Carrera: Desarrollo de software
Semestre 04

Programa de la Unidad didáctica
Métodos y modelos de desarrollo de software

Unidad 2. Modelos para el desarrollo de software

Ciudad de México, abril de 2025

Clave:

Licenciatura		TSU
15142420	**/**	**16142420**

Universidad Abierta y a Distancia de México

Índice

Presentación de la unidad 3
Logro 3
Competencia específica..... 3
Temario de la unidad 3
2. Modelos para el desarrollo de software..... 4
2.1. Modelos de diseño..... 4
2.2. Modelos de desarrollo..... 7
Cierre de la unidad..... 11
Para saber más 15
Fuentes de consulta..... 15

Presentación de la unidad

Bienvenido (a) a la segunda unidad de **Métodos y Modelos de Desarrollo de Software (MMDS)**. En la unidad anterior conociste conceptos y técnicas para diseñar diagramas UML, con el objetivo de especificar los procesos que cubren los requerimientospara la creación de sistemas solicitados por un cliente, además conociste una metodología para la administración y desarrollo de software conocido como RUP. Pero RUP no es la única metodología existente con un modelo de desarrollo que cubra a toda la gama de proyectos requeridos por los diferentes clientes, existen otros modelos de desarrollo de software, que te podrán ayudar para dar solución a esos proyectos, por ese motivo es importante en esta unidad conocer los diferentes tipos de modelos, los modelos de diseño (arquitectónicos) y los de desarrollo de software. Los modelos de diseño te servirán para definir la arquitectura que debe de existir en el sistema para que los usuarios encuentren disponibles los servicios y programas que tú desarrollarás y los modelos de desarrollo te servirán para definir la metodología apropiada para la creación de un software de acuerdo con las necesidades del cliente. Cada uno de los dos tipos de modelos contienen varios modelos, que estudiarás en el transcurso de la Unidad dos.

Al finalizar, deberás ser capaz de distinguir cada uno de los modelos de desarrollo de software para la creación de un sistema y seleccionar el adecuado para las problemáticas presentadas de los clientes.

Logro

Identificar los modelos utilizados para el desarrollo de software y diseñar gráficamente el modelo de sistemas mediante el uso de las herramientas de UML y RUP.

Competencia específica

Utilizar los modelos de desarrollo de software mediante UML y RUP para la creación de un sistema.

Temario de la unidad

2. Modelos para el desarrollo de software
   2.1. Modelos de diseño.
      2.1.1. Modelo de repositorios
      2.1.2. Modelo cliente servidor
      2.1.3. Modelo de capas
      2.1.4. Modelo de control centralizado
   2.2. Modelos de desarrollo
      2.2.1. Modelo de cascada o tradicional
      2.2.2. Modelo evolutivo
      2.2.3. Modelo basado en componentes
      2.2.4. Modelo por prototipos
      2.2.5. Modelo en espiral

2. Modelos para el desarrollo de software

Los modelos de desarrollo de software se clasifican en: modelos de diseño (arquitectónicos) y en los modelos de desarrollo de software.

Los modelos de diseño te servirán para definir la arquitectura que debe de existir en el sistema, para que los usuarios encuentren disponibles los servicios y programas que desarrollarás, entre ellos están:

- Modelo de repositorio.
- Modelo cliente – servidor
- Modelo de capas
- Modelo de control centralizado.

Los modelos de desarrollo de software te servirán para definir la metodología apropiada para la creación de un software de acuerdo con las necesidades del cliente. Entre ellos seencuentran:

- Cascada o tradicional.
- Modelos evolutivos
  - Modelo por prototipos
  - Modelo en espiral
- Modelos especializados en procesos
  - Modelo basado en componentes.

2.1. Modelos de diseño

Para este primer subtema, analizarás los modelos de diseño, entre ellos tenemos a:

- Modelo de repositorio.
- Modelo cliente – servidor
- Modelo de capas
- Modelo de control centralizado.

El grupo de modelos de diseño, pertenecen a un tipo de modelos de diseño arquitectónico, al conocer de estos temas, estas introduciéndote en el concepto de arquitectura de software. Antes de iniciar con cada modelo, **revisa** Sommerville, (2005, pp. 218-219). Ahí encontrarás la descripción de los modelos de diseño y te servirá para comprender su importancia; como: establecer estructuras básicas que identifiquen a los componentes principales de un sistema y las comunicaciones entre los componentes; es, además, importante reconocer que la arquitectura afecta al rendimiento, solidez, grado de distribución y mantenibilidad del sistema. Iniciamos con el primer modelo.

Modelo de repositorios

El modelo de **repositorio** es un modelo que abona a la arquitectura del software; su propósito es apoyar a que el software sea sólido en su funcionamiento. Este modelo de repositorio va en función de la base de datos.

Para comprender el tema, **lee** Sommerville, (2005, pp. 225-226), en estas páginas encontrarás la descripción de los modelos de **repositorio** y su importancia; su implementación física, ejemplos, ventajas y desventajas en su uso; éste te servirá para que fortalezcas tu conocimiento en la creación de bases de datos y los puntos que debes de cuidar para que se encuentre disponible en la red.

Es importante concretar que un modelo de **repositorio** es una base de datos, con una estructura pensada para que la información de diversa índole pueda estar disponible para los usuarios, en los distintos dispositivos y medios. Eso debes de tomar en cuenta cuando necesites crear un repositorio para un desarrollo.

Modelo Cliente – Servidor

Este modelo también es de arquitectura, su especialidad es brindar disponibilidad y rendimiento de una aplicación. Este modelo se centra en la disposición y organización de un conjunto de servicios y servidores asociados.

Para iniciar el tema, **lee** el modelo **cliente-servidor** de Sommerville, (2005 pp. 226-227). En esta lectura presta atención especial en la definición del modelo cliente-servidor, cuáles son sus componentes, cómo es el proceso de funcionamiento entre el cliente y el servidor, **observa** el ejemplo que se presenta las ventajas y desventajas de su uso.

No olvides que un modelo **cliente – servidor**, muestra los servicios que están disponibles en una red para los usuarios, qué es una arquitectura distribuida que fácilmente puedes agregar servidores. No debes de perder de vista que tú serás en el futuro próximo un desarrollador de software y deberás contemplar un modelo de **repositorio** y **cliente - servidor** para tus proyectos.

Modelo de capas

Recordando lo visto anteriormente: revisaste un modelo que va con la disponibilidad de las bases de datos y otro para la arquitectura de la disponibilidad de servicios que se facilitan en la red; conocerás ahora el modelo en **capas** que va en función de la arquitectura pensada para organizar un sistema y en cada capa del sistema se pone a disposición un conjunto de servicios.

Para profundizar en el tema, deberás **leer** *El modelo de capas* en Sommerville, (2005, pp. 227-229). Ahí encontrarás la descripción del modelo de **capas**, beneficios, desventajas y ejemplos. Hay más ejemplos que te ayudarán a comprender mejor el tema, por lo tanto, **revisa** los ejemplos que se muestran en el mismo libro Sommerville, (2005, pp.272-276), *tema 13.2.1 Sistemas de información y de gestión de recursos*.

Así pues, el modelo de **capas** trabajará modelando el sistema; este sistema que se creará deberá de ser pensando en **capas** y en los servicios que se ofrecerán en cada una de ellas, una puede estar la interfaz del usuario, en la que sigue el gestor de consultas a tus bases de datos, en la siguiente las políticas de los recursos y, por último, el gestor de transacciones de la base de datos. Estos cuatro niveles o **capas** son las más comunes en el desarrollo de *software*. Recuerda que su ventaja es que puedes hacer cambios en algunas de ellas, sin afectar a las demás, esto puede ser muy rápido.

Hasta este modelo has revisado modelos arquitectónicos, pero continuamos con un último modelo de este grupo.

Modelo de control centralizado

Este modelo pertenece a un subgrupo de modelos llamados: **estilos de control**, los anteriores pertenecían al subgrupo **organización del sistema** (repositorio, cliente-servidor y capas). Pero los dos subgrupos pertenecen al grupo de **diseño arquitectónico o modelo de diseño**.

En el modelo control centralizado, el sistema diseñado debe contemplar un subsistema que controla y tiene la responsabilidad de gestionar la ejecución de otros subsistemas.

Para iniciar el tema, **lee** el *Control centralizado* Sommerville, (2005, pp. 233-234). En estas páginas encontrarás la definición del modelo, la clasificación del modelo en dos clases o tipos, encontrarás dos ejemplos: el primero de tipo retorno (como los que se utilizan en los lenguajes de programación java, pascal, c, etc., y es de los más utilizados por los

principiantes en programación) y el segundo es de gestión de control centralizado. Estas lecturas te ayudarán a identificar características de los lenguajes de programación que te sirven para implementar control centralizado y la importancia de implementa una administración de concurrencia en un sistema.

A modo de resumen, recuerda que todos los modelos que hemos analizado son de tipo **arquitectónico** y que estos van en relación con el establecimiento de una estructura básica que influye en un sistema de desarrollo de software en cuanto al rendimiento, solidez, distribución y mantenibilidad del sistema. Revisaste el modelo **repositorio** que va en función de la disponibilidad de bases de datos; el modelo **cliente-servidor** en función de servicios disponibles en servidores mediante una red; el modelo de **capas**, los niveles que integran a un sistema (interfaz del usuario, gestor de consultas, políticas de los recursos y transacciones de la base de datos) y el último modelo el **control de centralizado**, que se clasifica en dos tipos *llamada-retorno* (se caracteriza por la forma estructurada de la programación en los lenguajes de 4º nivel en adelante). El modelo de **gestor** que se aplica en la concurrencia de sistemas, esto requiere de control de procesos, un proyecto típico de esto es como *software* sistema operativo. Con todo este tema, podrás adentrarte al contexto de los modelos de desarrollo de software.

2.2. Modelos de desarrollo

En el presente apartado revisarás varios modelos de desarrollo de *software*. Antes de todo conocerás una pequeña clasificación de los modelos que trabajarás en este tema, en donde se encuentra con respecto a toda la gama de modelos de desarrollo.

Roger Pressman, (2005) clasifica los modelos de desarrollo en:

- Modelos de proceso incremental:
  - El modelo incremental.
  - El modelo DRA.
- Modelos de proceso evolutivo.
  - Construcción de prototipos.
  - El modelo en espiral.
  - El modelo de desarrollo concurrente.
- Modelos especializados en procesos.
  - Desarrollo basado en componentes.
  - Modelo de métodos formales.
  - Desarrollo de *software* orientado a objetos.
- Proceso Unificado (UP).
- Cascada o tradicional.

Al modelo de **cascada** le da un tratamiento muy particular, por lo mismo no lo agrupa. De todos los anteriores, estudiaremos en este capítulo los siguientes modelos.

- Cascada o tradicional.
- Modelo evolutivo
  - Modelo por prototipos

    - Modelo en espiral
- Modelos especializados en procesos
    - Modelo basado en componentes.

A estos modelos también se les puede llamar paradigmas o ciclo de vida para el desarrollo de *software*. Con esta introducción ya estarás ubicado con el primer modelo que veremos, que es el de cascada.

Modelo de cascada o tradicional

Hablando de los modelos de desarrollo de *software* o ciclo de vida del desarrollo de *software*, hablamos de etapas por las que pasa un desarrollo, desde un inicio hasta el término del mismo, con su entrega y mantenimiento; existen varios modelos, desde los más sencillos hasta los complejos, es importante que los conozcas para que sepas cuando usar uno u otro.

El modelo de **cascada** es uno de los modelos básicos para desarrollar software, también se le conoce como el **tradicional**. Para profundizar en el tema, deberás **leer** a Weitzenfeld, (2005, pp. 50-51). En esta lectura conocerás creencias de personas involucradas en un desarrollo; es interesante que conozcas una corta historia del modelo. **Revisarás** las etapas por las que pasa un desarrollo, desde la especificación de requisitos hasta el mantenimiento, sus ventajas y desventajas.

Como habrás observado, se enlistan una serie de actividades mostradas como una cascada, dichas actividades se presentan en todos los modelos, pero pueden cambiar las circunstancias como se puedan utilizar, por lo mismo es importante que identifiques de qué se trata cada una de ellas. Para ello deberás **leer** el libro Weitzenfeld, (2005, pp.39-42). En estas páginas encontrarás una descripción detallada de cada etapa principal de los modelos de desarrollo de software, entre ellas están: requisitos, análisis, diseño, implementación, integración, pruebas, documentación y mantenimiento.

El modelo es muy sencillo, se usa cuando el proyecto no es complejo. **Recuerda** que para pasar a otra etapa del desarrollo debes haber terminado la anterior, en el modelo **tradicional** normalmente no te regresas a la etapa anterior, pero en ocasiones se acarrean errores, por lo tanto, se implementó la mejora de un método de **cascada**, en donde te permites regresar a la etapa anterior para corregir.

Modelo Evolutivo
Este es un modelo de desarrollo de software especialmente utilizado por proyectos complejos, pues requieren del constante contacto con el cliente, bajo este modelo de desarrollo, se realizan varias versiones del desarrollo hasta tener el sistema ideal.

Ahora, **revisa** el *tema Evolucionario* de Weitzenfeld, (2005, p. 52). En dicho texto encontrarás la descripción del modelo evolutivo, qué es DRA y las principales creencias del modelo. Es importante que reconozcas sus características para que sepas en qué tipo de proyectos te puede ser útil emplearlo.

Para complementar tu conocimiento **lee** *El modelo DRA*, en Pressman, (2005, pp. 53 -54). Ahí se te explicará que el modelo DRA, es un modelo de desarrollo incremental de *software*, no precisamente en paralelo, sino que se crean prototipos; primero se hace una primera versión del desarrollo y luego se planea otra versión, agregando nuevas funciones o complementando lo existente, en fin, en varios procesos vas desarrollando el *software*. Este tipo de modelo se utiliza para proyectos complejos, principalmente en donde desarrollas a pasos seguros, por eso mismo haces versiones con el paso del tiempo y entregas parciales de desarrollo.

Hasta el momento has revisado dos modelos de desarrollo, como habrás observado va enfocado en una metodología para abordar un desarrollo; las actividades principales son: requerimientos, análisis, diseño, implementación, pruebas, instalación y mantenimiento. Seguirás revisando otros modelos de desarrollo, pero antes reflexiona entorno a la siguiente pregunta: ¿identificas la diferencia entre modelos de desarrollo y modelos de diseño?, **recuerda** que la diferencia radica en que el primero va enfocado al proceso de desarrollo de software, y el segundo en cuanto al diseño arquitectónico (repositorio, capas, cliente-servidor y el control centralizado). Seguiremos con los modelos de desarrollo de *software*.

Modelo basado en componentes
Este es un modelo que se apoya en la reutilización de *software*, actualmente muchos desarrollos reutilizan código, pero en este modelo especialmente se realiza esta acción.

Revisa Somerville, (2005, pp. 64-66), en este bloque encontrarás la definición del modelo, las etapas involucradas en el desarrollo, los beneficios, y los riesgos. Esto te servirá para comprender una técnica de desarrollo de *software,* que consiste, en parte en la reutilización del *software*, como afecta en los tiempos y costos del desarrollo.

Recuerda que al reutilizar código se reduce la cantidad de *software* a desarrollar, se reduce los costos y los riesgos, permite entregas más rápidas, pero los compromisos de los requerimientos continúan, esto puede ocasionar que existan puntos clave que no cubran con las necesidades.

Modelo por Prototipos

Este modelo se aplica para la elaboración de proyectos complejos, se inicia con la construcción de prototipo rápido, esto obedece que en ocasiones un cliente o el mismo desarrollador no comprenden a detalle lo solicitado, por lo que un prototipo le ayuda a clarificar su solicitud. Para profundizar el tema, deberás **leer** Pressman, (2005, pp. 55-57). En donde entenderás cuando puedes utilizar dicho modelo, el proceso en cómo generalmente se aplica y cuando se vuelve complicado utilizarlo.

El modelo de desarrollo inicia con la comunicación con el cliente, donde se definen objetivos generales, se definen requisitos principales, se programa una iteración de desarrollo y al final se muestra un diseño rápido; en este se muestran los aspectos importantes y que son visibles para el cliente y usuario final. Una vez que se realiza el prototipo, el cliente puede afinar la solicitud de sus requerimientos. No obstante, este modelo ha ocasionado algunos problemas, cuando el cliente observa un prototipo, cree que con hacerle algunos cambios o actualizaciones quedará terminado rápido, pero no es así, pues el prototipo sirvió para darse a entender, pero, cuando el prototipo no tiene una alta calidad, debe de mejorarse y, en la mayoría de los casos, debe arrancar el proyecto desde el inicio, pues ahora ya se cuenta con todos los requerimientos. Otros de los problemas que se dan es que, en ocasiones el desarrollador se compromete a terminar el desarrollo a partir del prototipo, esto para hacer una entrega más rápida, esto ocasiona que la implementación de un algoritmo eficiente no lo sea lo suficientemente correcto, porque lo que hace el programador es adaptarlo. El último un modelo de desarrollo que revisarás es de los evolutivos.

Modelo en espiral

El modelo en **espiral** también es de los métodos de desarrollo más complejos, éste combina la construcción iterativa de prototipos usando en cada iteración las fases de desarrollo del modelo de cascada, en cada iteración se hacen incrementos del desarrollo.

Este es un modelo de desarrollo muy interesante por la forma en cómo aborda el desarrollo de proyectos complejos, los cuales son los apropiados para este tipo de modelos; por lo mismo, requiere de una metodología que ayude en la administración de todo el desarrollo. Para continuar, te solicito **leer** Pressman, (2005, pp. 58-60). En estas páginas encontrarás la definición del modelo en **espiral**, que tiene un enfoque cíclico y de incrementos de desarrollo, en las páginas del texto, observarás un gráfico del proceso de desarrollo, **leer** el tema te servirá para entender en que momentos iniciarás un proyecto utilizando este modelo de desarrollo de *software*. Para terminar el sub-tema, resumiremos que el modelo en **espiral** es un modelo que puedes utilizar para desarrollos de sistemas a gran escala, o *softwares muy complejos*; estos tipos de proyectos, tienen altos riesgos y son difícil de controlar, por eso es importante este tipo de modelo, pues con tantas iteraciones como sean necesarias y reuniones con los clientes, es más seguro terminar con un desarrollo de acuerdo a las necesidades del cliente.

Cierre de la unidad

La segunda unidad ha llegado a su conclusión. En esta unidad pudiste conocer dos tipos de modelos de desarrollo los de diseño centrado en la arquitectura del sistema y los modelos de desarrollo de software, centrados en las fases o etapas para desarrollar un *software*.

Recordando el material de tus lecturas, de los modelos arquitectónicos, analizamos el modelo ***repositorio*** que define la estructura para tener disponible las bases de datos en una red. El modelo ***Cliente –Servidor***, su importancia radica en la disposición de los servicios a través de la red, los servicios pueden estar ubicados en diferentes servidores. El modelo en ***capas*** radica en poner dentro de un mismo sistema capas para cada servicio yel modelo ***control centralizado***, este último, se clasifica en los de retorno como los lenguajes de programación que regresan un valor al terminar un método, función o procedimiento y el tipo de control centralizado que gestiona la concurrencia de sistemas y lo hace mediante el control de procesos.

Los métodos de diseño, como ya lo has visto se concentra en la arquitectura para que un usuario encuentre disponible los servicios, programas, etc.

En la presente unidad también revisaste los métodos de desarrollo, de estos viste 5 tipos. Todos los modelos están enfocados en metodología para construir *software*. Para elaborar un *software* además de los requerimientos debes de contemplar otros riesgos:

- Disponibilidad de recursos. ¿Los recursos están disponibles?
- Complejidad del proyecto. Por más que nos explican los requerimientos siguen existiendo dudas, pero es por la complejidad del desarrollo, esto sugiere ir seguros en el desarrollo.
- Entendimiento de los requerimientos. ¿Es necesario reuniones para entender los requerimientos?
- Tecnología del producto. ¿El software requiere de tecnología nueva?, la entendemos o se va aprendiendo a usar conforme se avanza en el desarrollo, ejemplo un lector biométrico, tarjetas inteligentes, etc.
- Manejo de la perspectiva del riesgo. Es un software con alto riesgo tanto como el desarrollo, o la orientación final del uso del software, aparatos médicos, equipos radiactivos, etc.
- Conocimiento y dominio del problema, entendemos todos los requerimientos.

Con base en los puntos anteriores se elaboró una tabla que te ayudará a decidir por cuál modelo decidirte; pero, aun así, si escoges el modelo más apropiado no te exime de que tengas fallas en la construcción del *software*, pues además se requiere de una metodología

de administrar los avances del desarrollo, así como al personal y los recursos, en esto pueden entrar otras metodologías y estándares, por ejemplo CMM, CMMI, o RUP, entre otros, a estos se les conoce como modelos de procesos de *software*.

La tabla es la siguiente:

CRITERIO	CASCADA	EVOLUTIVO	COMPONENTES	PROTOTIPOS	ESPIRAL
Disponibilidad de recursos.	Todos	Algunos	Algunos	Algunos	Algunos
Complejidad del proyecto.	Baja	Media	Media	Media	Alta
Entendimiento de los requerimientos	Específico	Vago	Vago	Vago	Vago
Tecnología del producto.	Existente	Nueva	Nueva	Nueva	Nueva
Manejo de la perspectiva del riesgo.	No	Si	Si	Si	Si
Conocimiento y dominio del problema	Alto	Regular	Regular	Regular	pobre

La tabla anterior puedes interpretarla de la siguiente forma desde un ejemplo:

Imagina la siguiente situación, en un hospital desean un sistema de *software* que sea alimentado por información de un electrocardiograma, cuando éste, está conectado a un paciente; el mismo sistema debe generar información impresa, evaluando con otra información que debe de ser capturada en otros momentos con entrevistas al paciente; además, la información debe de complementarse con resultados de otros estudios de laboratorios, todo ello para generar diagnósticos de enfermedades del corazón.

El proyecto se solicita con un tiempo límite de 6 meses, probado e instalado, y con las capacitaciones al personal del hospital.

Una vez analizado el problema de manera general, debes de comprender la situación especial de esta problemática, debes realizar un sistema que sea perfecto en su funcionamiento para no dar falsos pronósticos. Ahora, se realizará una sencilla evaluación para seleccionar un modelo, por lo tanto, te apoyarás de las siguientes preguntas y analizarás las respuestas hipotéticas:

- ¿Estás dispuesto a aceptar un proyecto con ese nivel de responsabilidad?, a veces podemos decir: No, pero otras veces la respuesta no depende de nosotros, pues pueden existir superiores que nos asignan tal proyecto.
- ¿Para elaborar el proyecto, tienes el personal suficiente?
- ¿Cuentas con los recursos de *hardware* y *software* para elaborar el proyecto, tanto para realizar pruebas, como para realizar una entrega del proyecto en tiempo?
- ¿Dispones de los aparatos para realizar pruebas, como es el caso del electrocardiograma, entre otros?, ¿con qué frecuencias podrás hacer uso de ellos? Recuerda que el proyecto es para 6 meses de entrega, ¿podrás levantar los detalles de los requerimientos, analizar, aprender conceptos de medicina, diseñar, modelar, codificar, probar los suficientes y variadas situaciones de enfermedades del corazón?
- Si no te prestan un electrocardiograma, ¿lo puedes conseguir?
- ¿Existirá un experto en enfermedades del corazón para revisar avances, resolver dudas, evaluar y aceptar las pruebas y con ella el producto final? O ese experto ¿estará siempre ocupado?

Analiza tú situación. Lo anterior son ideas importantes que debes de contemplar para decidir qué modelo utilizar. Con base en lo anterior debes tomar decisiones, tus opciones se encuentran en la tabla anterior:

- ¿Puedes decir qué complejo es el proyecto y catalogarlo? Recuerda las opciones son: baja, media y alta (según la tabla).
- ¿Los requerimientos son específicos o vagos para ti?
- ¿La tecnología del producto, es nueva en tu localidad, o ya existía desde hace mucho tiempo y hay muchos expertos que la dominan, tú la conoces?
- ¿El riesgo?, ¿qué impacto tendrá los errores, haciendo diagnósticos de enfermedades del corazón?, ¿podrás controlar los riesgos del proyecto? Si o No
- ¿Dominas el tema?, tus opciones: Alto, regular y pobre.

Una posible solución para el caso presentado es utilizar el método espiral, no obstante, puedes utilizar algún otro, pero también existen algunos modelos que pueden ser una mala decisión y te traerá muchos problemas. El menos recomendado sería el método de cascada, para justificar la recomendación, se desarrolló la siguiente tabla con los análisis que se hicieron: se seleccionó con colores verdes las opciones apropiadas, la tabla quedó de la siguiente forma:

CRITERIO	CASCADA	EVOLUTIVO	COMPONENTES	PROTOTIPOS	ESPIRAL
Disponibilidad de recursos.	Todos	Algunos	Algunos	Algunos	Algunos
Complejidad del proyecto.	Baja	Media	Media	Media	Alta
Entendimiento de los requerimientos	Específico	Vago	Vago	Vago	Vago
Tecnología del producto.	Existente	Nueva	Nueva	Nueva	Nueva
Manejo de la perspectiva del riesgo.	No	Si	Si	Si	Si
Conocimiento y dominio del problema	Alto	Regular	Regular	Regular	pobre

De todo lo anterior, podrás concluir que el modelo espiral es el más apropiado a utilizar para realizar el desarrollo de software en el caso presentado; los modelos evolutivo, componentes y prototipos casi cumplen con todas las características requeridas para iniciarel desarrollo del proyecto, usando alguno de estos modelos; el modelo que en definitiva nose recomienda es el modelo de cascada, si lo utilizas estarías lidiando con muchos problemas en el desarrollo y al final del proceso de desarrollo podrías entregar un proyectocon muchos errores, lo peor sería dar malos diagnósticos y empeorar la salud de los pacientes, provocando que el *software* sea inutilizable, inservible y que, al final sea abandonado, además de ser costoso en tiempo y recursos.

De todo lo anterior, podrás concluir que el modelo espiral es el más apropiado a utilizar para realizar el desarrollo de software en el caso presentado; los modelos evolutivo, componentes y prototipos casi cumplen con todas las características requeridas para iniciarel desarrollo del proyecto, usando alguno de estos modelos; el modelo que en definitiva nose recomienda es el modelo de cascada, si lo utilizas estarías lidiando con muchos problemas en el desarrollo y al final del proceso de desarrollo podrías entregar un proyectocon muchos errores, lo peor sería dar malos diagnósticos y empeorar la salud de los pacientes, provocando que el *software* sea inutilizable, inservible y que, al final sea abandonado, además de ser costoso en tiempo y recursos.

Espero este ejemplo te pueda servir para que puedas comenzar a analizar problemáticas a partir de los requerimientos del cliente y, con la tabla, apoyarte para identificar un método apropiado para que inicies tus desarrollos de software.

Con esto concluye la *Unidad 2. Modelos para el desarrollo de software*, según el dominio que tengas de esta unidad, te ayudará a comenzar con la *Unidad 3. Modelos de desarrollo de sistemas*, los cuales puedes encontrar clasificados en los modelos estructurados y orientado a objetos, al trabajar con estos temas, integrarás tus conocimientos previos de RUP y UML, para que no olvides lo que aprendiste en la Unidad uno; en la Unidad tres, estarás trabajando con todos los temas hasta hoy vistos.

Para saber más

En la presentación PPT del material de apoyo de la Unidad 2, existe un documento llamado ***ingeniería del proceso de software***, en el que encontrarás de manera resumida todos los métodos de desarrollo e incluso revisarás una tabla comparativa entre los métodos de desarrollo de software, este documento es proporcionado por el departamento de Sistemas Informáticos y Computación de la Universidad Politécnica de Valencia.

También te recomiendo que revises el material Piattini, (2004, pp. 102-105), en estas páginas encontrarás información sobre los momentos en que se integran los modelos de desarrollo en la metodología RUP, es muy importante su comprensión para que tengas una perspectiva de no solamente las etapas de desarrollo de software, sino también para que comprendas todo el proceso de la administración de proyectos de sistemas. Al final de las páginas que te indico viene un gráfico que te ayudará en su comprensión.

Fuentes de consulta

- Piattini, Mario, (2004). *Análisis y diseño de Aplicaciones Informáticas de Gestión, una perspectiva de Ingeniería del Software*. España: Alfaomega-RaMa.
- Pressman, Roger, (2005). *Ingeniería de software*. México: McGraw-Hill.
- Sommerville, Ian, (2005). *Ingeniería de Software*. Madrid España: Pearson educación.
- Weitzenfeld, Alfredo, (2005), *Ingeniería de software orientado a objetos con Uml, Java e internet*. México: Thompson Editores.