II del ENS

En relación con la seguridad aplican las recomendaciones establecidas en las guías u artículos publicados por el Centro Criptológico Nacional, como la Guía de CCN para la adquisición y control de calidad y el informe Lecciones aprendidas y recomendaciones de seguridad en el desarrollo de aplicaciones en relación al ciclo de vida de las mismas.

Contratación de sistemas de software

Orden EHA/1049/2008, de 10 de abril, de declaración de bienes y servicios de contratación centralizada.

Establece en su primer artículo de título "Declaración de suministros de contratación centralizada", que en el ámbito establecido en el artículo 206.1 del texto refundido de la Ley de Contratos del Sector Público aprobado por Real Decreto Legislativo 3/2011, de 14 de noviembre, se declaran de contratación centralizada los contratos de suministros de un conjunto enumerado entre los que se encuentra:

> *[…]*
> *c) Software de sistema, de desarrollo y de aplicación.*

Además, en su artículo 2 de título "Declaración de servicios de contratación centralizada", establece que en el ámbito establecido en el artículo 206.1 de la Ley de Contratos del Sector se declaran de contratación centralizada los contratos de servicios dirigidos al desarrollo de la Administración Electrónica cuyo presupuesto de licitación no supere 862.000 euros, I.V.A. excluido, cuyo objeto consista en:

> *1. Trabajos de consultoría, planificación, estudio de viabilidad, análisis, diseño, construcción e implantación de sistemas de información, y los mantenimientos de las aplicaciones desarrolladas bajo esta modalidad.*
> *2. Servicios de alojamiento en sus distintas modalidades, y los servicios remotos de explotación y control de sistemas de información que den soporte a servicios públicos de administración electrónica.*

Ley 9/2017, de 8 de noviembre, de Contratos del Sector Público, por la que se transponen al ordenamiento jurídico español las Directivas del Parlamento Europeo y del Consejo 2014/23/UE y 2014/24/UE, de 26 de febrero de 2014.

El ciclo de vida de los sistemas hay que tenerlo en cuenta tanto si se utiliza un modelo de desarrollo basado en capacidad propia en la administración, o como suele ser más habitual, en un modelo externalizado en el que será necesario acudir a la Ley de Contratos.

No se pretende hacer un análisis detallado de la Ley 9/2017, de 8 de noviembre, de Contratos del Sector Público, por la que se transponen al ordenamiento jurídico español las Directivas del Parlamento Europeo y del Consejo 2014/23/UE y 2014/24/UE, de 26 de febrero de 2014. que es objeto de otro tema, pero si se mencionará que, en primer lugar, en la contratación de herramientas de software y digitales, se debe tener en cuenta:

- La calificación del contrato como suministro o como servicio.
- La propiedad intelectual.
- Los modelos de licenciamiento.
- La continuidad del servicio.
- Los nuevos modelos de consumo de los servicios en la nube, e incluso la inteligencia artificial.

En base a esto, la Junta Consultiva de Contratación del Estado en su informe 58/2018, indica que, de forma general, los contratos que tienen por objeto la adquisición de programas de ordenador son contratos de suministro, con la excepción de que se trate de programas de ordenador confeccionados a medida, en cuyo caso constituyen contratos de servicios, sujetos ambos a la Ley 9/2017, de 8 de noviembre, de Contratos del Sector Público, y a los que serían de aplicación las previsiones señaladas del artículo 308. Este razonamiento es, además, aplicable a cualquier tipo de servicio o puesta a disposición de software informático, incluido, como recientemente recordaba la JCCPE, los contratos de prestación de servicios en la nube (Informe 13/2021 JCCPE).

Propiedad intelectual y Protección jurídica del SW

La protección de los programas de ordenador queda establecida, entre otros, en el artículo 1 de la Directiva 2009/24/CE, así como en las correspondientes transposiciones e implementaciones

nacionales. Este concepto queda definido de forma muy amplía en el texto del artículo 96.1 de la Ley de Propiedad Intelectual como:

> "*[…] toda secuencia de instrucciones o indicaciones destinadas a ser utilizadas, directa o indirectamente, en un sistema informático para realizar una función o una tarea o para obtener un resultado determinado, cualquiera que fuere su forma de expresión y fijación*".

Protección de datos personales

El Reglamento General de Protección de Datos establece aspectos a considerar tanto en la redacción y/o negociación de cualquier contrato de software como en la operativa ordinaria en materia de privacidad de responsables y encargados de tratamiento, además de obligar a trasladar el cumplimiento de la privacidad a lo largo de la cadena de subcontratación.

Respecto al encargado de tratamiento que en la administración suele ser un proveedor de servicio que será adjudicatario de un contrato, el RGPD le extiende las mismas obligaciones, responsabilidades y potenciales sanciones que se establecen para el responsable del tratamiento, por lo que deberá prestarse especial atención en la redacción de las condiciones de los pliegos de contratación a la cláusula de protección de datos, además de incorporar a sus procesos internos las medidas de seguridad necesarias.

Es importante prestar especial atención a los entornos de trabajo utilizados durante las fases de desarrollo y mantenimiento del SW, y a los sistemas de información secundarios, que pueden no tener una adecuada implantación de medidas técnicas y organizativas en relación a su nivel de riesgo y que pueden actuar de puerta de entrada a sistemas de información claves de la organización.

Reutilización de sistemas y aplicaciones de las Administraciones Públicas

En relación con la reutilización de los sistemas y aplicaciones, tanto la Ley 40/2015, de 1 de octubre, de Régimen Jurídico del Sector Público, como el ENI establecen que las Administraciones Públicas, con carácter previo a la adquisición, desarrollo o al mantenimiento a lo largo de todo el ciclo de vida de una aplicación, tanto si se realiza con medios propios o por la contratación de los servicios correspondientes, deberán consultar en el directorio general de aplicaciones de la Administración General del Estado para su libre reutilización, si existen soluciones disponibles para su reutilización, que puedan satisfacer total o parcialmente las necesidades, mejoras o actualizaciones que se pretenden cubrir, y siempre que los requisitos tecnológicos de interoperabilidad y seguridad así lo permitan.

Las conclusiones con respecto al resultado de dicha consulta al directorio general se incorporarán en el expediente de contratación y reflejarán, en su caso, que no existen soluciones disponibles para su reutilización que puedan satisfacer total o parcialmente las necesidades, mejoras o actualizaciones que se pretenden cubrir.

En el caso de existir una solución disponible para su reutilización total o parcial, la justificación de la no reutilización se deberá realizar en términos de eficiencia conforme a lo establecido en el artículo 7 de la Ley Orgánica 2/2012, de 27 de abril, de Estabilidad Presupuestaria y Sostenibilidad Financiera.

2.4 ESTÁNDARES NACIONALES E INTERNACIONALES

Este apartado aplica a los dos epígrafes de este tema.

A continuación, se enumeran algunos de los principales estándares y buenas prácticas en relación con la materia de este tema:

- Norma ISO/IEC 12207:2017 - Ingeniería de Sistemas y Software - Procesos de ciclo de vida del software, (Systems and software engineering — Software life cycle processes).
- Norma ISO/IEC/IEEE 15288:2023 Ingeniería de Sistemas y Software - Procesos de ciclo de los sistemas, (Systems and software engineering — System life cycle processes).
- Norma ISO/IEC 15289:2019 – Ingeniería de Sistemas y Software. Documentación. (Systems and software engineering — Content of life-cycle information items (documentation)

Además, el ciclo de vida descrito por la ISO/IEEC 12207 no incluye algunos aspectos de gestión concretos que son importantes tener en cuenta, y que se enumeran a continuación:

- La gestión de la calidad especificado por ISO/IEC 9001.
- La gestión de los servicios especificados por ISO/IEC 20000-1
- La gestión de la seguridad de la información seguridad especificado por ISO/IEC 27000 .
- La evaluación de los procesos especificado por ISO/IEC 33002:2015.

A continuación, se describen las normas ISO/IEC con más relevancia en este tema.

2.4.1 ISO/IEC 12207:2017

La norma **ISO/IEC 12207:2017** establece un marco de trabajo común en relación con el conjunto completo de etapas del **ciclo de vida del software**, con una terminología bien definida y un amplio reconocimiento e implantación en el ámbito de los sistemas de información y las comunicaciones. Además, incorpora los procesos que permiten la mejora continua. Estos procesos tienen doble alcance, tanto el desarrollo de software, como la adquisición de sistemas de información.

El estándar ISO/IEC 12207 es un marco de referencia que engloba los procesos, actividades y tareas involucradas en el desarrollo, explotación y mantenimiento de un producto software, es decir, abarca todas las etapas toda la vida del sistema desde la definición de requisitos hasta la retirada del sistema, bien sea por obsolescencia, por sustitución de otro sistema, o por cambios fundamentales en la misión de la organización.

Es aplicable a la adquisición de sistemas, productos y servicios software, al suministro, desarrollo, operación y mantenimiento de productos software. Esta norma además esta creada para ser utilizada tanto por personas adquirientes de sistemas, productos y servicios de software, como para desarrolladores, operadores, responsables de mantenimiento, administradores, responsables de aseguramiento de calidad y usuarios finales.

El estándar 12207 consta de procesos para la adquisición y suministro de proyectos y servicios del software, estableciendo pautas de control de calidad y mantenimiento integral, además de ofrecer un lenguaje común que facilita las interrelaciones entre los diferentes agentes (compradores, proveedores, desarrolladores, personal de mantenimiento, operadores, gestores y técnicos) que pueden beneficiarse del uso de este estándar.

En la **definición de los procesos** se tienen en consideración los siguientes aspectos:

- La modularidad del mismo, de forma que un proceso individual se dedica a una única función, logrando alta cohesión y bajo acoplamiento.
- Que la responsabilidad sobre el proceso sea individual.
- Pueden afectar a toda o a una parte de la organización.
- Los procesos se organizan por actividades y estas, a su vez, se implementan con tareas, que son un conjunto elemental o atómico de acciones y consume unas entradas (datos, información, control) y produce unas salidas (datos, información y control)

Los **procesos se clasifican en tres tipos**: primarios, de soporte y de organizacionales. Los procesos de soporte y de organización deben existir independientemente de la organización y del proyecto. Los procesos principales se aplican dependiendo de las circunstancias de la organización y son utilizados por los procesos de soporte y de la organización.

Los **procesos principales** del estándar 12207 son los siguientes:

- La adquisición tanto de productos como de servicios de software.
- El propio suministro realizado por el agente que proporciona el producto o servicio al cliente.
- Las tareas del propio desarrollador.
- Las tareas de operación y explotación del producto o servicio en su entorno final o productivo.
- Las tareas de mantenimiento posteriores a la implantación del sistema en producción tanto en aspectos correctivos, evolutivos, adaptativos o perfectivos.

Además, cuenta con un conjunto de **procesos de soporte o apoyo**:

- El registro de la documentación en el repositorio establecido para recoger la documentación completa del ciclo de vida tanto del desarrollo como del sistema de información.
- La gestión de la configuración, así como la aplicación de los procedimientos técnicos y administrativos de los sistemas y del software.

- El control de la calidad de forma que se asegure de forma objetiva que los productos y procesos son conformes a los procedimientos establecidos.
- Las actividades de verificación interna de evolución del proyecto.
- La validación formal del cumplimiento de los requisitos establecidos para el sistema o producto software.
- Revisión conjunta del cliente y proveedor del sistema del sistema o producto.
- El proceso de control de auditoría para determinar el nivel de adecuación o cumplimiento de los requerimientos, planes y contratos el sistema.
- La gestión integral y resolución de incidencias y problemas.

Por último, define un conjunto genérico de **procesos de la organización** que deben ser tenidos en cuenta de forma integral junto con los anteriores:

- La administración o gestión de los procesos durante el ciclo de vida.
- Las actividades para establecer la infraestructura hardware, software, instalaciones, comunicaciones, etc, necesaria para llevar a cabo los procesos.
- Las actividades de mejora para establecer indicadores y métricas que permitan seguimiento de los procesos.
- Los recursos humanos de la organización, la concienciación y formación, que deberá ser continua y enfocada a los distintos perfiles profesionales de la organización y su papel en cada uno de los procesos.

Esta norma indica **qué se debería conseguir, sin determinar cómo lograrlo**, ni el orden de los propósitos y salidas de los procesos. Una de las ventajas de la norma es esta flexibilidad, facilitando la selección de los procesos en función de las necesidades de la organización.

Existen dos formas para confirmar que una determinada implementación de ciclo de vida se ajusta a este estándar, y en ambas deben existir evidencias de su implementación y cumplimiento:

- Conformidad completa cuando se han implementado todos los procesos.
- Conformidad a medida o parcial, cuando se utiliza un subconjunto de procesos específicos.

Para la aplicación de este estándar se debe tener en cuenta entre otros, los siguientes aspectos:

- Fases del ciclo de vida del sistema: en general un ciclo de vida, cubre las fases de requisitos, análisis, diseño, documentación y prueba.
- Tipo de software: el tipo de software del proyecto debe de ser determinado, como nuevo software, firmware, reutilización de uno existente, software embebido, software independiente, etc.
- Rol en el ciclo de vida: el usuario dentro del estándar se debe determinar si es un comprador, un proveedor, un desarrollador, un operador o se dedica a las tareas de mantenimiento.
- Modelo de desarrollo: dentro del proyecto se debe de identificar uno o más modelos de desarrollo, como: cascada, incremental, evolutivo, espiral, etc.
- Características del proyecto: los requerimientos y especificaciones del producto o servicio dominan la determinación y selección de procesos, actividades y tareas.
- Las políticas organizacionales involucradas deben de ser identificadas y analizadas para poder determinar la relevancia del proyecto. Se deben de identificar y analizar las leyes nacionales y reglamentos de seguridad pública, la salud y el medio ambiente entre otros, que son aplicables.
- Respecto a la documentación, se va a determinar qué resultados son necesarios para las actividades y tareas, como deben de ser combinados, empaquetados y distribuidos. Se debe de asegurar que la operación y el soporte personal están involucrados en la determinación de la documentación necesaria.
- Incluir varios procesos y tareas que lleven a cabo evaluaciones, verificaciones y auditorías, que podrá realizarse dentro de un proceso o entre procesos, y que, en todo caso, deberán comprobar evidencias y tener como resultado documentación.

Actualmente, existe un número alto de metodologías de desarrollo flexibles, que permiten hacer uso en función de las circunstancias de la organización y los equipos de trabajo, sin necesidad de forzar procesos complejos que no aporten valor añadido al resultado del proyecto, esto las convierte en herramientas muy útiles para todos los perfiles y roles que trabajan en el desarrollo/adquisición de un sistema de información.

En todo caso la selección de cualquier de esas metodologías, no reduce la necesidad de la definición de un ciclo de vida del sistema de información.

2.4.2 ISO/IEC 15288:2015

La norma ISO/IEC 15288:2015 crea un marco que detalla el **modo de ejecución de los procesos de descripción del ciclo de vida** para la generación del desarrollo de software, además aporta un conjunto de indicadores para ser aplicados en cada etapa del ciclo de desarrollo dentro de un proceso integral de mejora continua y eficiencia. Tiene un enfoque dirigido al desarrollo de todo tipo de software.

Esta norma comprende **25 procesos que tienen 123 resultados derivados de 403 actividades**, lo que hace la aplicación de esta norma una labor muy costosa de aplicar casi por cualquier organización. Los procesos afectan al diseño, aplicación, análisis de requerimientos, procesos de verificación y validación, etc. Los procesos los divide en las siguientes cuatro **categorías**:

- Técnico.
- Proyecto.
- Acuerdo.
- Empresa.

Algunos de los procesos más destacados son:

- Definición de requisitos.
- Análisis de requisitos.
- Diseño de arquitectura.
- Implementaciones.
- Integraciones.
- Verificaciones.
- Validaciones.
- Mantenimientos.
- Eliminaciones o retiradas.

En la norma se incorporan, respecto a la ISO/IEC 12207, cambios en relación a procesos relacionados con la arquitectura, la gestión del riesgo y la gestión de la configuración, actualiza conceptos como iteración, recursividad, calidad etc., e incorpora un anexo con la relación con la ingeniería de sistemas basada en modelos o MBSE.

2.4.3 ISO/IEC 15289:2019

El propósito de la norma ISO/IEC 15289:2019 es proporcionar los requisitos en relación a la documentación generada y su mapeo con los procesos del ciclo de vida de los sistemas de información y el software, Es importante resaltar que este propósito no lo liga a ningún modelo, metodología o técnica concreta, y excluye de su alcance:

- El formato y contenido.
- Mecanismos de entrega, registro y publicación.
- Aspectos concretos de cada negocio en el que la organización lleve a cabo su misión.
- Cualquier producto que no quede en el ámbito propio de la tecnología

2.4.4 ISO/IEC 15504

La norma ISO/IEC 15504 define que todo modelo de evaluación de procesos debe definir:

- El modelo de procesos de referencia.
- Los niveles de capacidad y atributos de los procesos.

Los niveles de capacidad para todo modelo de evaluación de procesos podrán tener desde el 0 y al menos hasta el nivel 1 de los siguientes niveles de capacidad estándar, además, para cada nivel existirán unos atributos de procesos estándar que ayudan a evaluar los niveles de capacidad.

- Nivel 0: Incompleto.
- Nivel 1: Realizado.
- Nivel 2: Gestionado.
- Nivel 3: Establecido.
- Nivel 4: Predecible.

- Nivel 5: En optimización.

La norma establece un marco y requerimientos útiles en cualquier etapa del proceso de evaluación del proceso de desarrollo de software, ofrece modelos utilizados en la evaluación de organizaciones, proporcionar guías para definir las competencias del encargado de realizar el proceso de evaluación, y dispone de métodos para evaluar, mejorar y determinar la capacidad de los procesos.

En relación con el resto de normas referenciadas en el tema, la norma 15504 se ciñe a las fases del ciclo de vida del desarrollo, mantenimiento y operación de los proyectos software de la ISO 122007, proporciona un procedimiento de evaluación de procesos para el ciclo de vida, definidos en el estándar ISO/IEC 15288, y aporta un modelo de evaluación de procesos para los procesos de servicios TIC, definidos en el estándar ISO/IEC 20000. Además, ISO forma parte del modelo CMMI, siendo ambos compatibles con 15504.

2.4.5 VENTAJAS DE LA APLICACIÓN DE LAS NORMAS

Las principales **ventajas del uso de las metodologías ISO 12207 y 15504** son las siguientes:

- Permiten la evaluación, seguimiento y mejora de los procesos.
- Mejora la eficacia y eficiencia del proceso de desarrollo.
- Permiten mantener registros de gestión, procesos y procedimientos.
- Ayudan a mejorar el producto final y con ello la satisfacción de clientes/usuarios finales.
- Mejora la calidad del desarrollo y, por tanto, reducen las incidencias, errores y fallos en la producción.
- Definen los procesos del ciclo de vida del desarrollo de software, su mantenimiento, operación y explotación con los sistemas de software.

2.5 INGENIERÍA DEL SOFTWARE

La Ingeniería del Software (SW) se define como el establecimiento y uso de los principios de la ingeniería orientados a obtener de manera económica software que sea fiable, funcione eficientemente sobre maquinas reales y, por supuesto, ajustado a las necesidades de las organizaciones.

La ingeniería del software **se compone de**:

- Métodos que indiquen cómo construir técnicamente el SW, y que cubre las etapas de planificación, análisis, diseño, codificación, pruebas y mantenimiento.
- Herramientas que dan soporte a los métodos enumerados anteriores y que, si son adecuadas, se integran unas con otras.
- Procedimientos que definan la secuencia en la se aplican los métodos, los controles de calidad, la coordinación de cambios y elaboración de documentación que permite el seguimiento, y en definitiva la gestión del proyecto. Son el nexo entre los métodos y las herramientas.

Se pueden identificar **tres fases genéricas en los procesos de desarrollo** de software:

- Definición que comprende:
  - Análisis del sistema.
  - Planificación del proyecto.
  - Análisis de requerimientos que detalla el dominio y los requisitos funcionales del software.
- Desarrollo que comprende:
  - Diseño del software.
  - Codificación o programación.
  - Las pruebas de distinto tipo.
- Mantenimiento que, según la metodología elegida contempla un desglose más o menos detallado, por ejemplo, haciendo referencia a Métrica v3, comprendería:
  - Mantenimiento correctivo.
  - Mantenimiento evolutivo.
  - Mantenimiento adaptativo.
  - Mantenimiento perfectivo.

2.6 PROTECCIÓN DE DATOS EN LOS ENTORNOS DE TRABAJO EN EL CICLO DE DESARROLLO DEL SOFTWARE

La ingeniería de sistemas recomienda utilizar entornos diferenciados para las distintas fases del ciclo de desarrollo, es decir realizar las labores de desarrollo en el entorno de desarrollo, realizar las pruebas en el entorno de preproducción y desplegar aplicaciones y servicios en el entorno de producción.

El responsable y el encargado de tratamiento en cumplimiento del artículo 32 del RGPD, que establece que aplicarán medidas técnicas y organizativas apropiadas para garantizar un nivel de seguridad adecuado al riesgo para los derechos y libertades de los interesados, deberán establecer la diferenciación de entornos y limitar la exposición de datos personales reales, y/o que estén en los sistemas de producción, en las fases de desarrollo y pruebas.

En esta misma línea, el Supervisor Europeo de Protección de Datos (EDPS) en sus directrices "*Juilines on the protection of personal data in IT governance and IT management of EU institutions*" indica:

> *"80 En la fase de prueba, debe evitarse el muestreo de datos personales reales, ya que dichos datos no pueden utilizarse para fines para los que no fueron recogidos y su uso en entornos de prueba puede dar lugar a que personas no autorizadas dispongan de datos personales."*
>
> *"81 Siempre que sea posible, deben utilizarse datos de prueba creados artificialmente, o datos de prueba derivados de datos reales, de modo que se conserve su estructura, pero no contengan datos personales reales."*
>
> *"82 Cuando un análisis minucioso y prudente muestra que los datos de prueba generados no pueden proporcionar suficiente garantía de la validez de las pruebas, debe tomarse una decisión exhaustiva y documentada, que defina qué datos reales se utilizarán en la prueba, lo más limitado como sea posible, las salvaguardias técnicas y organizativas adicionales que se establezcan en el entorno de las pruebas. Las categorías especiales de datos sólo pueden utilizarse en las pruebas de datos reales con el consentimiento explícito de las personas afectadas."*

Conforme al **principio de minimización de datos y el principio de protección de datos desde el diseño y por defecto**, cuando sea posible debe evitarse la utilización de datos personales en entornos de desarrollo y preproducción, o cualquier otro entorno de pruebas.

Además, las pruebas de software con datos personales son, o forman parte de, tratamientos de datos personales y el responsable del tratamiento debe cumplir con todas las obligaciones que se desprenden del RGPD.

Siempre que sea posible se utilizarán datos sintéticos para evitar el tratamiento de datos personales en pruebas de desarrollo, y cuando esto no sea posible deberá documentarse mediante un análisis de necesidad y proporcionalidad, y en todo caso aplicar las medidas técnicas y organizativas que sean necesarias conforme al artículo 32 del RGPD y de acuerdo con el riesgo del tratamiento.

Hay que tener en cuenta que en ocasiones el riesgo en el entorno de desarrollo y preproducción es incluso mayor al riesgo del tratamiento en el entorno de producción, por ejemplo, por los siguientes motivos:

- Es frecuente que haya un mayor número de personas trabajando y con acceso a ese entorno, incluso de distintos adjudicatarios, lo que implicaría un número más alto de encargados de tratamiento.
- Puede ocurrir que los sistemas se alojen en CDPs diferentes a los entornos de producción.
- Por la dinámica de estos entornos de desarrollo y preproducción existirá mayor incertidumbre sobre la fiabilidad del código.
- Es frecuente realizar pruebas con nuevas tecnologías.

2.7 SEGURIDAD EN EL CICLO DE VIDA DEL SOFTWARE

Inicialmente, el ciclo de vida del desarrollo de los sistemas de información transcurría prácticamente con una total ausencia de seguridad, es decir:

- Sin requisitos ni controles de seguridad.
- Desarrolladores con insuficientes o nulos conocimientos de desarrollo seguro.
- Sin verificaciones de los parámetros de entrada a nivel de seguridad.
- Sin herramientas automáticas para cubrir defectos del código.
- Sin realizar pruebas de seguridad manuales.
- Sin tener en cuenta el flujo correcto de la lógica de navegación.
- Sin verificar las vulnerabilidades de las librerías dependientes.
- Incorrecto tratamiento del control de versiones en producción.
- Sin planificar una correcta gestión de incidentes de seguridad.

Es decir, en el ciclo de vida de vida del desarrollo del software (SDLC) se actuaba únicamente de forma reactiva cuando existían problemas materializados.

El SDLC es el marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento del software. En este ciclo de vida se utilizan herramientas que permiten tanto el desarrollo (por ejemplo, Visual Studio, Eclipse), el control de código fuente (por ejemplo, con TFS, Git), las pruebas automatizadas (por ejemplo, con ALM) y la gestión de incidencias y peticiones (por ejemplo, con JIRA).

El **ciclo de desarrollo seguro de aplicaciones** S-SDLC (*Secure Software Development Life Cycle*) es un conjunto de procesos y procedimientos diseñados para facilitar la identificación de los puntos débiles de seguridad en algunas o en todas las etapas del SDLC, mediante la aplicación de controles de seguridad que sirvan para tomar las acciones necesarias con el fin de asegurar el software lo máximo posible seguridad durante todo el ciclo de vida de la aplicación, desde su diseño hasta su puesta en producción y durante todo el tiempo de servicio.

Existen muchos modelos de definición de fases para estos procesos, pero las fases comúnmente identificadas son:

- Diseño: identificar los requisitos de seguridad, establecer políticas y estándares de seguridad para la aplicación, y diseñar la aplicación con perspectiva de seguridad aplicando todas las recomendaciones del diseño seguro.
- Implementación: codificar la aplicación teniendo en cuenta los requisitos y estándares de seguridad, así como todas las recomendaciones relativas al desarrollo seguro.
- Despliegue: despliegue de la aplicación en un primer entorno de ejecución similar a producción donde verificar automáticamente pruebas funcionales, requisitos de seguridad y otros controles.
- Testing: realización de las pruebas funcionales y de seguridad para detectar errores y vulnerabilidades. Normalmente suelen ser pruebas de verificación manuales en un entorno semejante al de producción.
- Operaciones: monitorizar, detectar amenazas y actualizar continuamente la seguridad de la aplicación. En esta fase también se tienen en cuenta los sistemas que soportan la aplicación, así como su configuración de seguridad/bastionado.

El ENS es la norma obligatoria en la Administración para medir y evaluar el grado de cumplimiento de los sistemas a los propios requisitos establecidos en el ENS. Otros estándares similares son PCI- DSS o NIST.

Los modelos de madurez de desarrollo del software seguro proporcionan un marco organizado de requisitos de seguridad cuyo nivel de cumplimiento permite evaluar la posición de madurez de la implementación de la seguridad en: una aplicación, un proyecto o una organización. Los modelos de madurez más utilizados son: OWASP SAMM, ISO 33000.

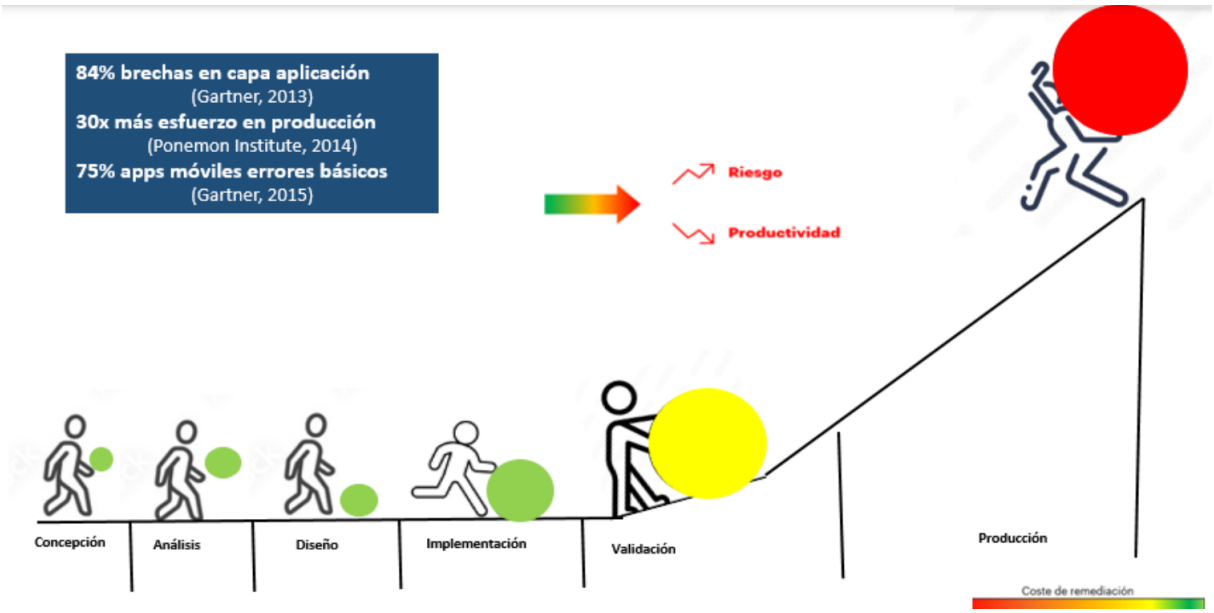

Ilustración 2. Ventajas en la aplicación de metodologías de desarrollo seguro.

Si la incorporación de la seguridad sólo se realiza en las fases avanzadas del ciclo de vida del desarrollo de las aplicaciones, seguirá existiendo un elevado riesgo, un mayor coste de recuperación de la operativa de los sistemas y una disminución de la productividad, debido a que la incorporación de controles y medidas de seguridad es más costosa, y requiere un mayor esfuerzo del equipo de desarrollo, lo que degrada su productividad e implica la asunción de riesgos mayores por parte de la dirección del proyecto.

Las consecuencias de no aplicar suficientes controles de seguridad en las aplicaciones son graves, numerosas, y variadas, siendo las más comunes:

- Pérdida o robo de información sensible o personal que conllevan juicios y multas en daños y reparaciones, así como una importante pérdida de imagen comercial.
- Interrupción de los servicios mediante ataques DDoS causando pérdida de ingresos, clientes insatisfechos y daños en la reputación.
- Violación de cumplimiento normativo que es causa de sanciones y multas.
- Dificultad para obtener seguros de responsabilidad civil al considerarse que la ausencia de medidas de seguridad adecuadas aumenta el riesgo de incidentes.
- Pérdida de competitividad en el mercado.
- Daño en los sistemas y servicios que requirieran altos costes en reparaciones y recuperaciones de la integridad.
- Riesgo de extorsión para recuperar los datos o para evitar la divulgación de información sensible.

En resumen, la seguridad en las aplicaciones conlleva una serie de **retos** a lo largo de su ciclo de vida como son:

- Disponer de herramientas para modelados de la amenaza.
- Crear catálogos de requisitos de seguridad.
- Obtener guías de desarrollo seguro.
- Crear los procesos para la gestión de vulnerabilidades.
- Desarrollo de técnicas y habilidades en el uso de herramientas para el escaneo y testeo automático.
- Realizar pruebas de seguridad manuales que suponen un alto coste.
- Planes de formación en desarrollo seguro individualizados con diseños curriculares.

El alcance de estos retos, hace emerger las siguientes **oportunidades**:

- Procesos y procedimientos mejor definidos.
- Aplicaciones de mayor valor.
- Mayor velocidad de desarrollo.
- Mejores herramientas de análisis y control.
- Menos errores y menos coste en correcciones.
- Pruebas antes disponibles, predefinidas y menos costosas.

- Automatización en la identificación de las vulnerabilidades.
- Mayor conocimiento por parte de los desarrolladores en desarrollo seguro.
- Obtención de métricas para mejora de los procesos.
- Integración de elementos de mayor valor como la automatización y la ayuda de procesos basados en inteligencia artificial.

Para detectar, eliminar y/o mitigar las vulnerabilidades de una aplicación se pueden utilizar los siguientes análisis de seguridad en diferentes fases del ciclo de vida del desarrollo de software (SDLC):

- Análisis estático del código fuente.
- Análisis de los componentes y librerías de terceros de los que depende la aplicación.
- Análisis dinámico de la aplicación en un entorno pre-productivo.

En todo caso, es importante tener en cuenta que, a pesar del incremento en la inversión en materia de seguridad, esta no podrá lograrse al 100%.

2.8 CONCLUSIONES

La Ingeniería del Software (SW) se define como el establecimiento y uso de principios, ya validados en otras disciplinas de la ingeniería, orientados a obtener de forma eficaz y eficiente, software fiable, portable y ajustado a las necesidades de las organizaciones.

Para avanzar por las fases principales de definición, desarrollo y mantenimiento, y de apoyo en relación a la calidad, configuración, seguimiento, entre otras, es necesario definir métodos, herramientas y procedimientos.

Los procesos de software se consideran al conjunto de actividades que conducen a la creación de un producto software.

El ciclo de vida es el marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software.

Por último, la metodología de desarrollo de software describe el trabajo a desarrollar en cada paso, los productos a obtener y las técnicas que se recomienda usar para generarlos.

La definición de un ciclo de vida tiene por objeto definir las actividades que se deben llevar a cabo por la organización, identificando, en este caso particular de los sistemas de información, las tareas para el logro eficaz y eficiente del desarrollo, explotación, mantenimiento, control de la calidad y de la gestión del riesgo, que proporcione la adecuada gestión del proyecto de forma que pueda asegurarse el cumplimiento de una planificación temporal y presupuestaria.

Dos normas muy importantes al hablar del ciclo de vida de los sistemas son la norma ISO/IEC 12207:2017 y la ISO/IEC 15288:2015 que establecen el conjunto completo de etapas del ciclo de vida del software y el modo de ejecución de los procesos de descripción del ciclo de vida para la generación del desarrollo de software.

El ciclo **de desarrollo seguro de aplicaciones** es el conjunto de procesos y procedimientos diseñados para facilitar la identificación de los puntos débiles de seguridad en algunas o en todas las etapas del desarrollo.

3 MODELOS DEL CICLO DE VIDA

3.1 Introducción

Considerando los **procesos de software** como el conjunto de actividades que conducen a la creación de un producto software, esta definición podría ser tan variada como la combinación entre los posibles productos software y los tipos de organizaciones existentes, es decir no existe un proceso óptimo, sino que este será tanto más válido cuanto más flexible sea y se adapte a las necesidades de la organización, el desarrollo a realizar y el equipo informático, siendo más útil un proceso muy estructurado cuando se trabaje para un sistema crítico y flexible y ágil cuando se trabaje en un sistema de requerimientos muy cambiantes.

En todo caso, en general todos estos procesos tienen las actividades básicas de:

- Especificación y toma de requerimientos.
- Desarrollo
- Validación y pruebas.
- Evolución y mantenimiento.

Los procesos de software se pueden mejorar utilizando estándares que ofrecen profesionalización, la eficiencia y eficacia en el uso de los recursos y la aplicación de mejores prácticas. En relación a esta materia son estándares ampliamente extendidos:

- El Modelo de Capacidad de Madurez Integrado (CCMI).
- El estándar SPICE (ISO/IEC 15504)
- ISO 9001:200

Una vez definido el proceso de software, se define el **modelo de ciclo de vida** de los sistemas como una representación simplificada y dada unas circunstancias concretas de un proceso de software que describe un conjunto de fases y las relaciones entre ellas. Este concepto se nombra también como ciclo de vida, modelo de desarrollo o modelo de proceso de software.

3.2 OBJETIVO

Un modelo de ciclo de vida debe cubrir los siguientes objetivos básicos:

- Definir las actividades a realizar y el orden de las fases del proceso software.
- Establecer los criterios de transición para pasar de una fase a la siguiente.
- Proporcionar puntos de control temporales y presupuestarios para la gestión del proyecto.
- Asegurar la consistencia con el resto de los sistemas de información de la organización.

3.3 TIPOS DE MODELOS DE CICLO DE VIDA

Existen diferentes clasificaciones de modelos según distintos puntos de vista, por ejemplo, tradicionales (orientados a proyecto), alternativos (orientados a producto), o bien modelos de proceso prescriptivo vs especializado, etc.

En todo caso, se podría decir que destacan por su conocimiento en el sector los siguientes modelos:

- Modelos en cascada.
- Modelos basados en prototipos e iterativos.
- Modelos basados en componentes.
- Modelos ágiles.

La decisión para utilizar un modelo u otro puede ser en base a distintos criterios, por ejemplo:

- Si el equipo de desarrollo posee la experiencia suficiente y los requisitos están perfectamente fijados, es recomendable un ciclo de vida en cascada.

- El ciclo de vida en espiral será el elegido si durante el desarrollo del proyecto se va a tener un enfoque especial en los riesgos e imprevistos.
- En el caso de tener que ir afinando con el cliente cada etapa, con el objetivo de mostrarle su utilidad, el ciclo de vida basado en prototipos será el elegido.

Estos criterios son del todo teóricos, en la realidad las circunstancias particulares de cada proyecto software, hace difícil que se ajusten completamente a un modelo, y lo habitual es trabajar con ellos de forma combinada.

3.4 MODELOS TRADICIONALES

3.4.1 MODELO CODIFICAR Y CORREGIR

El modelo codificar y corregir, conocido como “Code and Fix” es un modelo utilizado en las primeras décadas del desarrollo del software en que comienza desarrollando código y sobre este comprobar el cumplimiento de requisitos, diseño, validación, y mantenimiento, es decir contiene dos pasos básicos:

- Desarrollar código.
- Corregir los errores.

Esta forma de trabajar evidencio rápido los problemas de su uso:

- Mala estructura del código.
- Incumplimiento de las expectativas del cliente.
- Mantenimiento costoso y difícil.

3.4.2 MODELO POR ETAPAS

El modelo por etapas, conocido como “*Stage Wise*” considera el siguiente conjunto de etapas sucesivas:

- Planificación.
- Especificaciones de operación.
- Especificaciones de codificación.
- Codificación.
- Prueba de cada unidad.
- Prueba de integración.
- Eliminación de problemas.
- Evaluación del sistema.

3.4.3 MODELO EN CASCADA

El modelo en cascada está basado igual que el anterior en etapas, pero considera las etapas anteriores como fases de procesos separados, tales como la especificación de requerimientos, el diseño del software, la implementación, las pruebas, etc., de forma que considera la posibilidad de realizar ciclos.

El modelo en cascada se compone de una serie de fases que se suceden secuencialmente, generándose en cada una de ellas unos resultados que serán necesarios para iniciar la fase siguiente. Es decir, la evolución del producto software se produce a través de una secuencia ordenada de transiciones de una fase a la siguiente, según un orden lineal.

El modelo del ciclo de vida en cascada está regido por la documentación, es decir, la decisión del paso de una fase a la siguiente se toma en función de si la documentación asociada a dicha fase está completa o no.

Este modelo ha sido el más utilizado, por lo que se conoce como ciclo de vida clásico, y por sus características también es conocido como ciclo predictivo o basado en la planificación.

El número de fases de este modelo es irrelevante siempre que se produzcan secuencialmente, aunque las más habituales son:

- Planificación del sistema temporal, presupuestaria y de composición del equipo de trabajo.
- Definición y análisis de los requisitos funcionales, de rendimiento del sistema y de interfaz con otros sistemas, es decir su objetivo es definir el qué hacen los sistemas.
- Diseño integral en el que a partir del análisis realizado antes se define la estructura y todos los detalles para iniciar la codificación, es decir su objetivo es definir el cómo.
- Desarrollo de las especificaciones realizadas en la fase de diseño en los lenguajes de programación, junto con el resto de aspectos, como base de datos, etc.
- Conjunto de pruebas e integración que aseguren que es correcta la lógica del programa y cubre la funcionalidad prevista.
- Implementación del proyecto y su aceptación por parte de los usuarios.
- Mantenimiento del sistema hasta el final de su vida útil, este puede ser correctivo, adaptativo, perfectivo y evolutivo.

Las principales ventajas de este modelo su claridad, la facilidad para definir sus hitos, estimación y seguimiento del progreso, y que proporciona entregables intermedios que forman el conjunto final del producto (documentos de análisis y diseño etc.).

El principal inconveniente de este modelo es el que sus etapas sean secuenciales, ya que en la vida real es habitual que se produzcan interacciones, incluso en ocasiones es necesario realizar etapas en paralelo, además la fijación de requisitos en una etapa tan temprana, aunque deseable, no suele ser posible y, por último, aunque se incluyan actividades de verificación y validación, los problemas suelen detectarse tarde.

3.4.4 MODELO EN V

El modelo en “V” es similar al ciclo de vida en cascada, pero se caracteriza por tener dos tiempos diferenciados:

- Actividades de desarrollo (rama descendente).
- Actividades de prueba (rama ascendente).

La ventaja del ciclo en V son las mismas que las del modelo en cascada, además de facilitar la realización de las pruebas lo antes posible. Su principal inconveniente, al igual que en cascada, es su rigidez.

3.4.4.1 MODELOS BASADOS EN PROTOTIPOS

El modelo basado en prototipos se puede desglosar, a su vez en distintos tipos, pero todos permiten construir versiones tempranas o de ciertos aspectos del software en desarrollo para poder evaluarlo con los usuarios/clientes.

No hay que confundir el prototipo con otro concepto común en tecnología que es el producto mínimo viable. El prototipo se utiliza para validar un concepto o trabajo, mientras que el producto mínimo viable es más avanzado, es una versión final del sistema, pero con una funcionalidad reducida.

El uso de prototipo es utilizado con los siguientes motivos:

- Determinar si los requisitos son razonables o las especificaciones son completas.
- Construir versiones en etapas tempranas del desarrollo y que pueden ser evaluados por los usuarios.
- Ayudar a comprender los requisitos y por tanto a asegurar el entendimiento entre el equipo de desarrollo y los usuarios funcionales.
- Comprobar si el diseño es posible desarrollarlo.

Se presentan a continuación distintos modelos basados en prototipos, pero que en común presentan como ventaja principal que son un mecanismo ideal para obtener requisitos cuando no están claros, y como inconvenientes que el usuario se forme falsas expectativas al ver el prototipo y limitar el sistema real al mantener decisiones de implementación tomadas con el único objetivo de agilizar el desarrollo del prototipo.

3.4.4.1.1 MODELO DE PROTOTIPADO RÁPIDO

El Modelo de prototipado rápido tiene como objetivo generar un tipo de prototipo en una fase muy inicial del desarrollo, invirtiendo poco esfuerzo y presupuesto.

Las principales ventajas de este modelo son que permite la reutilización, el paralelismo en el desarrollo y facilidad para integrar la tecnología orientada a objetos (basada en componentes).

Este tipo de prototipo es muy importante que se deseche para evitar problemas futuros surgidos de su aprovechamiento, es decir debe construirse el sistema partiendo de cero y con los criterios de calidad y mantenimiento necesarios. Este problema es su principal inconveniente ya que suele generar un conflicto con el usuario/cliente por no comprenderlo. Otros inconvenientes son que no es adecuado para proyectos grandes porque requiere un alto número de recursos, que requiere un compromiso entre clientes y desarrolladores, muchas veces difícil de mantener en el tiempo, y que no es adecuado para todo tipo de proyectos como, por ejemplo, aquellos que no permiten la descomposición por módulos y suponen elevados riesgos técnicos.

Los objetivos de este prototipo son:

- Reducir el riesgo al tener una mayor seguridad de cubrir las necesidades del usuario/cliente.
- Reducir el coste de desarrollo al disminuir las correcciones en etapas avanzadas del desarrollo.
- Aumentar las posibilidades de éxito.

3.4.4.1.2 MODELO DE PROTOTIPADO EVOLUTIVO

En el modelo de prototipado evolutivo se construye una implementación parcial del sistema que satisface los requisitos conocidos, la cual es utilizada por el usuario para llegar a comprender mejor la totalidad de requisitos que desea del sistema, y a partir de ese punto evolucionarlo.

Este modelo es muy apropiado cuando el usuario tiene dificultad para describir lo que necesita, pero se observa que cuando se le muestra un prototipo si reconoce sus intereses.

Un subtipo de este modelo es el modelo RAD (Desarrollo rápido de aplicaciones) basado en los elementos: personas, herramientas, metodología y gestión, que hace uso de herramientas avanzadas y de sesiones de trabajo conjuntas del equipo de desarrollo con los usuarios, lo que podría hacer pensar en los posibles inicios de las metodologías ágiles.

La diferencia del modelo de prototipado rápido y el evolutivo, es que en el modelo de prototipado rápido los requisitos, aunque escasos son conocidos y reales, mientras que en el modelo de prototipado evolutivo se sabe desde el principio que los requisitos cambiaran continuamente.

En el prototipado rápido se debe comenzar realizando el prototipo de lo que menos se entiende, mientras que el evolutivo se debe comenzar realizando el prototipo de lo que mejor se comprende.

Como inconveniente del prototipado evolutivo, se puede decir que se le considera una versión “moderna” del modelo codificar y corregir. Además, de ser un error considerar no realista que el sistema operacional del usuario final será lo suficientemente flexible como para poder incorporar la evolución que experimente el sistema de forma no planificada.

3.4.4.1.3 MODELO DE DESARROLLO INCREMENTAL

En el Modelo de desarrollo incremental se procede incorporando de forma gradual e incremental los requisitos al desarrollo del sistema realizado hasta llegar al sistema final.

Este modelo tiene las siguientes ventajas:

- Disponer pronto de un sistema preparado para ser utilizado con la funcionalidad más necesaria para el cliente.
- Facilitar la adaptación gradual del cliente al sistema.
- La implicación del usuario/cliente en la planificación del proyecto.
- Evitar demanda de funcionalidad excesiva por parte de usuarios/clientes, al comenzar comunicando las esenciales y, en muchas ocasiones postponer a muy largo plazo las no tan necesarias.
- La detección de los errores de forma temprana.

Y como principales inconvenientes:

- La dificultad para definir el núcleo que formará parte del primer incremento, así como la definición de los incrementos,
- Las soluciones de incrementos anteriores pueden ser inválidas para incrementos posteriores.

La diferencia entre el modelo evolutivo y el incremental es que:

- En el modelo incremental se conocen todos los requisitos y éstos se van incorporando al sistema en versiones sucesivas, mientras que en el evolutivo sólo se conocen unos pocos requisitos y los demás se van descubriendo en sucesivas evoluciones del prototipo.
- Cada vez que se desarrolla una nueva versión, en el evolutivo es una versión de todo el sistema, mientras que en el incremental es una versión anterior sin cambios al que se le añade alguna funcionalidad.

3.5 MODELO EN ESPIRAL

Es un modelo de prototipado evolutivo en el que converge la construcción de forma iterativa de prototipos con los aspectos controlados y sistemáticos del modelo en cascada.

En el modelo espiral, el software se desarrolla en una serie de versiones incrementales. Durante las primeras iteraciones, la versión incremental podría ser un modelo en papel o un prototipo. Durante las últimas iteraciones, se producen versiones cada vez más completas del sistema diseñado.

El proyecto se divide en ciclos cada uno de ellos con las siguientes actividades:

- Planificación e identificación de las alternativas para realizar cada parte del producto.
- Análisis de riesgos de cada alternativa.
- Desarrollo y verificación del producto resultado del ciclo. Al estar esta implementación dirigida por el riesgo, para el desarrollo en sí, se podría seguir otro tipo de modelo.
- Evaluación del cliente de todos los productos desarrollados en el ciclo, incluido los planes del siguiente ciclo y la disponibilidad de recursos actualizada.

En el modelo en espiral existe un reconocimiento explícito de las diferentes alternativas para alcanzar los objetivos del proyecto, a la vez que se centra en la identificación de sus riesgos, ya sea que estos afecten al desarrollo o al rendimiento del sistema.

Algunos de los riesgos que considera el modelo son:

- La escasez de personal.
- Presupuesto y planificación temporal.
- Desarrollo de funcionalidad e interfaces de usuarios no adecuados.
- Cambios excesivos sobre los requisitos.
- Desarrollo o adquisición de productos software de baja calidad.

Una vez identificados y evaluado los riesgos, se pueden dar las siguientes circunstancias:

- Si predominan los riesgos de rendimiento e interfaces frente a los de desarrollo, y el prototipo ha cubierto operacionalmente las necesidades con bajo riesgo, se procederá a evolucionar ese mismo prototipo.
- Si predominan los riesgos de desarrollo, siendo mayor el número de riesgos de rendimiento e interfaces, el prototipo cubre todos los riesgos de rendimiento e interfaces, pero no los de desarrollo, se propone avanzar utilizando, por ejemplo, el modelo de ciclo de vida en cascada ajustado con conceptos del modelo incremental.

La ventaja más importante de este modelo es la alta posibilidad de configuración que tiene al permitir usar otros modelos en cada actividad. En cuanto a los principales inconvenientes es la incertidumbre en el número de iteraciones necesarias, su dificultad de aplicación en el desarrollo de software contratado externamente a la organización, además de la necesidad de un alto conocimientos en análisis y gestión del riesgo.

3.6 MODELO BASADO EN TRANSFORMACIONES

Los modelos basados en transformación, también conocidos como modelos de ciclo de vida con producción automática de diseño y código, se basan en la posibilidad de convertir automáticamente una especificación formal de un producto software en un programa que satisfaga las especificaciones utilizando herramientas que ayuden a esta labor.

En estos modelos tras la transformación es necesario realizar bucles iterativos para mejorar el rendimiento del código resultante, probar el producto resultante y reajustar las especificaciones para iniciar de nuevo la generación de código, y continuar iterativamente.

Estos modelos minimizan la necesidad de tener que modificar el código poco estructurado, pero como inconveniente solo son útiles para productos pequeños y aplicados a áreas limitadas, e incluso en estos casos, puede ocurrir que el entorno de producción del usuario/cliente no pueda asumir las evoluciones del producto no planificadas previamente.

3.7 MODELO BASADO EN COMPONENTES

El Modelo basado en componentes (*Computer Based System Engineering* CBSE), permite reutilizar piezas de código ya existente que ensamblados mediante interfaces estándar, lo que permite reducir el ciclo de desarrollo, un mayor retorno sobre la inversión y mejor calidad y fiabilidad del producto final.

Los principales inconvenientes de este modelo es la dificultad para seleccionar componentes reutilizables adecuados y realizar una correcta configuración, lo que ha mejorado notablemente gracias a la estandarización de la documentación y los actuales catálogos y repositorios.

3.8 MODELO UNIFICADO DE DESARROLLO DE SOFTWARE

Al proceso unificado de desarrollo de software (PUDS o RUP) es más acertado considerarlo una metodología.

Se basa en un ciclo de vida iterativo e incremental, centrado en una arquitectura que guía el desarrollo del sistema, cuyas actividades están dirigidas por casos de uso y la orientación a objetos, y que da una importancia relevante al control de la calidad y la gestión de riesgos.

Este modelo se compone de fases, considerando cada una de ellas como el intervalo de tiempo entre dos hitos importantes del proceso durante la cual se cumple un conjunto bien definido de objetivos, se completan entregables y se toman las decisiones sobre si pasar o no a la siguiente fase. Se consideran fases de este modelo las siguientes:

- La iniciación que incluye la elaboración de un plan de proyecto y el análisis de riesgos.
- La elaboración del análisis del problema y primer diseño de la arquitectura.
- La construcción o desarrollo de los componentes y casos de uso de un tamaño adecuado para producir prototipos funcionales.
- La transición que implica la implantación del producto, la formación de los usuarios y las pruebas de aceptación del sistema.

En cada fase puede haber varias iteraciones, cada una de ellas con un resultado de producto cada vez más avanzado, y siempre centrado en el riesgo.

Los flujos de trabajo del proceso son los siguientes:

- Modelado del negocio.
- Descripción de los requisitos.
- Análisis y diseño.
- Implementación (desarrollo y pruebas unitarias).
- Pruebas de integración.
- Generación de versión estable y despliegue o distribución del producto.
- Configuración y gestión de cambios.
- La propia gestión del proyecto a nivel de fases y de iteraciones.

3.9 MODELO DE MÉTODOS FORMALES

El modelo de métodos formales se basa en la utilización de una notación rigurosa y matemática que permite llevar a cabo el desarrollo de un producto libre de fallos, lo que supone desarrollos muy caros y que con llevan mucho tiempo, y que habitualmente no es posible utilizarlo con los clientes y usuarios finales por su dificultad y la necesidad de conocimiento técnico muy alto para la mayoría de los clientes y usuarios.

El uso de este modelo no es frecuente, siendo de mayor interés para los desarrolladores de software con requerimientos de seguridad y robustez muy altos.

3.10 MODELO DE CICLO DE VIDA ÁGIL

En el modelo de ciclo de vida ágil, a diferencia de las metodologías más tradicionales, es un marco que no tiene por qué tener etapas que sigan un orden predefinido, sino que cada equipo trabajará de forma interconectada permitiendo unas relaciones más flexibles y participativas. Además, será necesaria la integración del cliente en el proceso de creación, dando la validación de cada prototipo.

Se considera un ciclo iterativo e incremental porque la característica principal de este ciclo de vida es el carácter repetitivo.

Las etapas que, como requisito tendrán corta duración y se solaparán, se repiten para añadir funcionalidad al producto y realizar entregas parciales del producto que se validen con el usuario/cliente que participa de una forma mucho más cercana en el proyecto.

Uno de los inconvenientes que surge en este modelo, es la menor documentación, resultado natural de una mayor integración del equipo y de la participación en el equipo del cliente/usuario.

Habitualmente este ciclo de vida se compone de las siguientes fases:

- Definición del proyecto.
- Especular en relación a la planificación de iteraciones.
- Desarrollo y prueba de las funcionalidades.
- Revisión de los resultados de cada iteración.
- Cierre del proyecto.

Los ciclos de vida ágiles, también son conocidos como adaptativos o métodos orientados al cambio y responden a niveles altos de cambio y a la participación continua de los interesados.

Existen dos modelos básicos para este tipo de ciclos de vida, aquellos centrados en el flujo (Kanban) en los que se establecen limitaciones muy claras sobre la concurrencia de actividades (Work in Progress), y otros centrados en ciclos iterativos e incrementales (Scrum) en el que las iteraciones son muy rápidas (entre 1 y 4 semanas) donde se realiza el trabajo (Sprint).

Habitualmente en los modelos ágiles el alcance global del proyecto será descompuesto en un conjunto de requisitos o trabajos a realizar (Product Backlog). Al inicio de una iteración el equipo define las funcionalidades que serán abordadas en ese ciclo. Al final de cada iteración el producto estará listo para su revisión por el cliente. Este tipo de ciclo de vida requiere de equipos muy involucrados que incluyan al cliente para llevar a cabo revisiones y retroalimentación útil por estar actualizada continuamente.

Generalmente se opta por los métodos ágiles en entornos que cambian rápidamente, cuando el alcance es confuso o cuando la aportación de valor es muy cambiante y con equipos altamente involucrados.

En el tema 88 de este temario de título “Análisis funcional de sistemas, casos de uso e historias de usuario. Metodologías de desarrollo de sistemas. Metodologías ágiles: Scrum y Kanban se realiza una detallada descripción de los distintos tipos de metodologías ágiles, pero se enumeran a continuación como recordatorio:

- Scrum.
- Kanban.
- Extreme Programming (XP).
- Marco de Proyecto Adaptativo (APF).
- Crystal Clear.

3.10.1 MODELO DE PROGRAMACIÓN EXTREMA

En el modelo de programación extrema (eXtreme programming) se presentan las siguientes características:

- Todos los requerimientos se expresan como escenarios (historias de usuario) que se implementan como un conjunto de tareas por parte de un equipo de dos programadores.
- Las pruebas se definen antes de escribir el código.
- En la programación se realizan entregas de pequeño tamaño, frecuentes y a partir de historias de usuario.

- El usuario/cliente tiene participación plena en el equipo de desarrollo, este define, junto con el resto del equipo, las tarjetas de historias que recogen las necesidades y las pruebas de aceptación del sistema.
- Tras la aprobación del desarrollo y la superación de las pruebas de integración continuas, se realiza la implantación.
- El problema con la implementación de cambios imprevistos es que tienden a degradar la estructura del software, por lo que los cambios se hacen cada vez más difíciles de implementar. El mantenimiento de la simplicidad se lleva a cabo a través de la refactorización constante para mejorar la calidad del código y la utilización de diseños sencillos que no prevén cambios futuros en el sistema.

3.11 Conclusión

Considerando los procesos de software como el conjunto de actividades que conducen a la creación de un producto software, esta definición podría ser tan variada como la combinación entre los posibles productos software y los tipos de organizaciones existentes, es decir no existe un proceso óptimo, sino que este será tanto más válido cuanto más flexible sea y se adapte a las necesidades de la organización, el desarrollo a realizar y el equipo informático, siendo más útil un proceso muy estructurado cuando se trabaje para un sistema crítico y flexible y ágil cuando se trabaje en un sistema de requerimientos muy cambiantes.

En general todos estos procesos tienen las actividades básicas de:

- Especificación y toma de requerimientos.
- Desarrollo
- Validación y pruebas.
- Evolución y mantenimiento.

Una vez definido el proceso de software, se define el **modelo de ciclo de vida** de los sistemas como una representación simplificada y dada unas circunstancias concretas de un proceso de software que describe un conjunto de fases y las relaciones entre ellas. Este concepto se nombra también como ciclo de vida, modelo de desarrollo o modelo de proceso de software.

Un modelo de ciclo de vida debe cubrir los siguientes objetivos básicos:

- Definir las actividades a realizar y el orden de las fases del proceso software.
- Establecer los criterios de transición para pasar de una fase a la siguiente.
- Proporcionar puntos de control temporales y presupuestarios para la gestión del proyecto.
- Asegurar la consistencia con el resto de los sistemas de información de la organización.

Los tipos de modelos de ciclo de vida son los siguientes:

- Modelo codificar y corregir
- Modelo por etapas
- Modelo en cascada
- Modelo en v
- Modelos basados en prototipos
  - Modelo de prototipado rápido
  - Modelo de prototipado evolutivo
  - Modelo de desarrollo incremental
- Modelo en espiral
- Modelo basado en transformaciones
- Modelo basado en componentes
- Modelo unificado de desarrollo de software
- Modelo de métodos formales
- Modelo utilizado en metodologías ágiles

Las circunstancias particulares de cada proyecto software, hace difícil que se ajusten completamente a un modelo, por lo que lo habitual es trabajar con ellos de forma combinada.

4 RESUMEN ESQUEMÁTICO

- Ciclo de vida del sistema (modelo de desarrollo o modelo de proceso de software) es el marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso. Es decir, es el conjunto de etapas por las que atraviesa el sistema desde su concepción, hasta su retirada, pasando por su desarrollo y explotación.
- La metodología de desarrollo de software describe el trabajo a desarrollar en cada paso, los productos a obtener y las técnicas que se recomienda usar para generarlos.
- La norma ISO/IEC 12207:2017 establece un marco de trabajo común en relación con el conjunto completo de etapas del ciclo de vida del software, con una terminología bien definida y un amplio reconocimiento e implantación en el ámbito de los sistemas de información y las comunicaciones. Además, incorpora los procesos que permiten la mejora continua. Estos procesos tienen doble alcance, tanto el desarrollo de software, como la adquisición de sistemas de información.
- El estándar 12207 consta de procesos para la adquisición y suministro de proyectos y servicios del software, estableciendo pautas de control de calidad y mantenimiento integral, además de ofrecer un lenguaje común que facilita las interrelaciones entre los diferentes agentes (compradores, proveedores, desarrolladores, personal de mantenimiento, operadores, gestores y técnicos) que pueden beneficiarse del uso de este estándar. Clasifica los procesos en tres tipos primarios, de soporte y de organizacionales.
- La norma ISO/IEC 15288:2015 crea un marco que detalla el modo de ejecución de los procesos de descripción del ciclo de vida para la generación del desarrollo de software, además aporta un conjunto de indicadores para ser aplicados en cada etapa del ciclo de desarrollo dentro de un proceso integral de mejora continua y eficiencia. Tiene un enfoque dirigido al desarrollo de todo tipo de software. Esta norma comprende 25 procesos que tienen 123 resultados derivados de 403 actividades, lo que hace la aplicación de esta norma una labor muy costosa de aplicar casi por cualquier organización.
- Conforme al principio de minimización de datos y el principio de protección de datos desde el diseño y por defecto, siempre que sea posible se utilizarán datos sintéticos para evitar el tratamiento de datos personales en pruebas de desarrollo, y cuando esto no sea posible deberá documentarse mediante un análisis de necesidad y proporcionalidad, y en todo caso aplicar las medidas técnicas y organizativas que sean necesarias conforme al artículo 32 del RGPD y de acuerdo con el riesgo del tratamiento.
- Hay que tener en cuenta que en ocasiones el riesgo en el entorno de desarrollo y preproducción es incluso mayor al riesgo del tratamiento en el entorno de producción.
- El ciclo de desarrollo seguro de aplicaciones S-SDLC (Secure Software Development Life Cycle) es un conjunto de procesos y procedimientos diseñados para facilitar la identificación de los puntos débiles de seguridad en algunas o en todas las etapas del SDLC, mediante la aplicación de controles de seguridad que sirvan para tomar las acciones necesarias con el fin de asegurar el software lo máximo posible seguridad durante todo el ciclo de vida de la aplicación, desde su diseño hasta su puesta en producción y durante todo el tiempo de servicio.
- Los modelos de madurez de desarrollo del software seguro proporcionan un marco organizado de requisitos de seguridad cuyo nivel de cumplimiento permite evaluar la posición de madurez de la implementación de la seguridad en: una aplicación, un proyecto o una organización. Los más utilizados son: OWASP SAMM, ISO 33000.
- Si la incorporación de la seguridad sólo se realiza en las fases avanzadas del ciclo de vida del desarrollo de las aplicaciones, seguirá existiendo un elevado riesgo, un mayor coste de recuperación de la operativa de los sistemas y una disminución de la productividad, debido a que la incorporación de controles y medidas de seguridad es más costosa, y requiere un mayor esfuerzo del equipo de desarrollo, lo que degrada su productividad e implica la asunción de riesgos mayores por parte de la dirección del proyecto.
- En general los procesos de software tienen las actividades básicas de especificación y toma de requerimientos, desarrollo, validación y pruebas, y evolución y mantenimiento.
- Una vez definido el proceso de software, se define el modelo de ciclo de vida de los sistemas como una representación simplificada y dada unas circunstancias concretas de

un proceso de software que describe un conjunto de fases y las relaciones entre ellas. Este concepto se nombra también como ciclo de vida, modelo de desarrollo o modelo de proceso de software.

- Un modelo de ciclo de vida debe cubrir los siguientes objetivos básicos: definir las actividades a realizar y el orden de las fases del proceso software, establecer los criterios de transición para pasar de una fase a la siguiente, proporcionar puntos de control temporales y presupuestarios para la gestión del proyecto, y asegurar la consistencia con el resto de los sistemas de información de la organización.
- A continuación, se definen brevemente los distintos tipos de modelos de ciclo de vida.
  - Modelo codificar y corregir.
  - Modelo por etapas sucesivas, conocido como “Stage Wise.
  - El modelo en cascada se compone de una serie de fases que se suceden secuencialmente, generándose en cada una de ellas unos resultados que serán necesarios para iniciar la fase siguiente. Es decir, la evolución del producto software se produce a través de una secuencia ordenada de transiciones de una fase a la siguiente, según un orden lineal.
  - El modelo en V es similar al ciclo de vida en cascada, pero se caracteriza por tener dos tiempos diferenciados: el desarrollo y las pruebas.
  - El modelo de prototipado rápido tiene como objetivo generar un tipo de prototipo en una fase muy inicial del desarrollo, invirtiendo poco esfuerzo y presupuesto. Este tipo de prototipo es muy importante que se deseche para evitar problemas futuros surgidos de su aprovechamiento.
  - En el modelo de prototipado evolutivo se construye una implementación parcial del sistema que satisface los requisitos conocidos, la cual es utilizada por el usuario para llegar a comprender mejor la totalidad de requisitos que desea del sistema, y a partir de ese punto evolucionarlo. Este modelo es muy apropiado cuando el usuario tiene dificultad para describir lo que necesita, pero se observa que cuando se le muestra un prototipo si reconoce sus intereses.
  - Modelo de desarrollo incremental, se procede incorporando de forma gradual e incremental los requisitos al desarrollo del sistema realizado hasta llegar al sistema final.
  - Modelo en espiral, es un modelo de prototipado evolutivo en el que converge la construcción de forma iterativa de prototipos con los aspectos controlados y sistemáticos del modelo en cascada, en este modelo, el software se desarrolla en una serie de versiones incrementales. Durante las primeras iteraciones, la versión incremental podría ser un modelo en papel o un prototipo. Durante las últimas iteraciones, se producen versiones cada vez más completas del sistema diseñado.
  - Modelos basados en transformación, también conocidos como modelos de ciclo de vida con producción automática de diseño y código, se basan en la posibilidad de convertir automáticamente una especificación formal de un producto software en un programa. A partir de esta transformación se realizan bucles iterativos para mejorar el rendimiento del código, probarlo y reajustar las especificaciones para iniciar de nuevo el proceso.
  - Modelo basado en componentes, permite reutilizar piezas de código ya existente ensamblados mediante interfaces estándar, lo que permite reducir el ciclo de desarrollo, un mayor retorno sobre la inversión y mejor calidad y fiabilidad del producto final.
  - Modelo unificado de desarrollo de software, se basa en un ciclo de vida iterativo e incremental, centrado en una arquitectura, con actividades dirigidas por casos de uso y orientación a objetos, y que destaca por el control de la calidad y la gestión de riesgos.
  - Modelo de métodos formales basado en una notación rigurosa y matemática que permite llevar a cabo el desarrollo de un producto libre de fallos.
  - Modelo de ciclo de vida ágil, en el que cada equipo trabajará de forma interconectada permitiendo unas relaciones más flexibles y participativas. Se considera un ciclo iterativo e incremental porque la característica principal de este ciclo de vida es el carácter repetitivo.
  - Un ejemplo es el modelo de programación extrema (eXtreme Programming).

5 GLOSARIO

- SW: Software.
- MCV: Modelo de ciclo de vida.
- ENI: Esquema Nacional de Interoperabilidad.
- NTI: Norma Técnica de Interoperabilidad.
- ENS: Esquema Nacional de Seguridad
- ISO: Organización Internacional para la Estandarización (International Organization for Standardization).
- IEC: Comisión Internacional Electrotécnica (International Electrotechnical Commission).
- MBSE: Modelo basado en componentes (Computer Based System Engineering CBSE),
- CPD: Centro de Proceso de Datos.
- S-SDLC: Ciclo de desarrollo seguro de aplicaciones (Secure Software Development Life Cycle).
- PUDS o RUP: modelo unificado de desarrollo de software (Proceso Unificado de Desarrollo de Software, Rational Unified Process).

6 TEST

6.1 PREGUNTAS DE TEST

1) Cuáles son las principales ventajas del uso de las metodologías ISO 12207 y 15504 son las siguientes:
   a) Permiten la evaluación, seguimiento y mejora de los procesos, por tanto, del producto final y la satisfacción de clientes/usuarios finales.
   b) Mejora la eficacia y eficiencia del proceso de desarrollo, su calidad y, por tanto, reducen las incidencias, errores y fallos en la producción
   c) Definen los procesos del ciclo de vida del desarrollo de software, su mantenimiento, operación y explotación con los sistemas de software, manteniendo registros de todos los procesos y procedimientos.
   d) Todas las anteriores respuestas son ciertas.

2) La ingeniería del software se compone de:
   a) Métodos, herramientas y procedimientos.
   b) Listados, programas, y bases de datos.
   c) Líneas de comunicaciones, software y hardware.
   d) Ninguna de las anteriores respuestas es cierta.

3) Respecto a un modelo de ciclo de vida de los sistemas:
   a) Se considera como una representación simplificada y dada unas circunstancias concretas de un proceso de software que describe un conjunto de fases y las relaciones entre ellas.
   b) Es un concepto que también se denomina como ciclo de vida, modelo de desarrollo o modelo de proceso de software.
   c) Es una lista de verificación que es necesario superar antes de su implantación en producción.
   d) La respuesta a y b son ciertas.

4) Algunos de los objetivos de un modelo de ciclo de vida son los siguientes:
   a) Definir las actividades a realizar y el orden de las fases del proceso software.
   b) Establecer los criterios de transición para pasar de una fase a la siguiente.
   c) Proporcionar puntos de control temporales y presupuestarios para la gestión del proyecto.
   d) Todos los anteriores son ciertos.

5) La decisión para utilizar un modelo u otro puede ser en base a distintos criterios, por ejemplo:
   a) Si el equipo de desarrollo posee la experiencia suficiente y los requisitos están perfectamente fijados, es recomendable un ciclo de vida basado en prototipos
   b) Si el equipo de desarrollo posee la experiencia suficiente y los requisitos están perfectamente fijados, es recomendable un ciclo de vida en cascada.
   c) El ciclo de vida en codificar y corregir será el elegido si durante el desarrollo del proyecto se tendrán en cuenta riesgos o imprevistos.
   d) En el caso de tener que ir afinando con el cliente cada etapa, con el objetivo de mostrarle su utilidad, el ciclo de vida basado en cascada será el elegido.

6) Cuál de los siguientes motivos es cierto en el uso de prototipos:

a) Finalizar antes una versión mínima que puede ser usada por el usuario en su entorno real de producción.
b) Obtener un correcto producto mínimo viable.
c) Ayudar a comprender los requisitos y por tanto a asegurar el entendimiento entre el equipo de desarrollo y los usuarios funcionales.
d) Ninguna de las anteriores respuestas es cierta.

7) La diferencia del modelo de prototipado rápido y el evolutivo, es:
   a) Que en el modelo de prototipado rápido los requisitos, aunque escasos son conocidos y reales, mientras que en el modelo de prototipado evolutivo se sabe desde el principio que los requisitos cambiarán continuamente.
   b) Que en el modelo de prototipado evolutivo los requisitos, aunque escasos son conocidos y reales, mientras que en el modelo de prototipado rápido se sabe desde el principio que los requisitos cambiarán continuamente.
   c) Que en el modelo de prototipado rápido los requisitos son abundantes, mientras que en el modelo de prototipado evolutivo se conocen muy pocos requisitos.
   d) Ninguna de las respuestas anteriores es cierta.
8) La diferencia entre el modelo evolutivo y el incremental es que:
   a) En el modelo incremental se conocen todos los requisitos y éstos se van incorporando al sistema en versiones sucesivas, mientras que en el evolutivo sólo se conocen unos pocos requisitos y los demás se van descubriendo en sucesivas evoluciones del prototipo.
   b) Cada vez que se desarrolla una nueva versión, en el evolutivo es una versión de todo el sistema, mientras que en el incremental es una versión anterior sin cambios al que se le añade alguna funcionalidad.
   c) En el modelo evolutivo se conocen todos los requisitos y éstos se van incorporando al sistema en versiones sucesivas, mientras que en el incremental sólo se conocen unos pocos requisitos y los demás se van descubriendo en sucesivas evoluciones del prototipo.
   d) Las respuestas a y b son ciertas.

9) En relación a los modelos de ciclo de vida ágil:
   a) Generalmente se opta por los métodos ágiles en proyecto en los que está completamente claro su alcance.
   b) Generalmente se opta por los métodos ágiles en entornos que cambian rápidamente, cuando el alcance es confuso o cuando la aportación de valor es muy cambiante y con equipos altamente involucrados.
   c) Generalmente se opta por los métodos ágiles cuando se trabaja con un equipo con una alta rotación, es decir con continuas bajas.
   d) Ninguna de las anteriores repuestas es cierta.

10) Respecto al modelo de prototipado evolutivo:
   a) Es aquel en el que converge la construcción de forma iterativa de prototipos con los aspectos controlados y sistemáticos del modelo en cascada.
   b) El software se desarrolla en una serie de versiones incrementales.
   c) Durante las últimas iteraciones, se producen versiones cada vez más completas del sistema diseñado.
   d) Todas las respuestas anteriores son ciertas.

6.2 SOLUCIONES A LAS PREGUNTAS DE TEST

PREGUNTA	SOLUCIÓN
1	d
2	a
3	d
4	d
5	b
6	c
7	a
8	d
9	b
10	d

7 BIBLIOGRAFÍA

7.1 BIBLIOGRAFÍA BÁSICA

- Real Decreto 4/2010, de 8 de enero, por el que se regula el Esquema Nacional de Interoperabilidad en el ámbito de la Administración Electrónica.
- Ciclo de vida. Universidad de Granada
- El ciclo de vida. Universidad de Valladolid.
- Modelos de proceso de SW, UNEMI Online
- https://www.aepd.es/es/prensa-y-comunicacion/blog/brechas-datos-personales-en-entornos-de-desarrollo-y-preproduccion

7.2 BIBLIOGRAFÍA PARA AMPLIAR EL TEMA

- Real Decreto 311/2022, de 3 de mayo, por el que se regula el Esquema Nacional de Seguridad.
- Ley 9/2017, de 8 de noviembre, de Contratos del Sector Público, por la que se transponen al ordenamiento jurídico español las Directivas del Parlamento Europeo y del Consejo 2014/23/UE y 2014/24/UE, de 26 de febrero de 2014.
- Diccionario de conceptos y términos de la Administración Electrónica (9ª ed.) (2023)
- Repertorio de Referencias Directas y Cruzadas en la AGE.
- Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia.
- REGLAMENTO (UE) 2016/679 DEL PARLAMENTO EUROPEO Y DEL CONSEJO de 27 de abril de 2016 relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos y por el que se deroga la Directiva 95/46/CE (Reglamento general de protección de datos)
- Informe 58/2018 de la Junta Consultiva de Contratación del Estado.
- Directiva 2009/24/CE
- Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia
- Ley 40/2015, de 1 de octubre, de Régimen Jurídico del Sector Público