EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Especialidad Análisis, diseño y desarrollo de software

Análisis Avanzados de Software

D0.3 Proceso de Software
Enfoque Somerville

Asesor: M.T.I.C. Leonardo Enriquez
Ingeniero electrónico, sistemas digitales

Actividades sobre el proceso de software

Un proceso de software es una serie de actividades relacionadas que conduce a la elaboración de un producto de software. Existen muchos diferentes procesos de software, pero todos deben incluir cuatro actividades que son fundamentales para la ingeniería de software (Sommerville, 2011):

Actividad	En que consiste
1. Especificación del software	Tienen que definirse tanto la funcionalidad del software como las restricciones de su operación.
2. Desarrollo del software	Debe desarrollarse el software para cumplir con las especificaciones.
3. Validación del software	Hay que validar el software para asegurarse de que cumple lo que el cliente quiere.
4. Evolución del software	El software tiene que evolucionar para satisfacer las necesidades cambiantes del cliente.

Actividades sobre el proceso de software

Cuando los procesos se discuten y describen, por lo general se habla de actividades como especificar un modelo de datos, diseñar una interfaz de usuario, etcétera, así como del orden de dichas actividades. Sin embargo, al igual que las actividades, también las **descripciones de los procesos** deben incluir:

Otros puntos a incluir	En que consiste
1. Productos,	Son los resultados de una actividad del proceso. Por ejemplo, el resultado de la actividad del diseño arquitectónico es un modelo de la arquitectura de software.
2. Roles,	Reflejan las responsabilidades de la gente que interviene en el proceso. Ejemplos de roles: gerente de proyecto, gerente de configuración, programador, etcétera.
3. Precondiciones y postcondiciones	Son declaraciones válidas antes y después de que se realice una actividad del proceso o se cree un producto. Por ejemplo, antes de comenzar el diseño arquitectónico, una precondición es que el cliente haya aprobado todos los requerimientos; después de terminar esta actividad, una postcondición podría ser que se revisen aquellos modelos UML que describen la arquitectura.

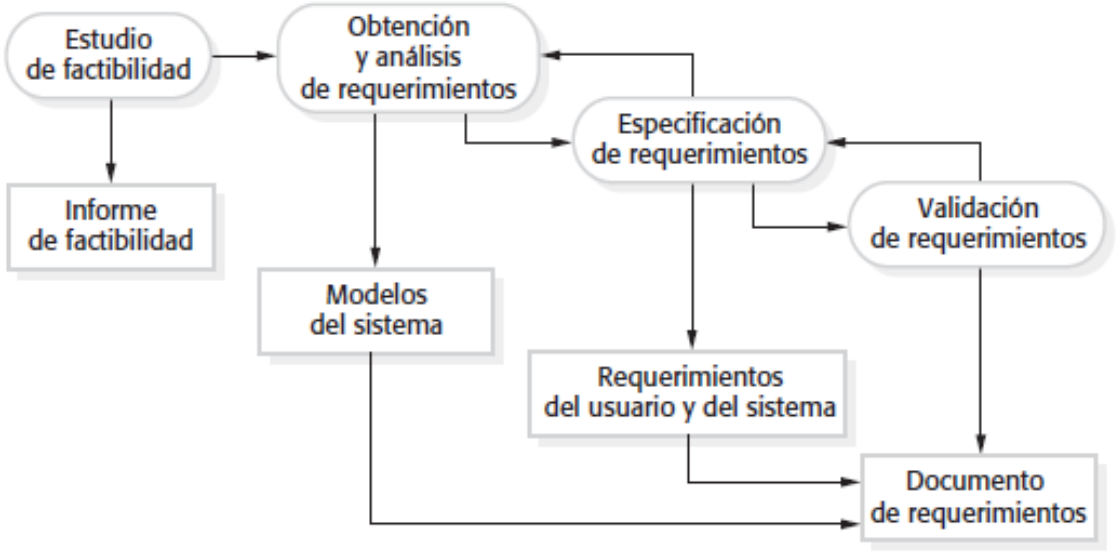
Ingeniería de requerimientos o especificación del software

La especificación del software o la ingeniería de requerimientos consisten en el proceso de comprender y definir qué **servicios se requieren del sistema**, así como la identificación de las restricciones sobre la operación y el desarrollo del sistema.

El proceso de ingeniería de requerimientos se enfoca en producir un **documento de requerimientos** especificando los requerimientos de los interesados del sistema.

Por lo general, los requerimientos se presentan en dos niveles de detalle:

- Los usuarios finales y **clientes** necesitan un informe de requerimientos de alto nivel.
- Los **desarrolladores** de sistemas precisan una descripción más detallada del sistema.



Proceso de ingeniería de requerimientos

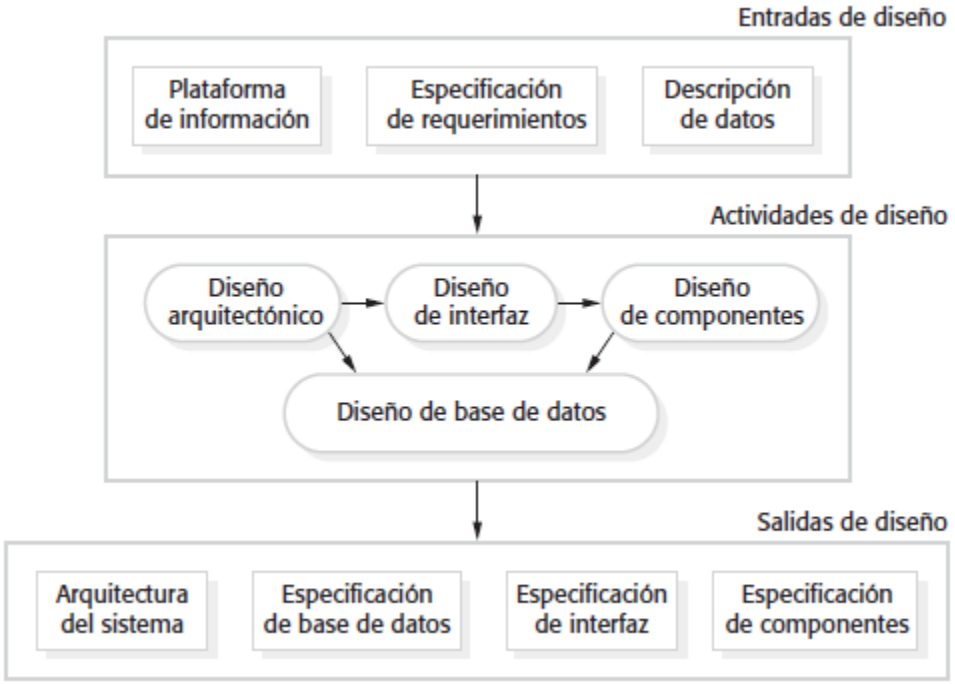
Especificación del software

Proceso de Ingeniería de requerimientos	En que consiste
1. Estudio de factibilidad	• Se realiza una **estimación** sobre necesidades identificadas del usuario y conocer si se cubren con las actuales tecnologías de software y hardware. • Se considera si el sistema propuesto tendrá un **costo-beneficio** desde un punto de vista empresarial, y si éste puede desarrollarse dentro del presupuesto existente. • El resultado debe informar la decisión respecto a si se continúa o no con un análisis más detallado.
2. Obtención y análisis de requerimientos	• Se deriva de los **requerimientos** del sistema mediante observación de los sistemas existentes, discusiones con usuarios y proveedores potenciales. • Se puede incluir el desarrollo de **uno o más modelos de sistemas** y prototipos, lo que ayuda a entender el sistema que se va a especificar.
3. Especificación de requerimientos	• Se transcribe la información recopilada durante la actividad de análisis, en un documento que define un conjunto de requerimientos, incluyen dos clases de requerimientos: Los requerimientos del **usuario** del sistema y los requerimientos del **sistema**.
4. Validación de requerimientos	• Esta actividad verifica que los requerimientos sean realistas, coherentes y completos. Al descubrir errores en el documento de requerimientos se modifican con la finalidad de corregirlos.

Diseño e implementación del software

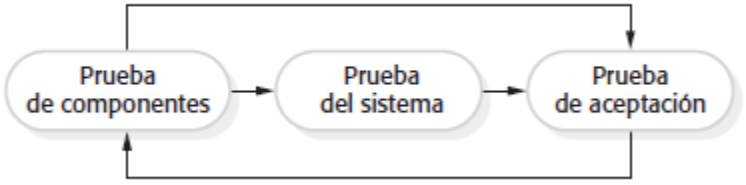
La etapa de implementación de desarrollo del software corresponde al proceso de convertir una especificación del sistema en un **sistema ejecutable**. Siempre incluye procesos de diseño y programación de software, aunque también puede involucrar la corrección en la especificación del software, si se utiliza un enfoque incremental de desarrollo.

Un diseño de software se entiende como una descripción de la estructura del software que se va a implementar, los modelos y las estructuras de datos utilizados por el sistema, las interfaces entre componentes del sistema y, en ocasiones, los algoritmos usados.



Modelo general del proceso de diseño

Por lo general, los programadores realizan algunas pruebas del código que desarrollaron. Esto revela con frecuencia defectos del programa que deben eliminarse del programa. A esta actividad se le llama depuración (***debugging***). La prueba de defectos y la depuración son procesos diferentes. La primera establece la existencia de defectos, en tanto que la segunda se dedica a localizar y corregir dichos defectos.

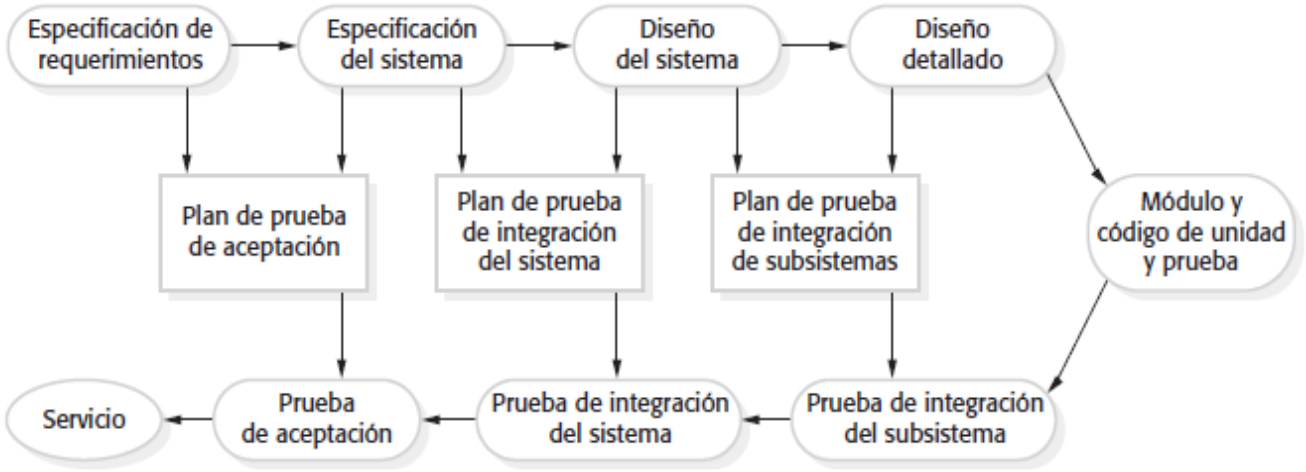


Diseño e implementación del software

Proceso de diseño para sistemas de información	En que consiste
1. Diseño arquitectónico	• Se identifica la estructura global del sistema. • Los principales componentes (llamados en ocasiones subsistemas o módulos), sus relaciones y cómo se distribuyen.
2. Diseño de interfaz	• Se definen las interfaces entre los componentes de sistemas. Esta especificación de interfaz no tiene que presentar ambigüedades. • Con una interfaz precisa, es factible usar un componente sin que otros tengan que saber cómo se implementó. • Una vez que se acuerdan las especificaciones de interfaz, los componentes se diseñan y se desarrollan de manera concurrente.
3. Diseño de componentes	• Se toma cada componente del sistema y se diseña cómo funcionará. Esto puede ser un simple dato de la funcionalidad que se espera implementar, y al programador se le deja el diseño específico. Como alternativa, habría una lista de cambios a realizar sobre un componente que se reutiliza o sobre un modelo de diseño detallado.
4. Diseño de base de datos	• Se diseñan las estructuras del sistema de datos y cómo se representarán en una base de datos.

Validación del software

La validación de software o, más generalmente, su verificación y validación (V&V), se crea para mostrar que un sistema cumple tanto con sus especificaciones como con las expectativas del cliente. Las pruebas del programa, donde el sistema se ejecuta a través de datos de prueba simulados, son la principal técnica de validación.



Fases en un proceso de software dirigido

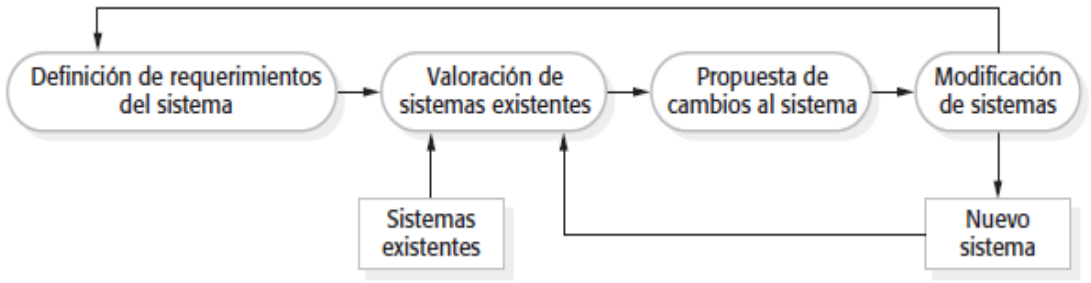
- En ocasiones, a las pruebas de aceptación se les identifica como “pruebas alfa”. El proceso de **prueba alfa** continúa hasta que el desarrollador del sistema y el cliente estén de acuerdo en que el sistema entregado es una implementación aceptable de los requerimientos.
- Cuando un sistema se marca como producto de software, se utiliza con frecuencia un proceso de prueba llamado **“prueba beta”.** Ésta incluye entregar un sistema a algunos clientes potenciales que están de acuerdo con usar ese sistema. Ellos reportan los problemas a los desarrolladores del sistema. Dicho informe expone el producto a uso real y detecta errores que no anticiparon los constructores del sistema.
- Después de esta retroalimentación, el sistema se modifica y libera, ya sea para más pruebas beta o **para su venta general**.

Validación del software

Etapas del proceso de prueba	En que consiste
1. Prueba de desarrollo	• Las personas que desarrollan el sistema ponen a prueba los componentes que constituyen el sistema. Cada componente se prueba de manera independiente, es decir, sin otros componentes del sistema. • Éstos pueden ser simples entidades, como funciones o clases de objeto, o agrupamientos coherentes de dichas entidades. • Por lo general, se usan herramientas de automatización de pruebas, como JUnit (Massol y Husted, 2003).
2. Pruebas del sistema	• Los componentes del sistema se integran para crear un sistema completo. • Este proceso tiene la finalidad de descubrir errores que resulten de interacciones no anticipadas entre componentes y problemas de interfaz de componente • Mostrar que el sistema cubre sus requerimientos funcionales y no funcionales, y poner a prueba las propiedades emergentes del sistema.
3. Pruebas de aceptación	• Se toma cada componente del sistema y se diseña cómo funcionará. Esto puede ser un simple dato de la funcionalidad que se espera implementar, y al programador se le deja el diseño específico. Como alternativa, habría una lista de cambios a realizar sobre un componente que se reutiliza o sobre un modelo de diseño detallado.

Evolución del software

El cambio es inevitable en todos los grandes proyectos de software. Los requerimientos del sistema varían conforme la empresa procura que el sistema responda a presiones externas y se modifican las prioridades administrativas. A medida que se ponen a disposición nuevas tecnologías, surgen nuevas posibilidades de diseño e implementación. Por ende, cualquiera que sea el modelo del proceso de software utilizado, es esencial que ajuste los cambios al software a desarrollar.



Evolución del sistema

Formas de enfrentar el cambio y los requerimientos cambiantes del sistema.

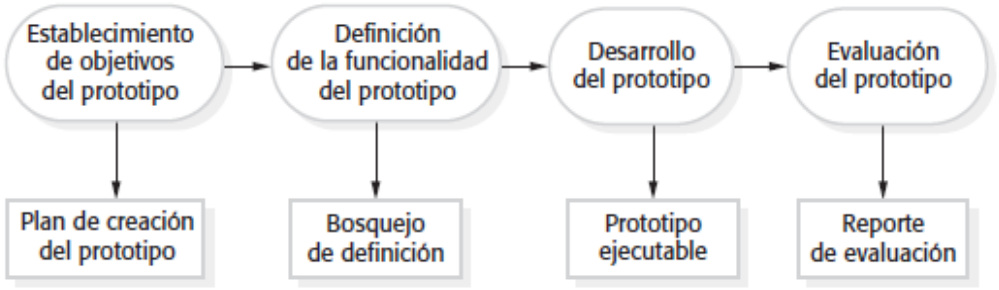
1. **Prototipo de sistema**, donde rápidamente se desarrolla una versión del sistema o una parte del mismo, para comprobar los requerimientos del cliente y la factibilidad de algunas decisiones de diseño. Esto apoya el hecho de evitar el cambio, al permitir que los usuarios experimenten con el sistema antes de entregarlo y así refinar sus requerimientos.
2. **Entrega incremental**, donde los incrementos del sistema se entregan al cliente para su comentario y experimentación. Esto apoya tanto al hecho de evitar el cambio como a tolerar el cambio. Por un lado, evita el compromiso prematuro con los requerimientos para todo el sistema y, por otro, permite la incorporación de cambios en incrementos mayores a costos relativamente bajos.

Evolución del software

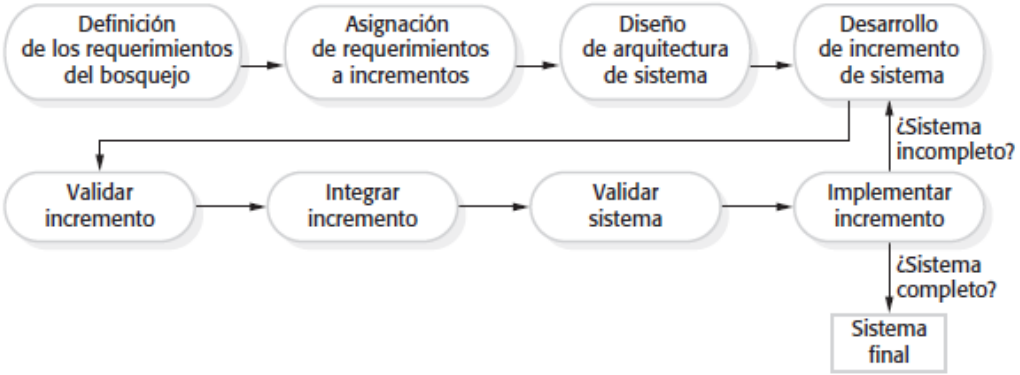
Un prototipo de software se usa en un proceso de desarrollo de software para contribuir a anticipar los cambios que se requieran:

1. En el proceso de ingeniería de requerimientos, un prototipo ayuda con la selección y validación de requerimientos del sistema.

2. En el proceso de diseño de sistemas, un prototipo sirve para buscar soluciones específicas de software y apoyar el diseño de interfaces del usuario.



La entrega incremental es un enfoque al desarrollo de software donde algunos de los incrementos diseñados se entregan al cliente y se implementan para usarse en un entorno operacional. En un proceso de entrega incremental, los clientes identifican, en un bosquejo, los servicios que proporciona el sistema. Identifican cuáles servicios son más importantes y cuáles son menos significativos para ellos. Entonces, se define un número de incrementos de entrega, y cada incremento proporciona un subconjunto de la funcionalidad del sistema.



Patrones de proceso

Patrones de proceso

Describe un problema relacionado con el proceso que se encuentra durante el trabajo de ingeniería de software, identifica el ambiente en el que surge el problema y sugiere una o más soluciones para el mismo. Dicho de manera general, un patrón de proceso da un formato [Amb98]: un método consistente para describir soluciones del problema en el contexto del proceso del software. Al combinar patrones, un equipo de software resuelve problemas y construye el proceso que mejor satisfaga las necesidades de un proyecto.

INFORMACIÓN

Conjunto de tareas

Un conjunto de tareas define el trabajo real por efectuar a fin de cumplir los objetivos de una acción de ingeniería de software. Por ejemplo, la *indagación* (mejor conocida como "recabar los requerimientos") es una acción importante de la ingeniería de software que ocurre durante la actividad de comunicación. La meta al recabar los requerimientos es entender lo que los distintos participantes desean del software que se va a elaborar.

Para un proyecto pequeño y relativamente sencillo, el conjunto de tareas para la indagación de requerimientos tendrá un aspecto parecido al siguiente:

1. Elaborar la lista de participantes del proyecto.
2. Invitar a todos los participantes a una reunión informal.
3. Pedir a cada participante que haga una relación de las características y funciones que requiere.
4. Analizar los requerimientos y construir la lista definitiva.
5. Ordenar los requerimientos según su prioridad.
6. Identificar las áreas de incertidumbre.

Para un proyecto de software más grande y complejo se requerirá de un conjunto de tareas diferente que quizá esté constituido por las siguientes tareas de trabajo:

1. Hacer la lista de participantes del proyecto.
2. Entrevistar a cada participante por separado a fin de determinar los deseos y necesidades generales.
3. Formar la lista preliminar de las funciones y características con base en las aportaciones del participante.
4. Programar una serie de reuniones para facilitar la elaboración de las especificaciones de la aplicación.
5. Celebrar las reuniones.
6. Producir en cada reunión escenarios informales de usuario.
7. Afinar los escenarios del usuario con base en la retroalimentación de los participantes.
8. Formar una lista revisada de los requerimientos de los participantes.
9. Usar técnicas de despliegue de la función de calidad para asignar prioridades a los requerimientos.
10. Agrupar los requerimientos de modo que puedan entregarse en forma paulatina y creciente.
11. Resaltar las limitantes y restricciones que se introducirán al sistema.
12. Analizar métodos para validar el sistema.

Los dos conjuntos de tareas mencionados sirven para "recabar los requerimientos", pero son muy distintos en profundidad y formalidad. El equipo de software elige el conjunto de tareas que le permita alcanzar la meta de cada acción con calidad y agilidad.

Bibliografía

Pressman, R. S. (2010). Ingeniería de Software, Un enfoque practico Séptima Edición. Ciudad de México: Mc Graw Hill.

Sommerville. (2011). Ingeniería de Software 9 Edición. Estado de México: Pearson.

UNID Universidad Interamericana para el desarrollo. (2018). Ingeniería de software. Ciudad de México: UNID.