Unidad 1 / Escenario 2

Lectura fundamental

Modelos de proceso de desarrollo de *software*

Contenido

Palabras clave: *software*, procesos, modelos, desarrollo de *software*, ingeniería del *software*, procesos del *software*, cascada, espiral

1. Modelos de procesos de *software*

En el Escenario anterior se abordaron las principales características de un proceso de *software*, sin embargo, la ejecución específica del proceso de *software* difiere significativamente de acuerdo con las características de cada proyecto, entre las diferencias más relevantes se encuentran las siguientes:

- Flujo general de las actividades, acciones y tareas, así como de las interdependencias entre ellas
- Grado en el que las acciones y tareas están definidas dentro de cada actividad estructural
- Grado en el que se identifican y requieren los productos del trabajo
- Forma en la que se aplican las actividades de aseguramiento de la calidad
- Manera en la que se realizan las actividades de seguimiento y control del proyecto
- Grado general de detalle y rigor con el que se describe el proceso
- Grado con el que el cliente y otros participantes se involucran con el proyecto
- Nivel de autonomía que se da al equipo de *software*
- Grado con el que son prescritos la organización y los roles del equipo (Pressman, 2010, 14)

De acuerdo con las necesidades de los diferentes proyectos se han definido distintos modelos del proceso de *software*, también conocidos como "ciclos de vida del *software*". Estos se clasifican en dos:

1. Modelos de procesos predictivos o tradicionales: enfatizan la definición, la identificación y la aplicación detallada de las actividades y tareas del proceso. En especificar muy bien los requerimientos y el camino a seguir antes de comenzar y ceñirse a dicho camino fielmente documentando todo. Si estas metodologías no se usan con un criterio que defina que adaptaciones son necesarias, puede resultar en un proceso demasiado burocrático que genera complejidades innecesarias.
2. Modelos de procesos ágiles: su énfasis está en la maniobrabilidad y la adaptabilidad, son mucho más flexibles que los modelos tradicionales por lo que se adaptan con más facilidad a las condiciones cambiantes.

Este Escenario se enfoca en los modelos de procesos prescriptivos o tradicionales y en el Escenario 2 de la Unidad 2 se ve con más detalles las características de los modelos de procesos ágiles y las metodologías que lo aplican.

1.1. Modelo en Cascada

La **Figura 1.** Modelo en Cascada presenta la distribución de las actividades del proceso de *software* en el tiempo. Este es el primer modelo de procesos formalmente definido y es conocido como el ciclo de vida clásico. De acuerdo con diferentes autores se pueden encontrar variaciones en el número de etapas a desarrollar y los nombres asignados a los mismos, sin embargo, su característica principal es que aplica el flujo de procesos lineal, es decir que es necesario lograr todos los objetivos de una actividad para continuar con la ejecución de la siguiente. De acuerdo con Pressman (2010) Las principales etapas de este modelo son las siguientes:

- Comunicación: en dónde se busca definir con detalle el alcance y los requerimientos del proyecto, otros autores la nombran como etapa de análisis y definición de requerimientos.
- Planeación: en esta etapa se define el plan de proyecto, es decir, cuáles son las actividades que se van a desarrollar y el tiempo y demás recursos.
- Modelado: en esta etapa y a partir de los requerimientos identificados de manera previa, se define la arquitectura y la estructura general interna del *software* a implementar, en otras aproximaciones esta etapa se conoce la etapa de diseño y también se puede encontrar dividida en: diseño preliminar en donde se definen la arquitectura de alto nivel y diseño detallado en donde se incrementa el nivel de especificidad de los modelos desarrollados.
- Construcción: en esta etapa se implementan las unidades de código que reflejen el diseño definido con anterioridad. Cada una de ellas es probada para garantizar que cumple con la especificación del diseño, esta etapa es también conocida como etapa de implementación o codificación.
- Despliegue: es la etapa en que el *software* es puesto en producción, incluye también la capacitación de los usuarios y las actividades de soporte y mantenimiento. Esta etapa es también conocida como: etapa de implantación, operación y mantenimiento o funcionamiento y mantenimiento.

Figura 1. Modelo en Cascada

Fuente: elaboración propia, modificado de Pressman E (2010)

Este modelo presenta las siguientes ventajas:

- Dado que las etapas están claramente definidas, se facilita su planeación, seguimiento y control ayudando a prevenir que se sobrepasen las fechas de entrega y los costes esperados.
- En el caso de proyecto donde los requerimientos están bien definidos y hay poca probabilidad de que se presenten cambios en la especificación hay bajos riesgos.
- Este modelo muestra los pasos que son intuitivos cuando se está desarrollando *software*.

Algunas de sus desventajas son las siguientes:

- Es difícil responder a los cambios de los requerimientos del cliente ya que cualquier cambio afecta a todo el proyecto y en el mundo real es muy difícil encontrar un proyecto en donde se pueda asegurar que los requerimientos se pueden identificar y no van a sufrir variaciones.
- Pasar por cada etapa de manera estrictamente secuencial puede hacer que se pierda tiempo en el proceso ya que parte del equipo puede estar esperando a que otros terminen determinada etapa para poder continuar a la siguiente.
- Las revisiones de proyectos de gran complejidad son muy difíciles.
- Entre más tarde se detecte un error más costoso resulta su corrección.
- El usuario no podrá tener una versión funcional sino hasta en las etapas finales del proceso.

1.2. Modelo de Procesos Incremental

La **Figura 2**, Modelo de procesos incremental, presenta las actividades y el desarrollo en el tiempo del modelo de procesos incremental, el cual combina el flujo de procesos lineal y paralelo para dar respuesta adecuada a proyectos donde no es posible esperar a tener toda la especificación de requerimientos del sistema y se tienen que tener partes funcionales del sistema rápidamente e ir ampliando esta funcionalidad en entregas posteriores. La primera entrega será el producto que corresponde a las funcionalidades básicas o mínimas requeridas que se van a ir complementando con cada incremento.

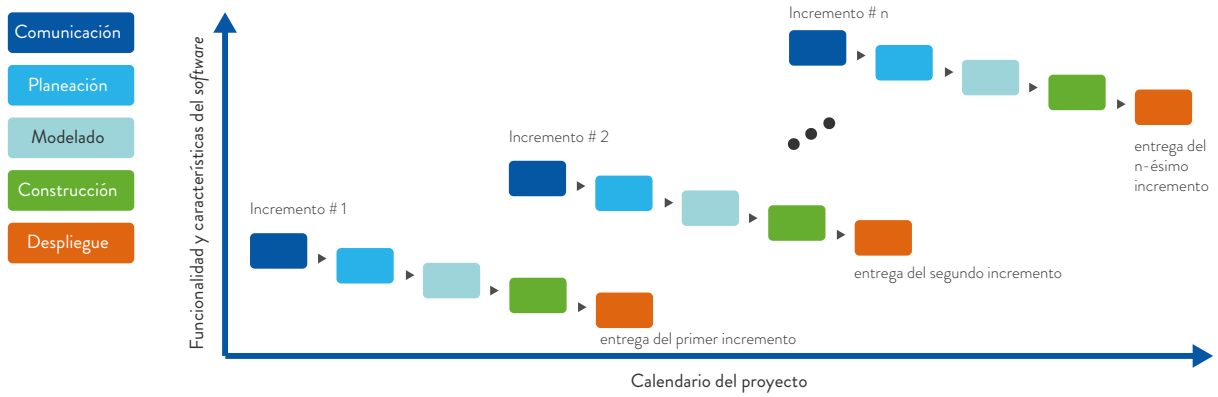

Figura 2. Modelo de procesos incremental

Fuente: elaboración propia, modificado Pressman (2010)

Las características fundamentales de este modelo son las que se explican a continuación:

- Se ajusta a entornos de alta incertidumbre donde los requerimientos no se conocen en su totalidad al inicio del proyecto o hay un alto riesgo de que sufran modificaciones.
- El sistema se crea por componentes que luego se integran para formar el sistema.
- El sistema se divide en entregas sucesivas que se van entregando e integrando a medida que se desarrollan y no un gran sistema con una única fecha de entrega.

Las principales ventajas de este modelo son que se explican a continuación.

- Se acortan los tiempos de entrega y el usuario ve algo que puede valorar del sistema con más frecuencia.
- El usuario participa más en el desarrollo de proyecto ya que las etapas en las que más participa que son las etapas de comunicación y despliegue están ocurriendo de manera iterativa.
- Al tener alguna funcionalidad más rápidamente el usuario percibe un retorno de la inversión más rápidamente.
- Al reducir la complejidad del sistema el más fácil gestionar los riesgos.

- Es más fácil gestionar los cambios en los requerimientos ya que se pueden aumentar o modificar en cada versión.
- Reduce costos, si algo sale mal solo volvemos a la antigua versión y continuamos desde ese punto, no tenemos que comenzar todo el proceso de nuevo.

Este modelo presenta las siguientes desventajas.

- Al no tener una especificación de requerimientos inicial se hace más difícil estimar el coste total del proyecto.
- Este modelo incrementa la complejidad en la gestión del proyecto, por lo que se requiere gestores experimentados.
- Difícil de aplicar a sistemas transaccionales que tienden a ser integrados y a operar como un todo.
- La interacción constante de los usuarios para retroalimentar puede hacer que se avance lentamente.
- La participación constante de los usuarios finales puede agregar costos o reducir la productividad de la empresa.

1.3. Modelo de procesos por prototipos

Este modelo de procesos, más que modificar las actividades o su flujo, agrega al modelo de cascada o incremental la posibilidad de modelar el producto final verificando algunas características antes de construirlo. La Figura 3. Modelo de procesos de prototipo presenta esta aproximación sobre el modelo de procesos en cascada, incluye la generación de prototipos al finalizar la etapa de comunicaciones, y durante la etapa de modelado incluyendo un prototipo del diseño de alto nivel y un prototipo del diseño detallado. Los prototipos tienen dos funcionalidades.

- El cliente ve el producto (en una versión no funcional) y refina sus requisitos.
- El ingeniero comprende mejor lo que va a hacer.

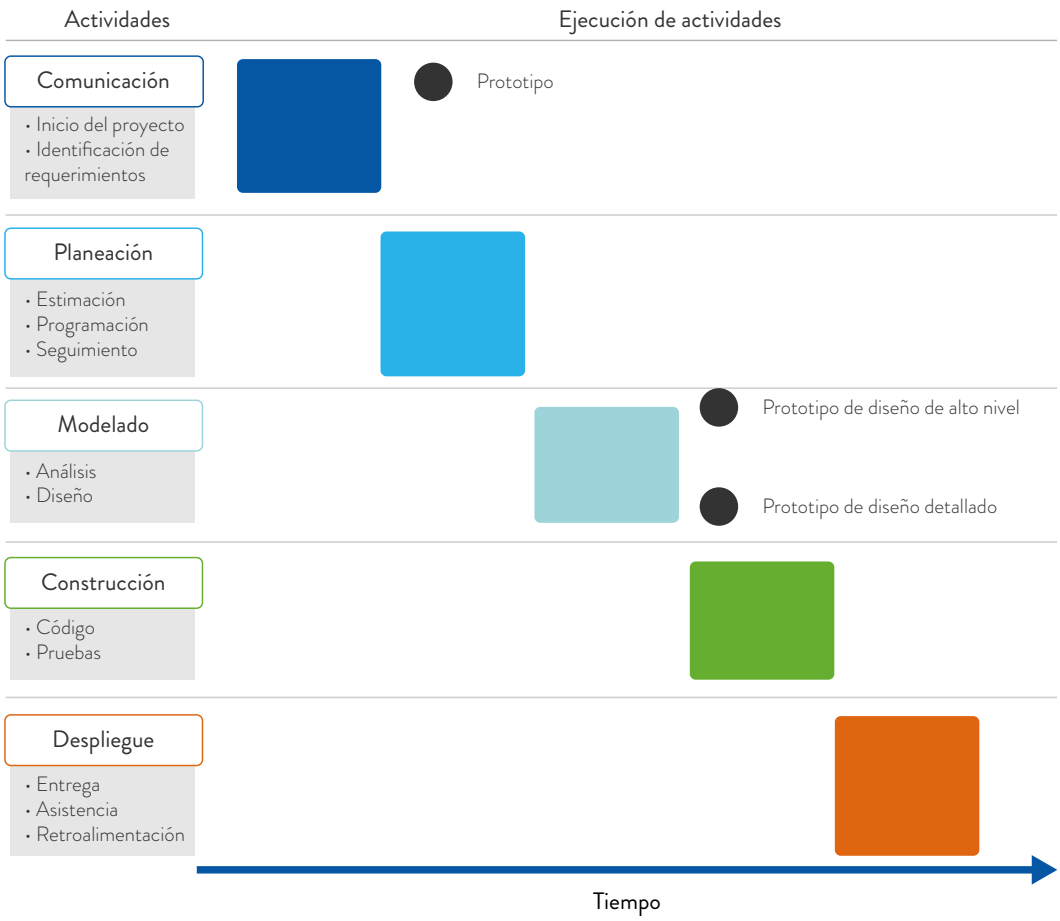

Figura 3. Modelo de procesos de prototipo

Fuente: elaboración propia

Las principales características de este modelo, se señalan a continuación.

- Reduce el riesgo de construir productos que no satisfagan las necesidades de los usuarios.
- Reduce costos y aumenta la probabilidad de éxito.
- Exige disponer de las herramientas adecuadas para la elaboración de los prototipos.
- Incrementa los riesgos de calidad y robustez en los diseños.

Las principales ventajas de esta aproximación, son las que se listan enseguida.

- Un prototipo es una buena forma de facilitar la comunicación con el usuario final.
- El cliente se va familiarizando con el nuevo producto.
- Se pueden desarrollar prototipos tanto de interfaz de usuario como de como de rendimiento.

Las desventajas más relevantes del modelo de procesos por prototipos, son las que se listan a continuación.

- La gestión de desarrollo puede hacerse lenta ya que se pueden hacer demasiadas iteraciones hacer que el usuario final este de acuerdo con el prototipo o se pongan límites.
- Imposibilidad de conocer a priori el tiempo de desarrollo.
- Dificultad para manejar las expectativas del usuario ya que la presentar una versión no funcional o parcialmente funcional al usuario final este puede subestimar el proceso de desarrollo que debe realizarse para la construcción de un producto con la calidad esperada.

1.4. Modelo de procesos en espiral

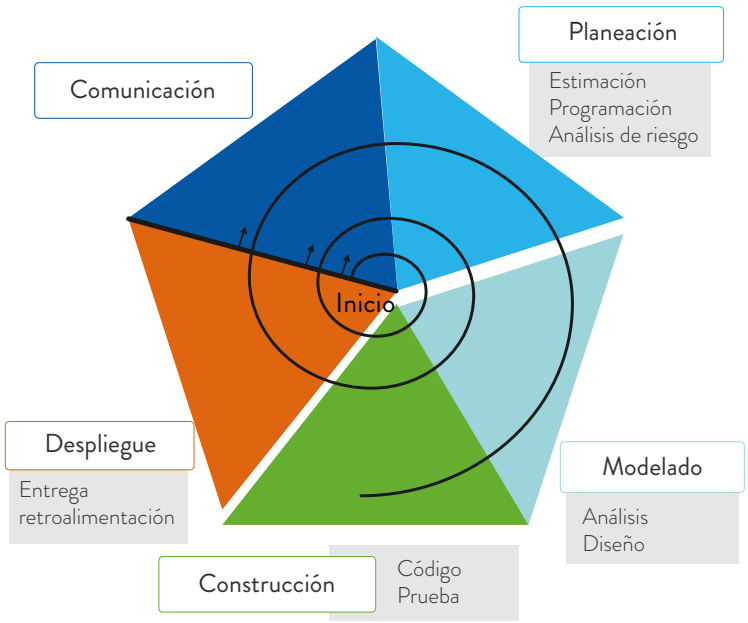

Figura 4. Modelo de procesos en espiral

Fuente: elaboración propia basada en Priessman (2010)

La **Figura 4.** Modelo de procesos en espiral se basa en el flujo de procesos evolutivo e incluye además la generación de prototipos en cada iteración. En este modelo, "el *software* se desarrolla en una serie de entregas evolutivas. Durante las primeras iteraciones, lo que se entrega puede ser un modelo o prototipo. En las iteraciones posteriores se producen versiones cada vez más completas del sistema" (Pressman, 2010, 39). Las características principales de este modelo son las siguientes.

- Cada ciclo empieza identificando los objetivos de la porción correspondiente, las alternativas de solución y las restricciones que permitan definir un alcance detallado.
- Se formula una estrategia efectiva para resolver las fuentes de riesgos (simulación, prototipado, etc.).
- Se plantea el próximo prototipo.
- Una vez resueltos los riesgos se sigue el ciclo en cascada.
- Cada ciclo se completa con una revisión que incluye todo el ciclo anterior y el plan para el siguiente.

Las principales ventajas de este modelo son las siguientes.

- No necesita una definición completa de los requisitos para empezar a implementar una iteración.
- Con la inclusión de los prototipos, desde la primera iteración se empiezan a validar los requerimientos.
- Solo se pone el riesgo el tiempo y recursos invertidos en generar cada iteración.
- Al verificar poder identificar los problemas más rápidamente se reduce los riesgos y se facilita el resolverlos a tiempo.
- Toma las ventajas de los modelos de procesos anteriores.

Las desventajas de aplicar este modelo son las siguientes.

- El cliente debe participar constantemente y esto puede generar inconvenientes en la organización.
- La gestión de proyectos con este tipo de modelos es bastante compleja por lo que requiere de gestores experimentados para garantizar el éxito del proyecto.

Como se puede observar, cada uno de los modelos expuestos tiene sus particularidades, sus ventajas y sus desventajas; las cuales se harán más evidentes de acuerdo con las características del proyecto a desarrollar. Por esta razón, es muy importante que el ingeniero de *software* tenga clara las alternativas que puede utilizar y use su criterio para definir el mejor modelo a aplicar, de acuerdo con las circunstancias de cada proyecto. Si lo requiere, que se atreva a diseñar su propio modelo de acuerdo con las necesidades.

2. Habilidades Blandas de un Ingeniero de *software*

Como se ve, en la primera parte del módulo el desarrollo del proceso de *software*, o proceso de desarrollo de *software*, es bastante complejo. Se hace aún más complejo de acuerdo con la complejidad del proyecto a desarrollar. Por esta razón, creo que este es un buen momento para poner la atención en las personas que ejecutan el proceso. Es decir, en los ingenieros de *software*. Como también se verá, en el proceso de *software* intervienen diversos roles con competencias técnicas claramente definidas. Sin embargo, en este momento lo invito a considerar cuáles son las habilidades blandas que tiene que desarrollar un ingeniero de *software* para desempeñarse exitosamente. Además, se empieza a aplicar en el desarrollo del proyecto.

De acuerdo con González (2010), las siguientes son las habilidades más importantes que debe desarrollar un ingeniero de *software*:

- **Búsqueda y clasificación de información:** no solamente se habla de las competencias técnicas de gestión de información estructurada, sino también de la capacidad de reconocer la información que es relevante en otros dominios más allá de la ingeniería del *software*. Generalmente, siempre que se construye un *software* nos enfrentamos a la necesidad de identificar lo que es relevante en su dominio de aplicación para poder construir una solución que se haga cargo de las necesidades de ese dominio. Es decir, por ejemplo, si vamos a construir un *software* que apoye los procesos contables de una organización, necesitaremos tener la habilidad de aprender del dominio de la contabilidad.
- **Capacidad de análisis:** consiste en la capacidad de identificar los elementos que componen un sistema hasta llegar a reconocer sus atributos y las relaciones que existen entre ellos.
- **Deducir, sintetizar, interpretar, analizar los fenómenos que observamos.** Esta habilidad es muy importante a la hora de modelar y proponer soluciones a través de los diseños propuestos.

- **Habilidades comunicativas:** una de las actividades básicas del proceso de ingeniería de *software* es la comunicación, ya que a través de esta podemos entender las necesidades de los clientes y los demás participantes en el desarrollo del *software* y ofrecer nuestras propuestas de solución garantizando que nos hacemos entender. Sólo a través de nuestras habilidades de comunicación podemos además coordinar las acciones requeridas para poder ejecutar el proceso de *software*. Por esta razón no deberíamos subestimar nuestras habilidades de: escuchar, leer, hablar y escribir además de la comunicación no verbal. Es mi experiencia personal he visto fracasar proyectos de *software* porque el ingeniero prefiere inferir sólo en su escritorio las necesidades de los clientes en una postura un poco arrogante antes que escuchar y sostener conversaciones con los clientes o usuarios.
- **Redacción de informes y documentos:** una habilidad comunicativa específica es la redacción de informes y documentos, como lo mencionaba anteriormente, una parte fundamental del *software* son los documentos que lo describen, el ingeniero de *software* debe estar en capacidad de comunicar a través de informes y documentos cómo está integrado y compuesto el *software* y cómo se ha desarrollado el proceso de construcción, teniendo en cuenta las competencias técnicas e intereses de los distintos públicos interesados en el *software*. No es lo mismo escribir un documento técnico dirigido a otro ingeniero que escribir un informe dirigido al patrocinador del proyecto o un manual de funcionamiento dirigido al usuario final, la información, el nivel de detalle y la forma de comunicación varían de uno a otro.
- **Habilidades de creatividad:** el ingeniero de *software* está llamado a proponer ideas y enfoques con cierto grado de originalidad, ampliando las miradas más allá de los espacios habituales y vislumbrar espacios de oportunidad para generar posibles cambios.
- **Sensibilidad frente a los problemas del cliente:** en algunas ocasiones vemos a las personas adaptándose con dificultad al diseño de un *software* en lugar de que el *software* se diseñe pensando en las necesidades del cliente. Sin embargo, los clientes de *software* son cada vez más exigentes y si el ingeniero no es sensible a las necesidades de los clientes para diseñar un *software* que se adapte a estas necesidades e incluso exceda sus expectativas seguramente será un producto destinado a fracasar.
- **Liderazgo:** es la capacidad de influir en la forma de actuar de los integrantes de un grupo de trabajo, logrando que este equipo se mantenga motivado y trabaje con entusiasmo enfocado en lograr los objetivos propuestos.
- **Toma de decisiones:** durante el proceso de ingeniería de *software* nos encontramos constantemente con la necesidad de elegir entre varias alternativas, cada una con ventajas, desventajas y riesgos asociados, esta capacidad tiene que ver con elegir una alternativa y comprometerse con esta posición gestionando la incertidumbre.

- **Gestión de conflictos:** durante el proceso de ingeniería de *software* es común que se presenten diferencias de opiniones que pueden convertirse en conflictos, con la gestión de conflictos buscamos que comprender e intervenir en la resolución pacífica no violenta de los conflictos.
- **Trabajo en equipo:** los equipos de desarrollo de *software* cada vez son más variados y complejos y el éxito del proyecto depende en gran medida de la capacidad de sus integrantes para trabajar efectivamente en equipo. Esta más que una habilidad, es un conjunto de habilidades que promueven que cada integrante conoce su papel en el equipo y se responsabiliza de cumplirlo coordinando efectivamente las acciones requeridas para el logro de los objetivos.

¿Sabía que...?

¿Obtener un buen producto de *software* depende de la correcta gestión de las 3P?

Personas, procesos y proyectos. Por esta razón es tan importantes conocer y aplicar los conceptos técnicos de la ingeniería del *software* como desarrollar las habilidades blandas que facilitan la gestión de personas y procesos.

Referencias

González-Morales, D., Moreno de Antonio, L. M. y Roda García, J. L. (2011). *Teaching "soft" skills in Software Engineering*. En la conferencia IEEE *Global Engineering Education Conference* (EDUCON), Amman, Jordania. (pp. 630-637)

Pressman, R. (2010). *Ingeniería del software. Un enfoque práctico*. (pp. 26.55) México, D.F, México: McGraw Hill

INFORMACIÓN TÉCNICA

FACULTAD DE
INGENIERÍA, DISEÑO
E INNOVACIÓN

Módulo: Ingeniería del *Software* I

Unidad 1: Ingeniería del *software* conceptos e historia

Escenario 2: Modelos de proceso de desarrollo de *software*

Autor: Diana Angélica Cruz Ortega

Asesor Pedagógico: Luisa Esperanza Rincón Jiménez

Diseñador Gráfico: Cristian Navarro

Asistente: Ginna Quiroga