

Ingeniería del software. Ciclo de desarrollo del software.

TEMA 48

ABACUS NT

Índice

1. *Introducción*

2. *El ciclo de vida del software*

2.1. *El ciclo de vida estructurado del proyecto*

- 2.1.1. Actividad 1: La Encuesta
- 2.1.2. Actividad 2: El Análisis de Sistemas
- 2.1.3. Actividad 3: El Diseño
- 2.1.4. Actividad 4: Implantación
- 2.1.5. Actividad 5: Generación de Pruebas de Aceptación
- 2.1.6. Actividad 6: Garantía de Calidad
- 2.1.7. Actividad 7: Descripción del Procedimiento
- 2.1.8. Actividad 8: Conversión de Bases de Datos
- 2.1.9. Actividad 9: Instalación
- 2.1.10. Resumen del Ciclo de Vida del Proyecto Estructurado

3. *Metodologías de desarrollo*

3.1. *Principales Metodologías de Desarrollo*

3.2. *Enfoques de desarrollo de software*

4. *Conclusión.*

4.1. *Relación con el sistema educativo*

5. *Bibliografía*

1. Introducción

Crisis del software

Con la progresiva implantación de los sistemas informáticos en empresas e instituciones, surge la denominada ***Crisis del Software***.

Este término, acuñado por **Dijkstra** en su libro “The Humble Programmer” -El humilde programador- se trata por primera vez en la primera conferencia de la OTAN en 1968 cuando Friedrich L. Bauer habló por primera vez del conjunto de dificultades o errores ocurridos en la planificación, estimación de los costos, productividad y calidad de un software, a raíz de graves incidentes ocurridos (por ejemplo En abril de 1986 un avión de combate F-16 se estrelló por culpa de un giro descontrolado atribuido a una expresión “if then”).

Tomando como inspiración la sistemática de otras disciplinas de ingeniería, la llegada de metodologías estructuradas de análisis y desarrollo de sistemas propició la salida de esta crisis, y aportó un auténtico enfoque de ingeniería al desarrollo de sistemas informáticos.

A partir de estos momentos hay una evolución en varias líneas o frentes:

- Surgen nuevas técnicas metodológicas como superación del ciclo de vida clásico (técnicas estructuradas).
- Empiezan a surgir nuevas herramientas de desarrollo basadas en el ordenador.
- El hardware evoluciona, aparecen nuevos entornos, máquinas cada vez más pequeñas y a la vez más potentes sobre las que son posibles nuevos sistemas operativos y nuevos entornos de desarrollo basados en las nuevas metodologías y herramientas.

Desarrollo del software

Actualmente, la tarea de desarrollo del software es una de las más complejas que existen en informática. Atrás quedó la época en que un solo programador estoicamente abordaba la elaboración de un programa completo sin pestañear. Actualmente el proceso de desarrollo del software involucra a un equipo de profesionales, incluyendo analistas, programadores especializados en unos determinados lenguajes, diseñadores gráficos, etc.



Daniel20av / CC BY-SA (<https://creativecommons.org/licenses/by-sa/3.0>)

Muchas veces tratamos de acortar el proceso de programación yendo directamente desde el análisis del problema a la codificación del mismo.

Este atajo puede resultar muy tentador e incluso parecer una buena forma de ahorrar tiempo. Sin embargo, este método necesita realmente más tiempo y esfuerzo .

El desarrollo de una solución general antes de escribir realmente el programa puede ayudar al programador a manejar el problema, tomar el camino correcto y evitar errores innecesarios.

Además, puesto que la mayoría de los programas se usarán una y otra vez, la documentación y mantenimiento de programas son partes importantes de la programación.

2. El ciclo de vida del software

ISO/IEC 12207 (2017) - Information Technology / Software Life Cycle Processes es el estándar para los procesos de ciclo de vida del software de la organización ISO.

La creación de software consta generalmente de una serie de pasos claramente diferenciados:

1. **Análisis de requisitos:** se recopila documentación, se estudian los **requisitos** del usuario final (qué se va a implementar, cuáles son las necesidades del usuario). Se suele empezar con una

descripción en lenguaje natural de lo que quiere el usuario. Al final de esta etapa tendremos una descripción clara y precisa de qué producto vamos a construir, qué funcionalidades aportará y qué comportamiento tendrá.

2. **Diseño:** una vez que sabemos qué debemos hacer, debemos determinar cómo lo haremos. Estudiaremos el problema y lo descompondremos en problemas más pequeños; definiremos la estructura de la solución, identificando los módulos que hay que implementar y sus relaciones; definiremos los algoritmos a emplear y el lenguaje de programación a utilizar, etc.
3. **Codificación:** se escribe el programa fuente en el lenguaje de programación establecido y se genera el código ejecutable.
4. **Pruebas:** se comprueba que el programa no tenga errores, y que se cumplen los criterios de calidad.
5. **Implementación:** En esta fase es en la que se instala el software en los equipos finales.
6. **Mantenimiento:** el programador se asegura de que el programa siga funcionando, y lo va adaptando también a nuevos requisitos que puedan ir surgiendo.

En el ciclo de vida clásico o convencional, cada proyecto atraviesa por algún tipo de análisis, diseño e **implantación** (implementación, prueba y mantenimiento). El ciclo de vida de proyecto utilizado, por ejemplo, en su organización, puede diferir en una o en todas las formas siguientes:

- La fase de análisis puede dividirse en una fase previa de estudio y otra de análisis (sobre todo en organizaciones en las cuales no aceptan cualquier proyecto).
- Puede no haber fase de análisis de hardware si se cree que cualquier sistema nuevo pudiera instalarse con los ordenadores existentes sin causar mayor problema operacional.
- Las fases de diseño, puede dividirse en dos: diseño preliminar y de diseño de detalles.
- Diversas fases de prueba pueden juntarse en una sola; de hecho, podrían incluirse total o parcialmente de forma paralela a la codificación.

De aquí que **el ciclo de vida del proyecto en una organización puede tener cinco fases o siete o doce**, pero seguir siendo todavía un ciclo de vida estructurado.

2.1. El ciclo de vida estructurado del proyecto

El ciclo de vida del software, se puede dividir, independientemente de la metodología empleada, en una serie de actividades que hay que realizar. En las siguientes secciones se resume cada una de esas actividades.

2.1.1. Actividad 1: La encuesta o entrevista inicial

Esta actividad también se conoce como el estudio de factibilidad o como el estudio inicial de negocios. Por lo común, empieza cuando el usuario solicita que una o más partes de su sistema se automaticen. Los principales **objetivos** de la encuesta son los siguientes:

- *Identificar a los usuarios responsables y crear un "campo de actividad" inicial del sistema.*
Esto puede comprender la conducción de una serie de entrevistas para determinar qué

usuarios estarán comprendidos en (o serán afectados por) el proyecto propuesto. Pudiera también implicar el desarrollo de un diagrama inicial de contexto, que es un diagrama de flujo de datos sencillo en el cual se representa el sistema completo con un solo proceso.

- *Identificar las deficiencias actuales en el ambiente del usuario.* Esto en general comprenderá la lista de funciones que hacen falta o que se están llevando a cabo insatisfactoriamente en el sistema actual. Por ejemplo, esto pudiera incluir declaraciones como las siguientes:
 - El hardware del sistema actual no es confiable y el vendedor se acaba de declarar en quiebra.
 - El software del sistema actual no se puede mantener, y no podemos ya contratar programadores de mantenimiento dispuestos a darle mantenimiento en el lenguaje que originalmente se utilizó para desarrollarlo.
 - El tiempo de respuesta del sistema telefónico de pedidos actual es tan malo que muchos clientes cuelgan frustrados antes de hacer su pedido.
 - El sistema actual no es capaz de producir los informes requeridos por la modificación a los impuestos decretada el año anterior.
 - El sistema actual no es capaz de recibir los informes sobre límites de crédito del departamento de contabilidad, y no puede producir los informes de promedio de volumen de pedidos que requiere el departamento de mercadotecnia.
- *Establecer metas y objetivos para un sistema nuevo.* Esto puede ser también una simple lista narrativa que contenga las funciones existentes que deben reimplantarse, las nuevas que necesitan añadirse y los criterios de desempeño del nuevo sistema.
- *Determinar si es factible automatizar el sistema y de ser así, sugerir escenarios aceptables.* Esto implicará algunas estimaciones bastante rudimentarias y aproximadas del costo y el tiempo necesarios para construir un sistema nuevo y los beneficios que se derivarán de ello; también involucrará dos o más escenarios (por ejemplo, el escenario con una computadora grande, el de procesamiento distribuido, etc.). Aunque a estas alturas la administración y los usuarios usualmente querrán una estimación precisa y detallada, el analista tendrá mucha suerte si logra determinar el tiempo, los recursos y los costos con un error menor del 50% en esta etapa tan temprana del proyecto.
- *Preparar el esquema que se usará para guiar el resto del proyecto.* Este esquema incluirá toda la información que se lista anteriormente, además de identificar al administrador responsable del proyecto. También pudiera describir los detalles del ciclo de vida que seguirá el resto del proyecto.

En general, la encuesta ocupa sólo del 5 al 10 por ciento del tiempo y los recursos de todo el proyecto, y para los proyectos pequeños y sencillos pudiera ni siquiera ser una actividad formal. Sin embargo, aun cuando no consuma mucho del tiempo y de los recursos del proyecto, es una actividad verdaderamente importante: al final de la encuesta, la administración pudiera decidir cancelar el proyecto si no parece atractivo desde el punto de vista de costo-beneficio.

Como analista, usted podrá o no estar involucrado en la encuesta; pudiera ser que antes de que siquiera se haya enterado del proyecto, el usuario y los niveles apropiados de la administración ya la hayan hecho. Sin embargo, en proyectos grandes y complejos, la encuesta requiere trabajo tan detallado que a menudo el usuario solicitará la colaboración del analista lo más pronto posible.

2.1.2. Actividad 2: El Análisis de Sistemas

El propósito principal de la actividad de análisis es transformar sus dos entradas principales, las políticas del usuario y el esquema del proyecto, en una especificación estructurada. Esto implica modelar el ambiente del usuario con diagramas de flujo de datos, diagramas de entidad-relación, diagramas de transición de estado y demás herramientas.

El proceso paso a paso del análisis de sistemas (es decir, las subactividades de la actividad de análisis) implica el desarrollo de un modelo ambiental y el desarrollo de un modelo de comportamiento. Estos dos modelos se combinan para formar el modelo esencial, que representa una descripción formal de lo que el nuevo sistema debe hacer, independientemente de la naturaleza de la tecnología que se use para cubrir los requerimientos.

Además del modelo del sistema que describe los requerimientos del usuario, generalmente se prepara un conjunto de presupuestos y cálculos de costos y beneficios más precisos y detallados al final de la actividad de análisis.

2.1.3. Actividad 3: El Diseño

La actividad de diseño se dedica a asignar porciones de la especificación (también conocida como modelo esencial) a procesadores adecuados (sean máquinas o humanos) y a labores apropiadas (o tareas, particiones, etc.) dentro de cada procesador. Dentro de cada labor, la actividad de diseño se dedica a la creación de una jerarquía apropiada de módulos de programas y de interfaces entre ellos para implantar la especificación creada en la actividad 2. Además, la actividad de diseño se ocupa de la transformación de modelos de datos de entidad-relación en un diseño de base de datos.

Parte de esta actividad le interesará como analista: el desarrollo de algo conocido como el *modelo de implantación del usuario*. Este modelo describe los asuntos relacionados con la implantación que le importan al usuario al grado de que no se los quiere confiar a los diseñadores y programadores. Los asuntos principales que suelen preocupar al usuario son aquellos relacionados con la especificación de la frontera humano-máquina y la especificación de la interfaz hombre-máquina. Esta frontera separa las partes del modelo esencial que llevará a cabo una persona (como actividad manual), de las partes que se implantarán en un o más ordenadores. De manera similar, la interfaz hombre-máquina es una descripción del formato y de la secuencia de entradas que los usuarios proporcionan al ordenador (por ejemplo, el diseño de pantallas y el diálogo en línea entre el usuario y el ordenador), además del formato y la secuencia de salidas -o productos- que el ordenador proporciona al usuario.

2.1.4. Actividad 4: Implantación

Esta actividad incluye la codificación y la integración de módulos en un esqueleto progresivamente más completo del sistema final. Por eso, la actividad 4 incluye tanto programación estructurada como implantación descendente.

Como podrá imaginar, el analista típicamente no se verá involucrado en esta actividad, aunque hay algunos proyectos (y organizaciones) donde el análisis, el diseño y la implantación de sistemas los hace la misma persona.

2.1.5. Actividad 5: Generación de Pruebas de Aceptación

La especificación estructurada debe contener toda la información necesaria para definir un sistema que sea aceptable desde el punto de vista del usuario. Por eso, una vez generada la especificación, puede comenzar la actividad de producir un conjunto de casos de prueba de aceptación desde la especificación estructurada.

Dado que el desarrollo de las pruebas de aceptación puede suceder al mismo tiempo que las actividades de diseño e implantación, pudiera ser que al analista le sea asignada esta labor al término del desarrollo del modelo esencial en la actividad 2.

2.1.6. Actividad 6: Garantía de Calidad

La garantía de calidad también se conoce como la prueba final o la prueba de aceptación. Esta actividad requiere como entradas los datos de la prueba de aceptación generada en la actividad 5 y el sistema integrado producido en la actividad 4.

El analista pudiera estar involucrado con la actividad de garantía de calidad, pero por lo regular no lo está. Pueden tomar la responsabilidad uno o más miembros de la organización usuaria, o pudiera llevarla a cabo un grupo independiente de prueba o un departamento de control de calidad. Consecuentemente, no se discutirá con más detalle la función de garantía de calidad.

Nótese, por cierto, que algunas personas le llaman a esta actividad "control de calidad" en lugar de "garantía de calidad". Sin importar la terminología, se necesita una actividad que verifique que el sistema tenga un nivel apropiado de calidad; le hemos llamado garantía de calidad en este tema. Nótese también que es importante llevar a cabo actividades de garantía de calidad en cada una de las actividades anteriores para asegurar que se hayan realizado con un nivel apropiado de calidad. Por eso, se esperaría que esto se haga durante toda la actividad de análisis, diseño y programación para asegurar que el analista esté desarrollando especificaciones de alta calidad, que el diseñador esté produciendo diseños de alta calidad y que el programador esté escribiendo códigos de alta calidad. La actividad de garantía de calidad que se menciona aquí es simplemente la prueba final de la calidad del sistema.

2.1.7. Actividad 7: Descripción del Procedimiento

Nos debe preocupar el desarrollo de un sistema completo: no sólo de la porción automatizada, sino también de la parte que llevarán a cabo las personas. Por ello, una de las actividades importantes a realizar es la generación de una descripción formal de las partes del sistema que se harán en forma manual, lo mismo que la descripción de cómo interactuarán los usuarios con la parte automatizada del nuevo sistema. El resultado de la actividad 7 es un manual para el usuario.

Como podrá imaginar, esta también es una actividad en la que pudiera verse involucrado como analista.

2.1.8. Actividad 8: Conversión de Bases de Datos

En algunos proyectos, la conversión de bases de datos involucraba más trabajo (y más planeación estratégica) que el desarrollo de programas de ordenador para el nuevo sistema. En otros casos, pudiera no haber existido una base de datos que convertir. En el caso general, esta actividad requiere como entrada la base de datos actual del usuario, al igual que la especificación del diseño producida por medio de la actividad 3.

Según sea de la naturaleza del proyecto, el analista podría tener que ver con la actividad de conversión de la base de datos. Sin embargo, no discutiremos esta actividad con mayor detalle en este tema.

2.1.9. Actividad 9: Instalación

La actividad final, desde luego, es la instalación; sus entradas son el manual del usuario producido en la actividad 7, la base de datos convertida que se creó con actividad 8 y el sistema aceptado producido por la actividad 6. En algunos casos, sin embargo, la instalación pudiera significar simplemente un cambio de la noche a la mañana al nuevo sistema, sin mayor alboroto; en otros casos, la instalación pudiera ser un proceso gradual, en el que un grupo tras otro de usuarios van recibiendo manuales y entrenamiento y comenzando a usar el nuevo sistema.

2.1.10. Resumen del Ciclo de Vida del Proyecto Estructurado

La red de flujos de datos que conectan las actividades hace ver con claridad que pudieran estarse llevando a cabo diversas actividades paralelamente.

Debido a este aspecto no secuencial, usamos la palabra actividad en el ciclo de vida del proyecto estructurado en lugar de "fase", que es más convencional. El término fase tradicionalmente se refiere a un período particular en un proyecto en el cual se estaba desarrollando una, y sólo una, actividad.

Hay otra cosa que debe recalarse acerca del uso de un diagrama de flujo de datos para describir el ciclo de vida del proyecto: un diagrama de flujo de datos clásico, no muestra en forma explícita la retroalimentación, ni el control. Prácticamente todas las actividades pueden y suelen producir información que puede llevar a modificaciones adecuadas de una o más de las actividades precedentes. De aquí que la actividad de diseño puede producir información que acaso cambie algunas de las decisiones de costo-beneficio en la actividad de análisis; de hecho, el conocimiento que se obtiene a partir de la actividad de diseño pudiera incluso llevar a revisar decisiones anteriores acerca de la factibilidad básica del proyecto.

Más aún, en casos extremos, ciertos eventos que pudieran darse en cualquier actividad pueden causar que todo el proyecto termine repentinamente. Las entradas de la administración se muestran sólo para la actividad de análisis pues ésta es la única que requiere datos de la administración; sin embargo, se supone, que la administración ejerce control sobre todas las actividades.

3. Metodologías de desarrollo

Una metodología de desarrollo es una recopilación de técnicas y procedimientos estructurados en fases que van a dar como resultado de su aplicación un producto software de calidad.

Al comienzo de la era de la informática las técnicas de desarrollo eran totalmente artesanas y sin metodología ninguna, A finales de los años setenta hay un fuerte tirón en la demanda de aplicaciones institucionales de gran complejidad con requerimientos muy exigentes a veces difíciles de identificar. Estos dos factores, aumento de demanda y aumento de complejidad, contribuyen a una importante crisis del software, que no es capaz de dar solución con las herramientas y metodologías existentes a las peticiones de los usuarios, a esto se le puede sumar el estado en que se encontraba el hardware.

A partir de estos momentos hay una evolución en varias líneas o frentes:

- Surgen nuevas técnicas metodológicas como superación del ciclo de vida clásico (técnicas estructuradas).
- Empiezan a surgir nuevas herramientas de desarrollo basadas en el ordenador.
- El hardware evoluciona, aparecen nuevos entornos, máquinas cada vez más pequeñas y a la vez más potentes sobre las que son posibles nuevos sistemas operativos y nuevos entornos de desarrollo basados en las nuevas metodologías y herramientas.

En este escenario en los años ochenta nace el término **CASE**, Computer Aided Software Engineering, o Ingeniería de Software Asistida por Ordenador, una evolución dentro del desarrollo de sistemas de información que pretende ser la solución a la crisis del software tal y como la hemos descrito anteriormente, es decir, en el entorno institucional con aplicaciones cada vez más complejas.

3.1. Principales Metodologías de Desarrollo

MetricaV3

Se definía como “metodología de planificación, desarrollo y mantenimiento de sistemas de información” y fue creada por el Ministerio de Administraciones Públicas (MAP) español.

La versión 3 de Métrica fue, y es, la más famosa de las versiones. Y aún la usan muchas empresas, ya que en muchas subcontrataciones de la administración era, y es, un requisito obligado.

Aquella era una época en la que muchos países querían tener su propia metodología: **Merise** en Francia, **SSADM** en Reino Unido y **Métrica** en España.

Agile

Con esta metodología, el desarrollo de software se divide en proyectos más pequeños, por lo que es ideal para los ciclos de lanzamiento corto y finitos, como los requeridos para las actualizaciones de un producto existente. Es conveniente cuando se tiene experiencia en la preparación de documentación, el equipo de desarrollo trabaja estrechamente y, preferiblemente, también en la misma ubicación física.

Scrum

Los proyectos que utilizan esta metodología otorgan un peso considerable a la inteligencia, experiencia y habilidades que los miembros del equipo de desarrollo aportan a la hora de resolver problemas. Las tareas de proyecto se llevan a cabo en ciclos cortos conocidos como **sprints**, muy manejables y adecuadamente priorizados, que permiten mostrar el progreso de forma muy sencilla. Los proyectos más largos se beneficiarían de la estructuración de este método, en comparación con otros modelos de desarrollo de software y uno de los motivos es que los desarrolladores se sienten comprometidos con las metas y responsables del éxito de la iniciativa.

Programación Extrema

Su enfoque de evolución constante resulta muy beneficioso, sobre todo, si se tiene en cuenta que, uno de los procedimientos a aplicar es descomponer las tareas de desarrollo para incluir sólo las actividades críticas. Precisamente por eso resulta un método muy indicado para los casos en que el entorno de desarrollo carece de estabilidad o si los requisitos aplicables al proyecto son inciertos.

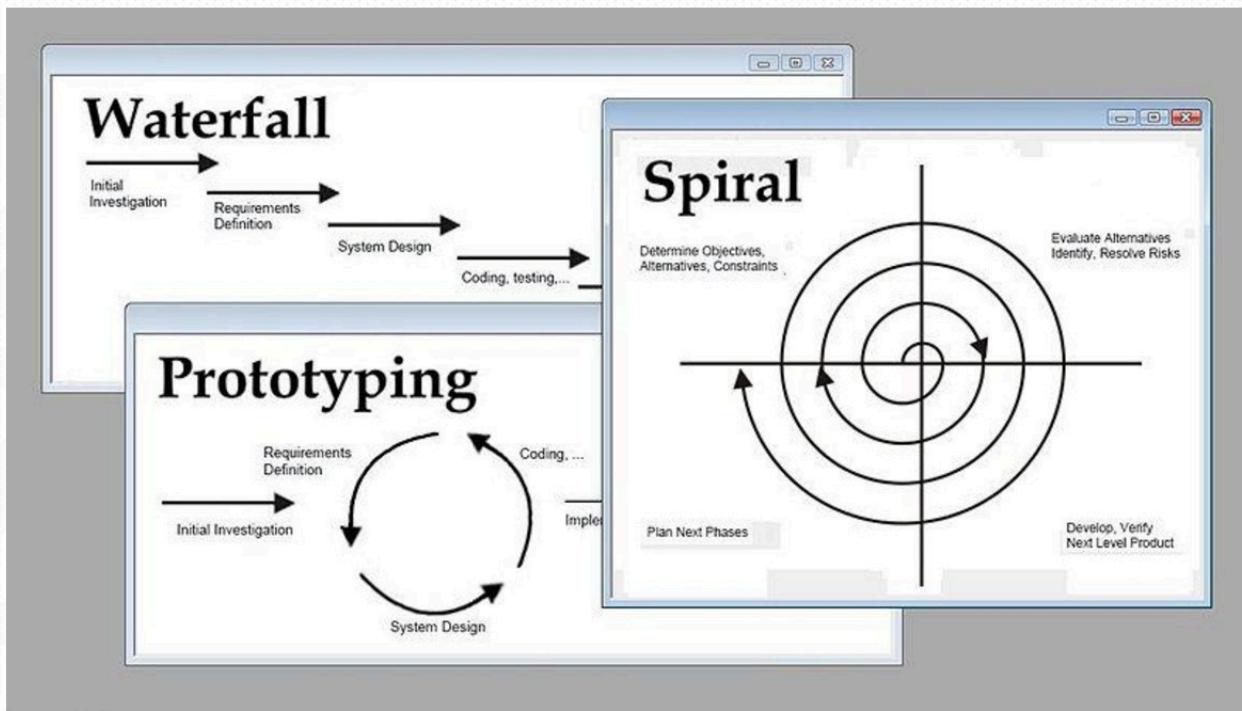
Programación en pareja

Consiste en juntar a dos programadores para que trabajen juntos en una sola estación de trabajo, facilitando que cada desarrollador se centre en diferentes aspectos del proyecto para conseguir que se reduzcan los costes de las pruebas y también los defectos en los programas. Merece la pena apostar por esta alternativa cuando el equipo de desarrollo es pequeño y seguro, y siempre que los desarrolladores estén de acuerdo.

Kanban

en base al sistema just in time, esta metodología trata de evitar los cuellos de botella, mediante una identificación muy precisa de las tareas en curso que funciona muy bien en proyectos que no sean demasiado complejos. Permite ser complementado por otras metodologías y es muy recomendable su utilización cuando se tiene confianza plena en el equipo de desarrollo.

3.2. Enfoques de desarrollo de software



Marcel Douwe Dekker - Own work by uploader, based on [Selecting a development approach](#) at cms.hhs.gov

CC BY-SA 3.0

Cada metodología de desarrollo de software tiene más o menos su propio enfoque para el desarrollo de software. Estos son los enfoques más generales, que se desarrollan en varias metodologías específicas. Estos enfoques son los siguientes:

Modelo en cascada

Es un proceso secuencial, fácil de desarrollo en el que los pasos de desarrollo son vistos hacia abajo (como en una cascada de agua) a través de las fases de análisis de las necesidades, el diseño, implantación, pruebas (validación), la integración, y mantenimiento.

Los principios básicos del modelo de cascada son los siguientes:

- El proyecto está dividido en fases secuenciales.
- Se hace hincapié en la planificación, los horarios, fechas, presupuestos y ejecución de todo un sistema de una sola vez.
- Un estricto control se mantiene durante la vida del proyecto a través de la utilización de una amplia documentación escrita, así como a través de comentarios y aprobación.

Iterativo

El prototipo permite desarrollar modelos de aplicaciones de software que permiten ver la funcionalidad básica de la misma, sin necesariamente incluir toda la lógica o características del

modelo terminado. El prototipo permite al cliente evaluar en forma temprana el producto, e interactuar con los diseñadores y desarrolladores para saber si se está cumpliendo con las expectativas y las funcionalidades acordadas. Los Prototipos no poseen la funcionalidad total del sistema pero si condensa la idea principal del mismo, Paso a Paso crece su funcionalidad, y maneja un alto grado de participación del usuario.

Incremental

Provee una estrategia para controlar la complejidad y los riesgos, desarrollando una parte del producto software reservando el resto de aspectos para el futuro.

Espiral

La atención se centra en la evaluación y reducción del riesgo del proyecto dividiendo el proyecto en segmentos más pequeños y proporcionar más facilidad de cambio durante el proceso de desarrollo, así como ofrecer la oportunidad de evaluar los riesgos y con un peso de la consideración de la continuación del proyecto durante todo el ciclo de vida.

Cada viaje alrededor de la espiral atraviesa cuatro cuadrantes básicos: (1) determinar objetivos, alternativas, y desencadenantes de la iteración; (2) Evaluar alternativas; Identificar y resolver los riesgos; (3) desarrollar y verificar los resultados de la iteración, y (4) plan de la próxima iteración.³

Cada ciclo comienza con la identificación de los interesados y sus condiciones de ganancia, y termina con la revisión y examinación.

Rapid Application Development (RAD)

El desarrollo rápido de aplicaciones (RAD) es una metodología de desarrollo de software, que implica el desarrollo **iterativo** y la construcción de **prototipos**.

Intenta reducir los riesgos inherentes del proyecto partiéndolo en segmentos más pequeños y proporcionar más facilidad de cambio durante el proceso de desarrollo.

Orientación dedicada a producir sistemas de alta calidad con rapidez, principalmente mediante el uso de iteración por prototipos, promueve la participación de los usuarios y el uso de herramientas de desarrollo computarizadas.

Hace especial hincapié en el **cumplimiento de la necesidad comercial**, mientras que la ingeniería tecnológica o la excelencia es de menor importancia.

Otros enfoques de desarrollo de software

Metodologías de desarrollo Orientado a objetos, Diseño orientado a objetos (OOD) de Grady Booch, también conocido como **Análisis y Diseño Orientado a Objetos (OOAD)**. El modelo incluye seis diagramas: de clase, objeto, estado de transición, la interacción, módulo, y el proceso.

Top-down programming, evolucionado en la década de 1970 por el investigador de IBM Harlan Mills (y Niklaus Wirth) en Desarrollo Estructurado.

Proceso Unificado, es una metodología de desarrollo de software, basado en UML. Organiza el desarrollo de software en cuatro fases, cada una de ellas con la ejecución de una o más iteraciones de desarrollo de software: creación, elaboración, construcción, y las directrices. Hay una serie de herramientas y productos diseñados para facilitar la aplicación. Una de las versiones más populares es la de **Rational Unified Process**.

4. Conclusión.

La industria del desarrollo del software se encuentra muy adelantada comparándola con su estado hace unos pocos años. Los métodos de desarrollo de software han madurado y contemplan todo el proceso de desarrollo y mantenimiento. Las herramientas CASE son cada vez más poderosas y permiten analizar, diseñar, generar, probar y mantener los productos de software de una manera integrada y controlada.

La Web se ha popularizado tanto en los últimos años, que se ha convertido en la interfaz de usuario de facto para los productos de software y ha obligado al uso de tecnologías nuevas. Las bases de datos son cada vez más sofisticadas y soportan desde un usuario en computadoras de bolsillo a miles de usuarios en mainframes. Poco a poco, el desarrollo hecho a medida se va abandonando y los negocios compran productos de software muy probados, genéricos y con una buena base instalada de clientes.

4.1. Relación con el sistema educativo

- GS-DAW/DAM – Entornos de Desarrollo

5. Bibliografía

- Pressman, R. S. **Ingeniería del Software. Un enfoque práctico**, 3^a edición. Ed. McGraw-Hill, 2000.
- Sommerville, I.: **Ingeniería de Software**. 6^a Edición. Addison-Wesley Iberoamericana, 2002.
- Piattini, M. y otros: **Análisis y diseño detallado de Aplicaciones Informáticas de Gestión**. RA-MA, 2003.
- Cerrada, J. A.: **Introducción a la Ingeniería del Software**. 1^a Edición de 2000. Editorial Centro de Estudios Ramón Areces, S.A.