INSTITUTO de ENSEÑANZAS a DISTANCIA de ANDALUCÍA

2º de Bachillerato

Tecnologías de la Información y Comunicación

Contenidos

Ciclo de vida: Metodologías de desarrollo de software

Llegados a este punto del camino, dispones de herramientas suficientes para la construcción de programas informáticos. También conoces el paso de traducción de un programa escrito en pseudocódigo a un lenguaje de programación concreto.

En este tema se intenta unificar todo el proceso de creación de programas informáticos, detallando los pasos o etapas a seguir, hasta conseguir un software más o menos comercial, o dicho de otro modo, con un resultado final justificado según unas reglas o estándares de calidad.

En la práctica hablamos de software para referirnos a un programa informático que podemos usar habitualmente y, que en cierto modo, nos garantiza que su uso nos ayudará en nuestro trabajo, facilitando cálculos o tareas que de forma manual serían bastante tediosas y complejas.

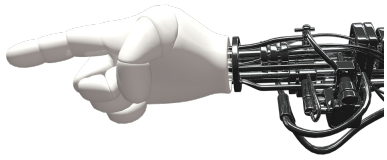
Para conseguir dicho software, **es necesario considerar el llamado ciclo de vida del software**, concepto que se usa para especificar dichos pasos o etapas, las cuales se deben tener en cuenta para conseguir un software o programa informático fiable y eficiente.

El proceso de creación de software **necesita de una metodología de desarrollo** del mismo que, de una forma organizada, vaya completando u obteniendo resultados en cada etapa de su ciclo de vida hasta tener el producto final.

1. ¿Qué voy a conseguir?

Al finalizar el tema,

- Conocerás el significado de los **conceptos** básicos usados **en ingeniería del software**.
- Comprenderás el proceso de **evolución hasta** lo que se conoce como **ingeniería del software**.
- Entenderás la **función de cada fase** o etapa del desarrollo de software diferenciándolas.
- Conocerás y distinguirás los **diferentes tipos de ciclos de vida** valorando sus ventajas e inconvenientes.
- Distinguirás la diferencia entre desarrollo de software **estructurado y** desarrollo de software **orientado a objetos**.
- Comprenderás según un **caso práctico** el proceso de desarrollo de software según el modelo de ciclo de vida clásico o **en cascada**.



2. Sistemas de Información

Todos sabemos que **el objetivo último** de la creación **de un programa informático es** ayudar, facilitar y, en definitiva, **obtener beneficios** de las ventajas, **en** cuanto a rapidez y eficiencia, que se reflejan en las metas generales que espera una organización, comunidad de usuarios o como se suele denominar comúnmente **una empresa**.

En este sentido, hablamos de **empresa como un conjunto** formado por personas, materiales, herramientas y productos que reman juntos en una misma dirección: mejorar para obtener mayores beneficios según interese (académicos, económicos, etc).

El concepto de empresa **engloba** lo que se denominan **los sistemas como partes** que, **según una función concreta**, trabajan en un campo específico **pero con el objetivo común** de alcanzar esa meta global a nivel de empresa.

De esta forma, **el sistema de información es el subsistema dentro de la empresa que permite el uso y las transferencias de información entre unos subsistemas y otros de la empresa**.

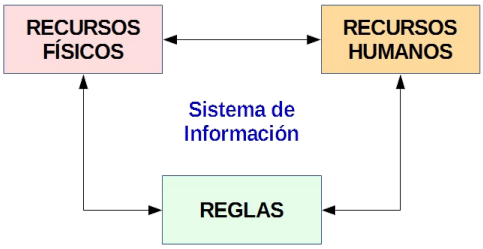
Imagen en pixabay de JuralMin bajo licencia Creative Commons

Subsistemas de una empresa

Subsistema A		
Subsistema B		
Subsistema C: **Sistema de información**	→	Meta u **objetivo** global: mejorar para obtener mayores beneficios
...		
Subsistema N		

Los elementos o **componentes** principales **de un sistema de información** son tres:

- **Recursos físicos**: documentos, archivos, equipos informáticos, equipos de telecomunicaciones, etc, que lo forman.
- **Recursos humanos**: personas, grupos, equipos de trabajo, etc, que trabajan en él.
- **Reglas**: protocolos, normas, métodos, etc, que establecen cómo debe hacerse la transmisión de información y cómo debe utilizarse esta.



(Imagen de creación propia)

3. Ingeniería del software. Etapas

Imagen en pixabay de IB306660 bajo licencia Creative Commons

En la práctica, los sistemas de información, también llamados **sistemas informáticos (cuando parte o toda la gestión del sistema de información se realiza con ordenadores)**, son grandes y complejos, es decir, no son simples programas abordables por una única persona. Se hace necesario disponer de:

- **Tiempo y recursos materiales** suficientes, lo que obliga a planificar previamente el trabajo a desarrollar.
- **Un equipo de informáticos** especializados y jerarquizados (jefes de proyecto, analistas, programadores, etc).
- **Una serie de métodos o técnicas** de resolución de problemas para desarrollar los distintos pasos a seguir en todo el proceso de desarrollo del sistema, esto es, el desarrollo de un software apropiado.

Para este cometido nace la **ingeniería del software**, la idea es **convertir el proceso artesanal de construcción de programas informáticos en una ingeniería**, para así producir software industrial en serie y de calidad.

Como cualquier rama o disciplina dentro de la Informática, la ingeniería del software ha ido evolucionando en el tiempo hasta la fecha actual. Las etapas o generaciones por las que ha ido pasando se asemejan al desarrollo de un programador desde que se inicia en el mundo de la programación hasta que se hace más o menos experto en el análisis, diseño y construcción de programas informáticos de un nivel aceptable.

.

Etapas de la ingeniería del software

En ingeniería del software se consideran **cuatro etapas o generaciones** claramente diferenciadas:

En la **primera generación** de desarrollo de software, la construcción de programas informáticos se llevaba a cabo de una forma directa, esto es, planificando a groso modo las partes de un programa sin entrar a detallar el proceso de análisis y diseño previos. Se **carecía de una metodología específica** y la experiencia del programador era clave en la construcción de software. Esta etapa se prolongó hasta mediados de la década de los 70.

La **segunda generación** nace con la idea de centrarse más en el análisis del problema para afrontar con mayor garantías la programación. **Aparecen las primeras metodologías de programación estructurada**. El ya clásico "*divide y vencerás*" es el referente para diseñar programas más legibles.

El **desarrollo orientado a objetos** aparece en la **tercera generación** como alternativa a las técnicas clásicas de programación estructurada. Ello supuso una forma distinta de representar la realidad para representar y tratar los problemas de programación a través del concepto de clase. Dicha manera de representar la realidad supuso un mayor acercamiento a la forma más natural de pensar del ser humano. El **lenguaje UML** (Unified Modeling Languaje) es el elegido para modelar la realidad y resolver problemas según esta filosofía.

Por último, en la **cuarta generación** surgen las llamadas **herramientas CASE** (Computer Aided Software Engineering) **y los lenguajes de cuarta generación** (centrados en el concepto de **evento**) como verdaderas herramientas para desarrollar software de una forma más rápida y eficaz, construyendo cada parte del software final según una herramienta gráfica cómoda y amigable, y automatizando el proceso de generación de código.

Para saber más

El lenguaje UML

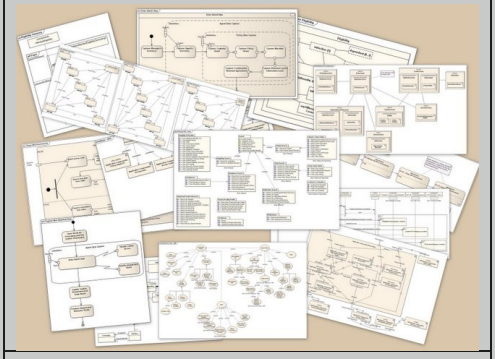


Imagen en Wikimedia bajo licencia Creative Commons

El **lenguaje unificado de modelado** (**UML**, por sus siglas en inglés, *Unified Modeling Language*) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el *Object Management Group* (OMG).

Se puede aplicar en el desarrollo de software gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional, *Rational Unified Process* o RUP), pero no especifica en sí mismo qué metodología o proceso usar.

UML no puede compararse con la programación estructurada, pues UML significa Lenguaje Unificado de Modelado, no es programación, solo se diagrama la realidad de una utilización en un requerimiento. Mientras que, programación estructurada, es una forma de programar como lo es la orientación a objetos, la programación orientada a objetos viene siendo un complemento perfecto de UML, pero no por eso se toma UML solo para lenguajes orientados a objetos.

Comprueba lo aprendido

En las secciones anteriores han aparecido muchos conceptos. Intenta recordar.

En la práctica hablamos de [ ] para referirnos a un programa informático que podemos usar habitualmente y, que en cierto modo, nos garantiza que su uso nos ayudará en nuestro trabajo.

Para conseguir dicho programa, es necesario considerar el llamado [ ] del software, concepto que se usa para especificar dichos pasos o [ ] , las cuales se deben tener en cuenta para conseguir un programa informático fiable y eficiente.

Hablamos de empresa como un conjunto formado por [ ], materiales, [ ] y productos que reman juntos en una misma dirección: mejorar para obtener mayores [ ] según interese (académicos, económicos, etc).

El sistema de [ ] es el subsistema dentro de la empresa que permite el uso y las [ ] de información entre unos subsistemas y otros de la empresa. Sus elementos o componentes principales son tres: recursos [ ], recursos [ ] y [ ] .

Se habla de sistemas [ ] cuando parte o toda la gestión del sistema de información se realiza con ordenadores.

La [ ] nace con la idea es convertir el proceso artesanal de construcción de programas informáticos en una [ ] , para así producir [ ] industrial en serie y de calidad. En este sentido se consideran cuatro etapas o [ ] claramente diferenciadas:

En la primera [ ] de desarrollo de software, la construcción de programas informáticos se llevaba a cabo de una forma [ ] , esto es, planificando a groso modo las partes de un programa sin entrar a detallar el proceso de análisis y diseño previos. Se carecía de una [ ] específica y la [ ] del programador era clave en la construcción de software. Esta etapa se prolongó hasta mediados de la década de los 70.

La segunda [ ] nace con la idea de centrarse más en el [ ] del problema para afrontar con mayor garantías la [ ] . Aparecen las primeras metodologías de programación [ ] . El ya clásico "*divide y vencerás*" es el referente para diseñar programas más legibles.

El desarrollo orientado a [ ] aparece en la tercera [ ] como alternativa a las técnicas [ ] de programación estructurada. Ello supuso una forma distinta de representar la [ ] para representar y tratar los problemas de programación a través del concepto de [ ] . Dicha manera de representar la realidad supuso un mayor acercamiento a la forma más natural de pensar del ser humano. El lenguaje [ ] es el elegido para resolver problemas según esta filosofía.

Por último, en la cuarta [ ] surgen las llamadas herramientas [ ] y los [ ] de cuarta generación (centrados en el concepto de [ ]) como verdaderas herramientas para desarrollar software de una forma más rápida y eficaz, construyendo cada parte del software final según una herramienta gráfica cómoda y amigable, y [ ] el proceso de generación de código.

Enviar

5. Ciclo de vida de desarrollo de software

Supongamos que, por **ejemplo**, una **empresa de venta de ropa deportiva** decide ampliar su campo de acción considerando la posibilidad de vender, además de la forma tradicional en tienda, de manera online. Contacta con una empresa de desarrollo de aplicaciones web con el objetivo de informarse sobre **qué necesita** para ello.

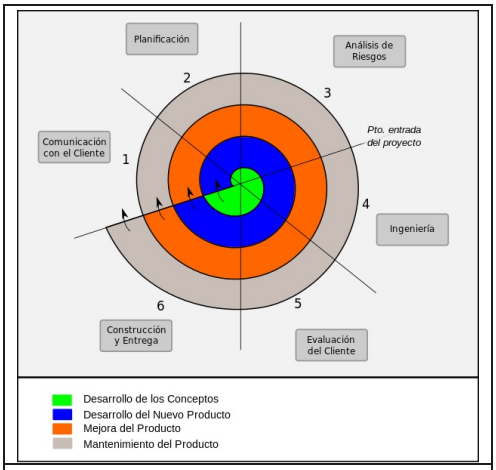


Imagen de Ebnz en wikimedia bajo licencia Creative Commonsn

En una primera supuesta reunión, la empresa llamémosle de informática (en adelante EI), necesita conocer exactamente **qué pretende** la empresa de ropa deportiva (en adelante ERD). Como vemos **ambas empresas necesitan conocer información de de la otra** para ir "puliendo" la posible solución o programa final.

En este proceso la EI pregunta, para ir construyendo **paso a paso** de una forma acertada, y muestra a la ERD una primera aproximación de dicha **solución final**. En definitiva, se hace necesario el seguimiento de una serie de pasos ordenados que garanticen un éxito aceptable.

De esta forma, **el ciclo de vida** (en adelante CV) **de una aplicación** o proyecto informático **es el conjunto de etapas y estados por los que pasa desde que se plantea como necesidad o problema, por parte del cliente, hasta que se da por terminado y se considera como una solución completa, correcta y estable** (que resuelve el citado problema inicial).

Sin entrar en detalle, las principales **etapas del CV** son las siguientes:

(a) Especificación de requisitos: en esta primera etapa el objetivo es **conocer el problema** a resolver (características, detalles, limitaciones, etc). Una vez concluida se puede estimar el coste del proyecto.

(b) Análisis: el lema "divide y vencerás" se hace patente en esta etapa. El problema principal se descompone en partes, obteniendo una serie de **subproblemas** más pequeños y abordables. En esta etapa nos centramos en el **QUÉ** y no en el CÓMO, esto es, identificar qué funciones realiza el sistema sin entrar en detalle.

(c) Diseño: para los subproblemas identificados en la fase de análisis se buscan **soluciones para posteriormente integrarlas** en una solución global. Ahora el interés está en el **CÓMO** y no en el QUÉ, describiendo cada subproblema obtenido en la etapa de análisis.

(d) Implementación: en esta etapa se **codifica**, según uno o varios lenguajes de programación, la solución diseñada.

(e) Pruebas: se **comprueba** el funcionamiento de la aplicación respetando los requisitos y necesidades fijados con el cliente.

(f) Instalación y mantenimiento: es la etapa de explotación del sistema, esto es, la puesta en **funcionamiento** en su entorno real.

En la práctica se hace necesaria cierta experiencia a la hora de completar el ciclo de vida de un proyecto software. Así, el trabajo puede realizarse en equipo y delegar las tareas correspondientes a cada etapa del ciclo de vida al equipo humano correspondiente.

De esta forma, se consigue una especialización del trabajo, asegurando o dando mayores garantías de que cada grupo de expertos se centrará únicamente en su parte o etapa, **respetando lo obtenido en la etapa anterior y proporcionando lo necesario para la siguiente**.

Aunque todo ciclo de vida de un proyecto de desarrollo de software contiene las etapas que hemos visto anteriormente, la manera de llevarlas a cabo varía según cada proyecto, ello da lugar a **distintos tipos de ciclos de vida**. Los principales son:

- Ciclo de vida **clásico o en cascada**.
- Ciclo de vida **en cascada con vuelta atrás**.
- Ciclo de vida **basado en prototipos**.
- Ciclo de vida **en espiral**.

5.1. Ciclo de vida clásico o en cascada

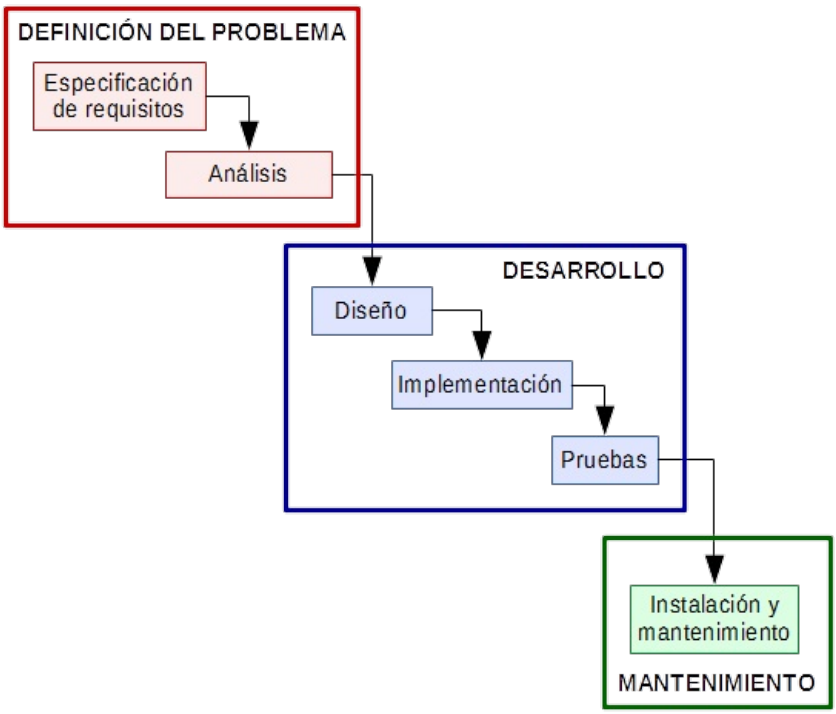
El modelo de ciclo de vida en cascada es **el modelo más simple** en desarrollo de software. En él **las etapas se llevan a cabo una detrás de otra de forma lineal**, así sólo cuando la primera fase se termina se puede empezar con la segunda, y así progresivamente.

Este modelo asume que todo se lleva a cabo y tiene lugar tal y como se había planeado en la fase anterior, y no es necesario pensar en asuntos pasados que podrían surgir en la siguiente fase. Este modelo no funcionará correctamente si se dejan asuntos de lado en la fase previa. La naturaleza secuencial del modelo **no permite volver atrás** y deshacer o volver a hacer acciones.

Este modelo es recomendable cuando **el desarrollador** ya ha diseñado y desarrollado aplicaciones similares con anterioridad, es decir, **tiene** la **experiencia** suficiente para terminar con una etapa y comenzar la siguiente.

Son tres las **fases en que se agrupan las etapas** de este tipo de ciclo de vida:

- **Definición del problema**, que incluye tanto la especificación de requisitos como el análisis del sistema.
- **Desarrollo**, que abarca el diseño, implementación y pruebas del sistema.
- **Mantenimiento**, es decir, la instalación y el mantenimiento del sistema.



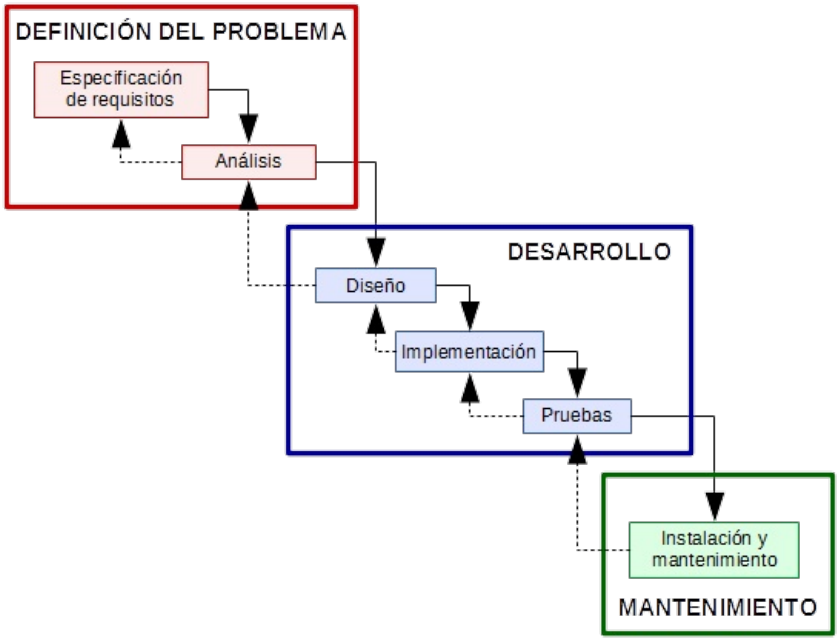
(Imagen de creación propia)

5.2. Ciclo de vida en cascada con vuelta atrás

El ciclo de vida clásico o en cascada no siempre es la mejor opción a adoptar en el desarrollo de software, **hay ocasiones en las que**:

- **una etapa del CV no puede ser completada** por no poder detallar la definición del problema. En una situación así, es necesario dejar dicha etapa sin terminar y pasar a las siguientes, para regresar más tarde a completarla.
- las decisiones tomadas en **etapas posteriores obligan a modificar otras** ya dadas por terminadas o definitivas.
- pueden **detectarse errores** cometidos en etapas ya superadas.

En estas situaciones podríamos considerar la **posibilidad de volver atrás desde cualquier etapa**, ello supone una mejora al ciclo de vida clásico.



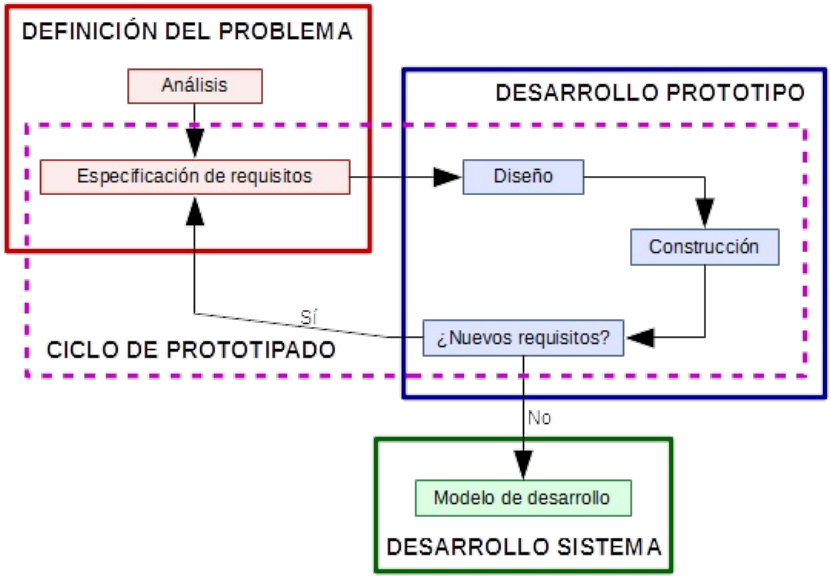
(Imagen de creación propia)

5.3. Ciclo de vida basado en prototipos

El **problema de no poder completar una etapa** por no poder detallar o definir alguna de las partes de dicho problema, puede suponer una gran dificultad para desarrollar una solución definitiva. Los llamados prototipos surgen con la idea ayudar en este sentido.

Un prototipo es un modelo inicial (no la solución final) **que se irá refinando en sucesivas pasadas**, adaptándolo a las necesidades del cliente, hasta evolucionar en la solución definitiva.

En la práctica, se parte de un modelo base inicial aproximado que **con la ayuda y aportación del cliente y usuarios se va puliendo**, desarrollando así el prototipo según dichas especificaciones hasta completar el proyecto software.



(Imagen de creación propia)

5.4. Ciclo de vida en espiral

Hasta ahora en lo referente al CV sólo se habían considerado aspectos técnicos. El tipo de ciclo de vida en espiral **tiene en cuenta además el factor riesgo**. En este sentido, es un CV que intenta **aprovechar las ventajas de los modelos anteriores pero dando importancia a los factores económicos**.

Como se observa en la figura siguiente, este tipo de CV se divide en cuatro grandes etapas: planificación, análisis de riesgos, ingeniería y evaluación. **En cada etapa se va pasando de forma circular y creciente a la siguiente**, así en cada paso es preciso valorar y considerar más recursos y trabajo (tiempo, dinero, recursos humanos, etc) para acercarse a la solución final.

En este sentido, existe una similitud con el modelo de ciclo de vida basado en prototipos pero incluyendo en cada pasada todas las partes del CV del proyecto.

(Imagen de creación propia)

Imagen en pixabay de metsi bajo licencia Creative Commons

Según las características particulares de cada proyecto software, podremos **decidirnos por un tipo de CV u otro**:

- 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 tendrán en cuenta riesgos** o 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.

En la práctica se suelen dar situaciones que incluyen parte de una o varias de estas características. Es necesario establecer formalmente una metodología de desarrollo de software que ayude en todo el proceso para garantizar una solución final acertada.

Son **muchas las metodologías existentes en el mercado**, cada una con sus ventajas e inconvenientes. Sin embargo, todas tienen una estructura común en el sentido de detallar paso por paso cada etapa a seguir.

En cada paso se suele describir el **trabajo** a desarrollar en él, los **productos** a obtener y las **técnicas** que se aconseja usar para generarlos. Los productos generados en un paso de la metodología sirven como "materia prima" para el trabajo de los siguientes.

En este sentido, **las metodologías tratan de trazar un camino repetible** que, si es seguido completa y fielmente, conduce de manera predecible a los objetivos buscados.

En el ámbito del **desarrollo de software estructurado** algunas **metodologías oficiales** son:

- Metrica v.3: Administración Pública Española.
- Merisse: Administración Pública Francesa.
- SSADM: Administación Pública de Reino Unido.

Para saber más

La metodología Métrica v.3

La metodología MÉTRICA Versión 3 ofrece a las Organizaciones un instrumento útil para la sistematización de las actividades que dan soporte al ciclo de vida del software. Metodología de Planificación, Desarrollo y Mantenimiento de sistemas de información.

Se presentan en esta página los documentos que componen la metodología MÉTRICA VERSIÓN 3.

MÉTRICA versión 3 puede ser utilizada libremente con la única restricción de citar la fuente de su propiedad intelectual, es decir, el Ministerio de Hacienda y Administraciones Públicas.

Ejercicio resuelto

¿Recuerdas el **ejemplo** planteado en la sección correspondiente al CV de desarrollo de software? Considerábamos el caso de una empresa de ropa deportiva (ERD) que acudía a una empresa de informática (EI) para el desarrollo de un software que le permitiera vender de forma online.

Te proponemos que tomes el mando y asumas tú el papel de la EI, en este sentido, **diseña los pasos a seguir** para el desarrollo de dicho software **siguiendo el modelo de ciclo de vida clásico o en cascada**.

Somos conscientes de que pueden existir varias soluciones para este problema, cada una con sus ventajas e inconvenientes. Fijamos el criterio elegido suponiendo por ejemplo que nos viene impuesto o recomendado por terceros (dirección, jefes de proyecto, etc).

Según este tipo de ciclo de vida, las fases a considerar son las siguientes:

1. **Definición del problema**, que incluye tanto la especificación de requisitos como el análisis del sistema.
2. **Desarrollo**, que abarca el diseño, implementación y pruebas del sistema.
3. **Mantenimiento**, es decir, la instalación y el mantenimiento del sistema.

El detalle de cada etapa dentro de cada fase es el siguiente:

1. DEFINICIÓN DEL PROBLEMA

(a) Especificación de requisitos

- Objetivo: venta online de ropa deportiva.
- Forma de pago: con tarjeta.
- Forma de envío: por correo.
- Pedido mínimo de un cliente: 30 €
- Detalles de un producto: nombre, precio, tallas, colores.
- Limitaciones: en el detalle de un producto concreto no incluye información relacionada con otros productos similares.

(b) Análisis

b.1) Secciones o partes

- **Página principal** con menú de enlaces a las distintas secciones o páginas.
  - Sección 1: Ropa de hombre
    - Sección 1.1: Camisetas
      - Sección 1.1.1. Manga larga
      - Sección 1.1.2. Manga corta
    - Sección 1.2: Pantalones
      - Sección 1.2.1. Pantalón largo
      - Sección 1.2.2. Pantalón corto
  - Sección 2: Ropa de mujer
    - Sección 2.1: Camisetas
      - Sección 2.1.1. Manga larga
      - Sección 2.1.2. Manga corta
    - Sección 2.2: Pantalones
      - Sección 2.2.1. Pantalón largo
      - Sección 2.2.2. Pantalón corto

b.2) Orden de navegación

P. Principal --> Sección 1 --> Sección 1.1 --> Sección 1.1.1 --> Lista productos --> Producto concreto (detalles) --> Cesta de compra --> Pasarela de pago --> Justificante de compra

b.3) Producto concreto (detalles)

- Fotografía del producto.
- Nombre, descripción, tallas y precio.
- Posibilidad de "añadir a la cesta" el producto.

b.3) Proceso de compra

- Acceder a la cesta de compra.
- Formulario de entrada de datos de cliente para facturar.
- Acceder a la pasarela de pago para entrar datos de tarjeta bancaria.
- Comprobar que los datos de compra y envío son correctos.
- Obtener justificante de compra.

2. DESARROLLO

(c) Diseño

c.1.) Página principal

Cabecera (logo, eslogan, enlace de contacto)
Menú principal (ropa de hombre, ropa de mujer)
Banner publicitario (ofertas destacadas)
Información de la empresa (dirección postal, ayuda al usuario)

c.2) Ropa de hombre

Cabecera (logo, eslogan, enlace de contacto)

Menú principal (ropa de hombre, ropa de mujer)
Menú de categorías (camisetas, pantalones)
Información de la empresa (dirección postal, ayuda al usuario)

c.3) Camisetas manga larga hombre

Cabecera (logo, eslogan, enlace de contacto)
Menú principal (ropa de hombre, ropa de mujer)
Lista de productos con foto
Información de la empresa (dirección postal, ayuda al usuario)

c.3) Sección mujer: análoga a sección hombre

c.4) Detalles de un producto

<table>
<tr><td colspan="2">Cabecera (logo, eslogan, enlace de contacto)</td></tr>
<tr><td colspan="2">Menú principal (ropa de hombre, ropa de mujer)</td></tr>
<tr><td>Foto del producto</td><td>Detalles del producto:
Nombre
Descripción
Precio
Tallas
Añadir a la cesta</td></tr>
<tr><td colspan="2">Información de la empresa (dirección postal, ayuda al usuario)</td></tr>
</table>

c.5) Cesta de compra

Cabecera (logo, eslogan, enlace de contacto)
Menú principal (ropa de hombre, ropa de mujer)
Estado actual del proceso de compra
Información de cada paso del proceso de compra • Paso 1: Mostrar cesta de compra • Paso 2: Identificación/registro de cliente • Paso 3: Comprobación/entrada de datos bancarios • Paso 4: Detalle de compra y posibilidad de impresión de justificante
Información de la empresa (dirección postal, ayuda al usuario)

(d) Implementación

Código fuente en HTML, CSS, PHP.

(e) Pruebas

Comprobación de que en cada página web la información es correcta:

- Por parte de la EI.
- Junto a la ERD.

3. MANTENIMIENTO

(f) Instalación y mantenimiento

- Instalar (subir todos los archivos al servidor web).
- Proponer posibles mejoras.

Aviso legal

El presente texto (en adelante, el "**Aviso Legal**") regula el acceso y el uso de los contenidos desde los que se enlaza. La utilización de estos contenidos atribuye la condición de usuario del mismo (en adelante, el "**Usuario**") e implica la aceptación plena y sin reservas de todas y cada una de las disposiciones incluidas en este Aviso Legal publicado en el momento de acceso al sitio web. Tal y como se explica más adelante, la autoría de estos materiales corresponde a un trabajo de la **Comunidad Autónoma Andaluza, Consejería de Educación (en adelante Consejería de Educación**).

Con el fin de mejorar las prestaciones de los contenidos ofrecidos, la Consejería de Educación se reservan el derecho, en cualquier momento, de forma unilateral y sin previa notificación al usuario, a modificar, ampliar o suspender temporalmente la presentación, configuración, especificaciones técnicas y servicios del sitio web que da soporte a los contenidos educativos objeto del presente Aviso Legal. En consecuencia, se recomienda al Usuario que lea atentamente el presente Aviso Legal en el momento que acceda al referido sitio web, ya que dicho Aviso puede ser modificado en cualquier momento, de conformidad con lo expuesto anteriormente.

1. Régimen de Propiedad Intelectual e Industrial sobre los contenidos del sitio web

1.1. Imagen corporativa

Todas las marcas, logotipos o signos distintivos de cualquier clase, relacionados con la imagen corporativa de la Consejería de Educación que ofrece el contenido, son propiedad de la misma y se distribuyen de forma particular según las especificaciones propias establecidas por la normativa existente al efecto.

1.2. Contenidos de producción propia

En esta obra colectiva (adecuada a lo establecido en el artículo 8 de la Ley de Propiedad Intelectual) los contenidos, tanto textuales como multimedia, la estructura y diseño de los mismos son de autoría propia de la Consejería de Educación que promueve la producción de los mismos.

Descargar PDF