EL PROCESO DE SOFTWARE

FAVA - Formación en Ambientes Virtuales de Aprendizaje

SENA - Servicio Nacional de Aprendizaje.

Estructura de contenidos

EL PROCESO DE SOFTWARE

Introducción

Las tareas llevadas a cabo por computadores han crecido en complejidad de manera sostenida desde su creación en la década de los cuarentas.

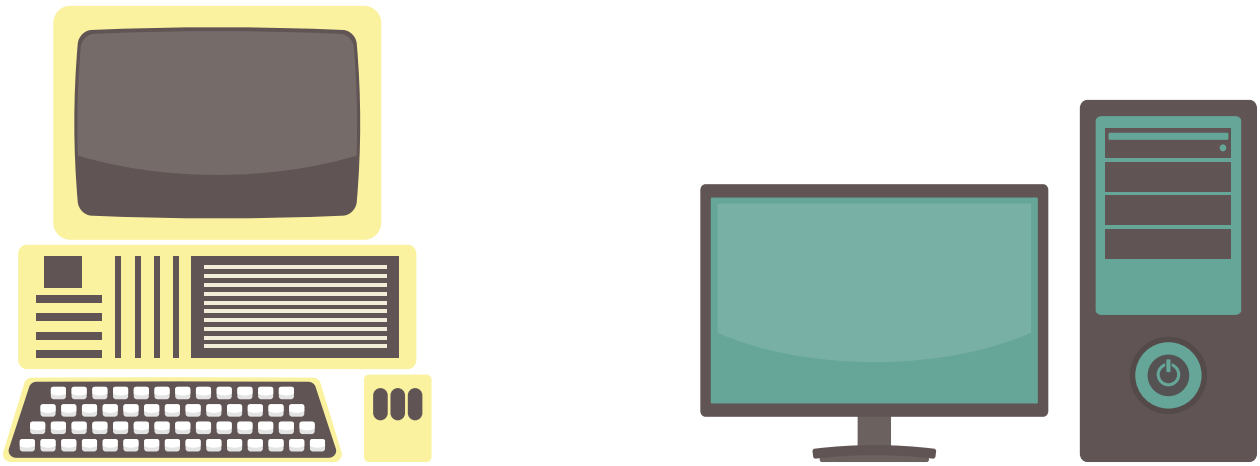
En un principio su poder de cómputo estuvo centrado en realizar complejos cálculos matemáticos para aplicar principalmente en la industria militar. Esta programación era llevada a cabo por científicos y las herramientas de programación que disponían eran limitadas.

Posteriormente cuando el poder de cómputo se hizo accesible a las grandes y medianas empresas aparecieron lenguajes de programación multi-propósito que permitieron la automatización de las tareas mecánicas que se llevaban a cabo como por ejemplo la contabilidad, costos, inventarios, entre otras.

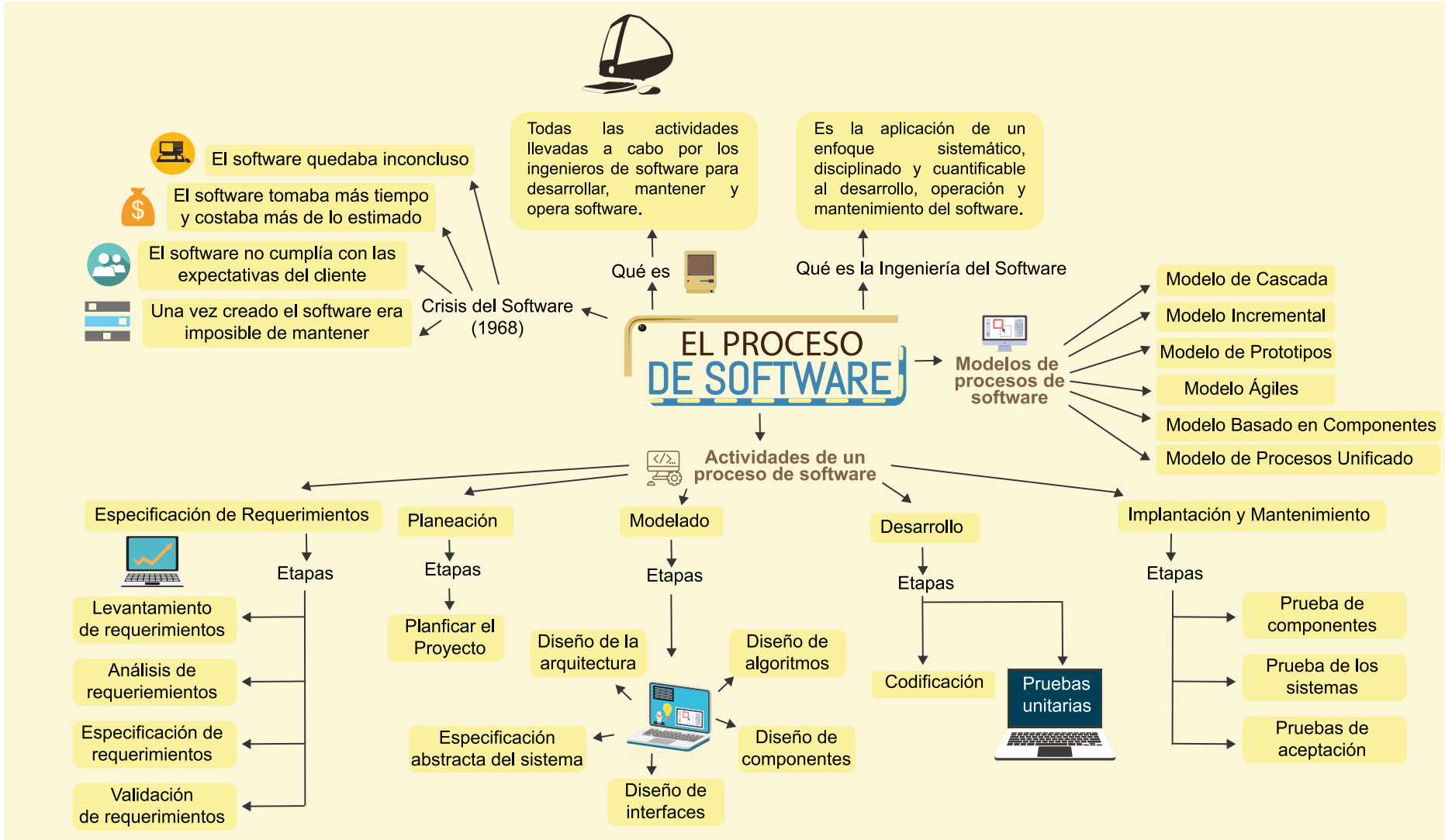
Hoy en día los computadores son accesibles a un ciudadano promedio y los sistemas informáticos llevan a cabo un amplio rango de tareas en todas las áreas del conocimiento y todo tipo de industria.

Esto ha llevado a que el proceso del software se haya vuelto tan complejo que dio origen a una disciplina llamada “Ingeniería del Software” la cual ha introducido principios, técnicas y metodologías que permiten “industrializar” este proceso.

En esta ayuda didáctica se presentan los distintos enfoques que se tienen en la actualidad para la construcción o el proceso de software.



Mapa de contenido



Generalidades

1.1 ¿Qué es el proceso del software?

Según el SWEBOK © el proceso del software son todas las actividades llevadas a cabo por los ingenieros de software para desarrollar, mantener y operar software. Estas actividades incluyen: levantamiento de requerimientos, diseño, construcción, pruebas, configuración, entre otras. (Bourque,2014).

Es este enfoque el que hace que el desarrollo de software pase de ser una tarea artesanal a un proceso industrial más sistemático, cuantificable y predecible.

Sin este enfoque de proceso industrial sería muy difícil construir sistemas complejos como por ejemplo un sistema autorizador de pagos o un sistema de punto de venta (POS), entre otros.

1.2. La crisis del software

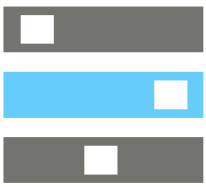
A finales del la década de los sesentas la industria del software llegó a la conclusión que la complejidad del mismo y la creciente demanda por un número mayor de aplicaciones superaba las habilidades y conocimientos que los programadores tenían en ese momento.

Lo anterior era un corolario de los problemas más comunes que se presentaban en los proyectos informáticos a saber:

El software quedaba inconcluso o el proyecto era abandonado.

El software tomaba más tiempo y costaba más de lo estimado.

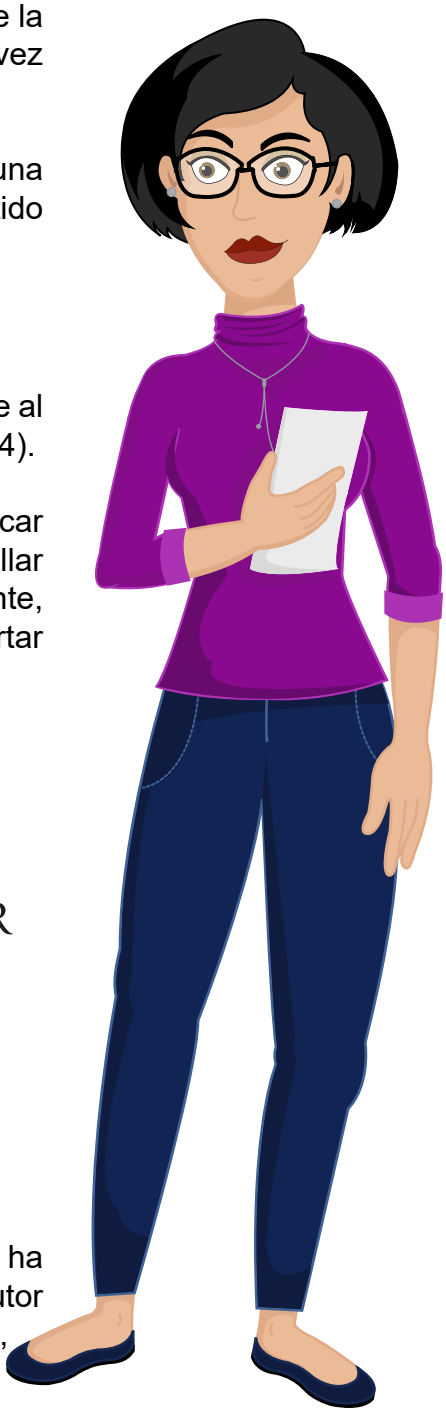
El software no cumplía con las expectativas del cliente.



Una vez creado el software era imposible o muy costoso darle mantenimiento.

Las anteriores conclusiones fueron compiladas en una conferencia de la OTAN llevaba a cabo en Alemania (NATO,1968) donde por primera vez se usó el término “Ingeniería del Software”.

A partir de entonces la ingenería del software es reconocida como una disciplina independiente cuyas técnicas y procedimientos han permitido a este proceso llegar a los niveles que hoy en día se tienen.



1.3. La ingeniería del software

Es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software (BOURQUE, 2014).

Se puede decir que la ingeniería del software se encarga de aplicar técnicas usadas en otras ámbitos de la ingeniería para lograr desarrollar productos de software que cumplan con los requerimientos del cliente, estén dentro del presupuesto y tengan la duración estimada sin importar su complejidad.

La ingeniería del software es promovida por la Computer Society (www.computer.org) de la IEEE quienes han creado el SWEBOK (Software Enginieering Body of Knowledge) que se considera la biblia de esta disciplina porque engloba sistemáticamente los diferentes autores, contenidos, técnicas y mejores prácticas de la disciplina en un solo libro.

IEEE COMPUTER SOCIETY

2. Actividades de un proceso de software

Para aplicar un enfoque sistemático al proceso de software se ha propuesto una serie de actividades que pueden variar según el autor en las siguientes (PRESSMAN, 2010): comunicación, planeación, modelado, construcción y despliegue.

En su conjunto estas etapas conformarán lo que se conoce como el ciclo de vida del software.

Estas actividades estructurales o fundamentales van acompañadas o apoyadas de otras actividades transversales o sombrilla (PRESSMAN, 2010) que ocurren a lo largo del proceso del software. Ejemplo de estas

actividades son: aseguramiento de la calidad, administración de riesgos y gerencia de proyecto, entre otras **(Figura 1)**.

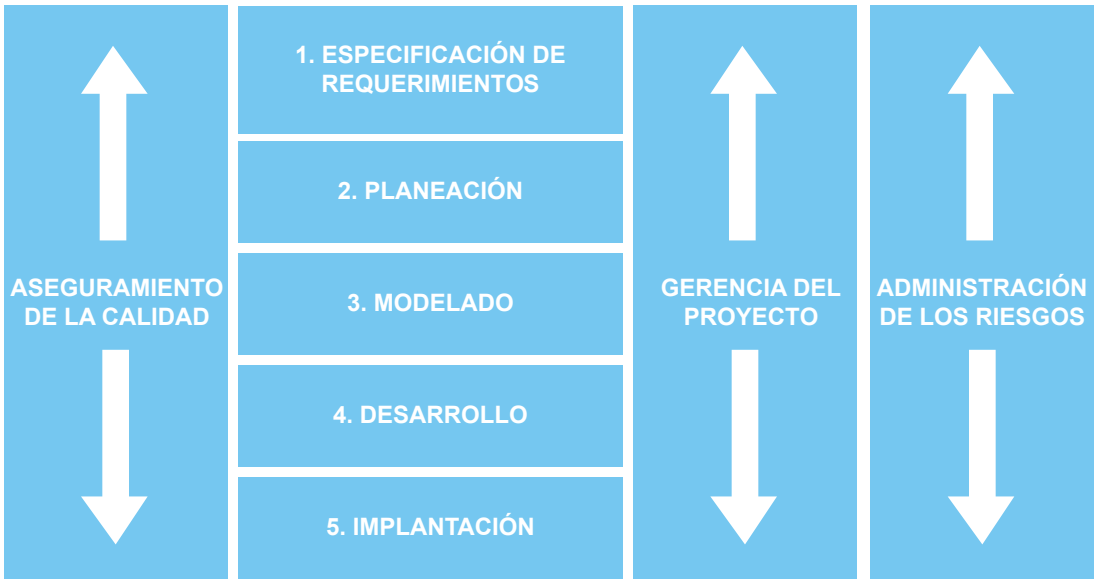


Figura 1. Actividades de un proceso de software

2.1. Especificación de requerimientos

En esta fase los interesados en el proyecto o stakeholders acuerdan y establecen los requerimientos que el software como producto debe cumplir.

Esta actividad se puede descomponer en las siguientes etapas (SOMMERVILLE,2005):

a) Obtención de requerimientos.

En esta etapa se hace un inventario de los requerimientos que debe cumplir el software a desarrollar. Se apoya en entrevistas, cuestionarios, recolección de datos a los interesados claves del proyecto. Al final estos requerimientos son consolidados para entregarlos a la siguiente etapa.

b) Análisis de requerimientos

En esta etapa se determina si un requerimiento es viable dentro de las restricciones tecnológicas, de costo y tiempo que tienen todos los proyectos. El analista deberá entender muy bien el requerimiento y asegurar que el mismo sea único, medible y alcanzable, entre otros.

c) Especificación de requerimientos

Una vez analizado el requerimiento ha sido individualizado se procede a realizar la especificación formal de manera que pueda ser socializado con todos los interesados del proyecto.

d) Validación de requerimientos

En esta etapa se procede a validar que los requerimientos cumplan con las características inherentes como son: no redundante, completo, alcanzable, entre otros.

Las anteriores etapas no necesariamente se deben hacer secuencialmente sino que se pueden hacer en forma iterativa **(Figura 2)**, es decir, que un mismo requerimiento puede ser varias veces analizado, especificado o validado con el objeto que el conjunto de requerimientos finales sean más coherentes, no redundantes, completos y que cumplan con el objeto del desarrollo del software. (SOMMERVILLE, 2005).

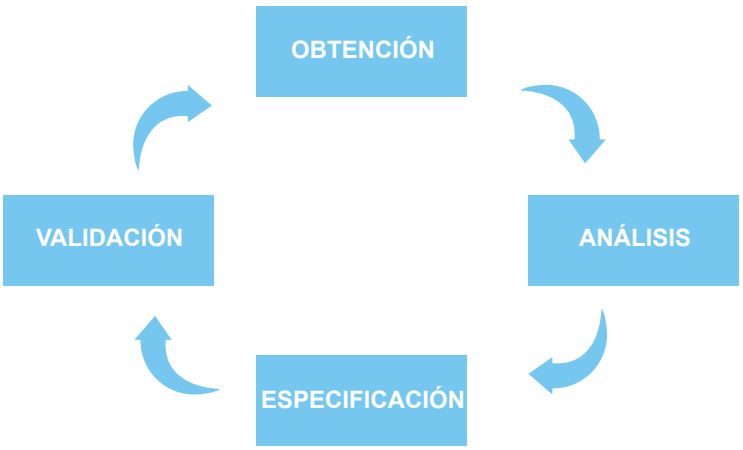


Figura 2. Etapas de la especificación de requerimientos

2.2. Planeación

En esta etapa se especifica un plan para el proyecto que guiará a todos los interesados (stakeholders) a conseguir el producto especificado en la primera fase.

Según Pressman la actividad de planeación es “un conjunto de prácticas administrativas y técnicas que permiten que el equipo de software defina un mapa mientras avanza hacia su meta estratégica y objetivos tácticos” (PRESSMAN, 2010).

Hoy en día los proyectos de desarrollo de software empresarial se están apoyando en los principios y prácticas proporcionados por el PMBOK del Project Management Institute (PMI) .

2.3. Modelado

En esta etapa los analistas o ingenieros de software crean un modelo abstracto del sistema a construir basado en los requerimientos funcionales y no funcionales planteados en la fase de especificación de requerimientos. Este modelo proporciona detalles sobre arquitectura del software, estructuras de datos, interfaces y componentes que se necesitan para implementar el sistema (PRESSMAN, 2010).

Son actividades propias de esta fase las siguientes (SOMMERVILLE, 2005):

a) Diseño de la arquitectura
Se identifican y documentan los subsistemas que forman el sistema y sus relaciones.

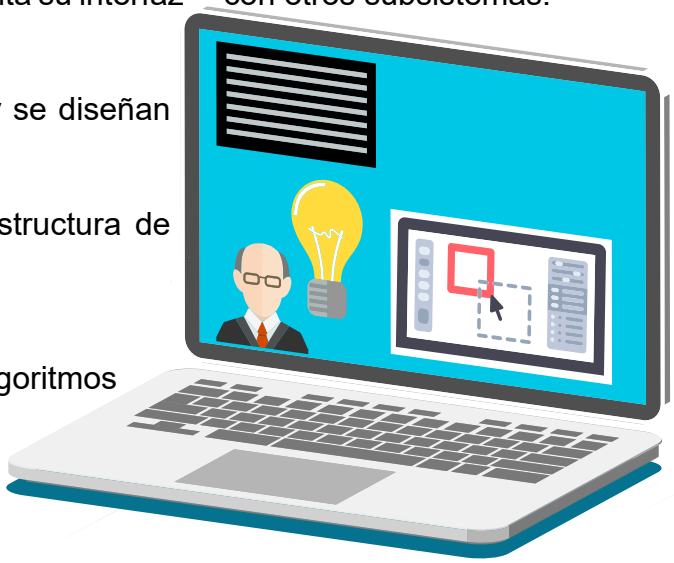
b) Especificación abstracta del sistema
Para cada subsistema se produce una especificación abstracta de sus servicios y las restricciones sobre las cuales debe funcionar.

c) Diseño de interfaces
Para cada subsistema se diseña y documenta su interfaz con otros subsistemas.

d) Diseño de componentes
Se asignan servicios a los componentes y se diseñan

e) Diseño de las estructuras de datos
Se diseña en detalle y se especifica la estructura de implementación del sistema.

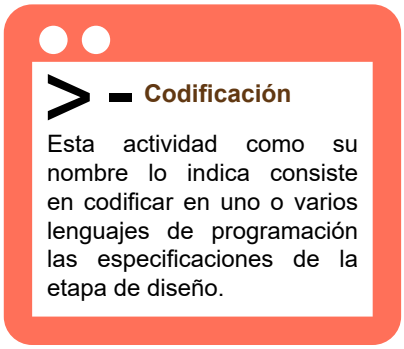
f) Diseño de algoritmos
Se diseñan en detalle y se especifican los algoritmos para proporcionar los servicios.

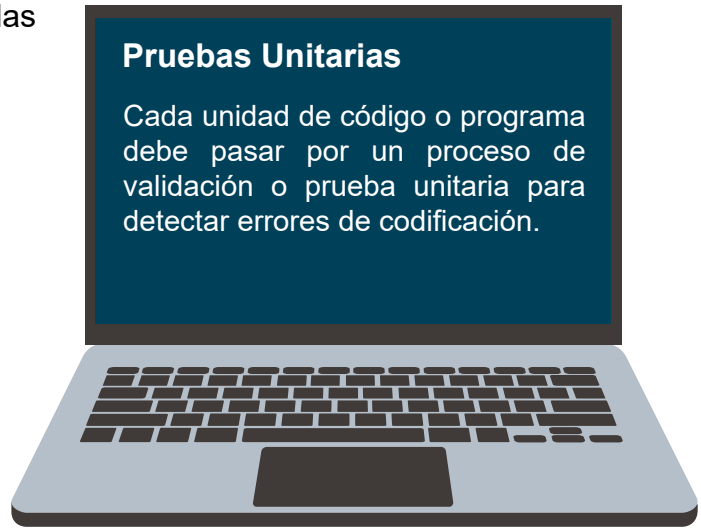


2.4. Desarrollo

En esta actividad ,también llamada construcción, se genera el código fuente en el lenguaje o lenguajes de programación seleccionados para el proyecto. El entregable de esta actividad además del código puede ser también un conjunto de parámetros o parametrización de software.

Son actividades de esta etapa las siguientes:





Esta actividad se puede apoyar también en herramientas de cuarta generación o CASE que generan código a partir de una especificación formal. Un ejemplo de este tipo de herramientas es GENEXUS (www.genexus.com) producida por una empresa uruguaya de software.

También se pueden usar sistemas integrados de desarrollo o IDE's como ECLIPSE (www.eclipse.org) o Microsoft Visual Studio Code, entre otros.

En esta etapa es importante seguir algunos principios de programación (PRESSMAN, 2010):

1. Reducir la complejidad.
2. Anticiparse a la diversidad o cambios.
3. Facilitar las pruebas de software.
4. Ajustarse a la normatividad vigente para el sistema a desarrollar.

2.5. Implantación

En esta fase se lleva a cabo la puesta en marcha o salida en vivo del producto de software desarrollado una vez los requerimientos son validados y aceptados por el cliente.

Antes de la puesta en marcha se realiza una validación y prueba del software que consta de las siguientes actividades (SOMMERVILLE, 2005):

- Prueba de componentes.
- Prueba de los sistemas.
- Prueba de aceptación.

Posterior a la salida en vivo se pueden presentar situaciones que requieran algunas de las siguientes actividades propias del mantenimiento del software:

- Corrección de errores de programación.
- Implementación de nuevos requerimientos.
- Ajustes.

3. Modelos de procesos de software

Las actividades del proceso de software vistas en el numeral anterior fueron en un principio secuenciales y formaron parte del modelo en Cascada (PRESSMAN, 2010).

Sin embargo en la medida que los sistemas se hicieron más complejos surgieron nuevos enfoques como el modelo en espiral o modelos ágiles que pretendían disminuir costos o afrontar los riesgos e incertidumbres propios de los sistemas actuales donde los requerimientos no siempre están totalmente especificados o son muy susceptibles a cambios.

El programa ADSI se guiará por este modelo de procesos en cascada, no obstante, se exponen otros modelos a manera de ejemplo para que el estudiante tenga una visión más amplia y pueda profundizar más adelante.

3.1. Modelo en cascada

En este modelo las etapas del proceso a saber: especificación de requerimientos, modelado, desarrollo e implantación se dan de manera secuencial **(Figura 3).**

Este fue el primer modelo propuesto y sigue siendo el padre de los demás modelos de procesos.(PRESSMAN, 2010).

Este modelo es apropiado cuando los requerimientos de un sistema están claramente definidos con anticipación y no se espera que cambien en el transcurso de las demás etapas.

Su ventaja es que contempla toda la funcionalidad del sistema desde un principio.

Su desventaja es que los cambios posteriores o inclusión de nuevos requerimientos son muy costosos porque impactan todas las fases llevadas a cabo con anterioridad.

Por otra parte toma tiempo importante del equipo la formalización o documentación de cada actividad una vez se finaliza.

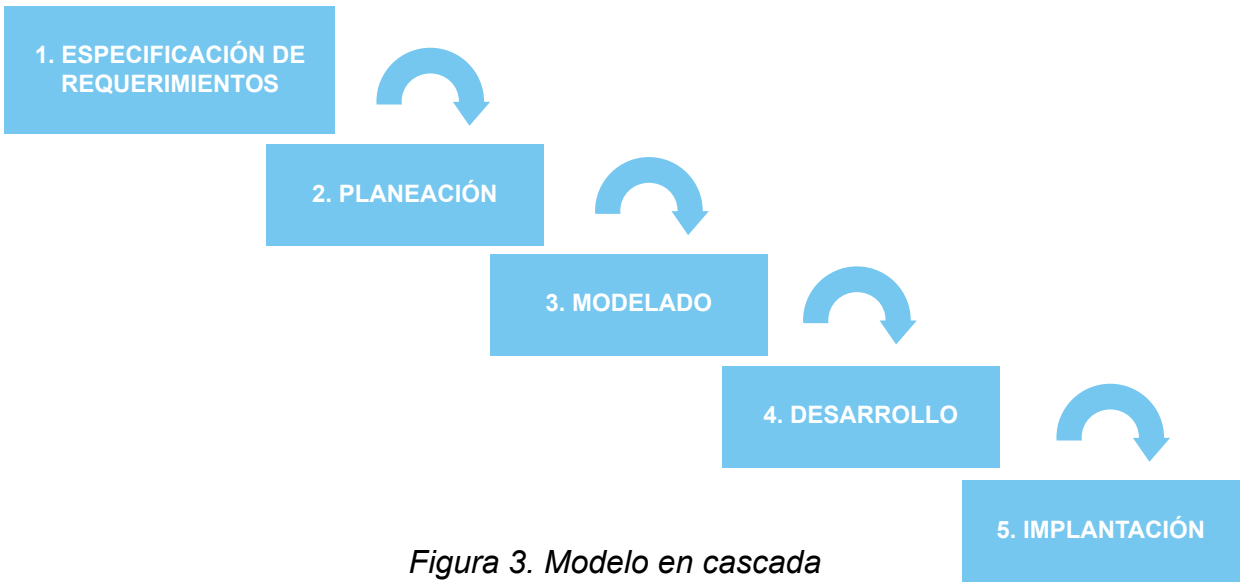


Figura 3. Modelo en cascada

3.2. Modelos incrementales

Este modelo está pensado para cuando se requiere entregar un producto de software con una funcionalidad reducida que posteriormente se va incrementando hasta llegar finalmente a la versión deseada del producto. Los requerimientos nuevos que se van agregando en cada entrega se denominan incrementos.

Un ejemplo de este tipo es el modelo en espiral (PRESSMAN, 2010) **(Figura 4)**.

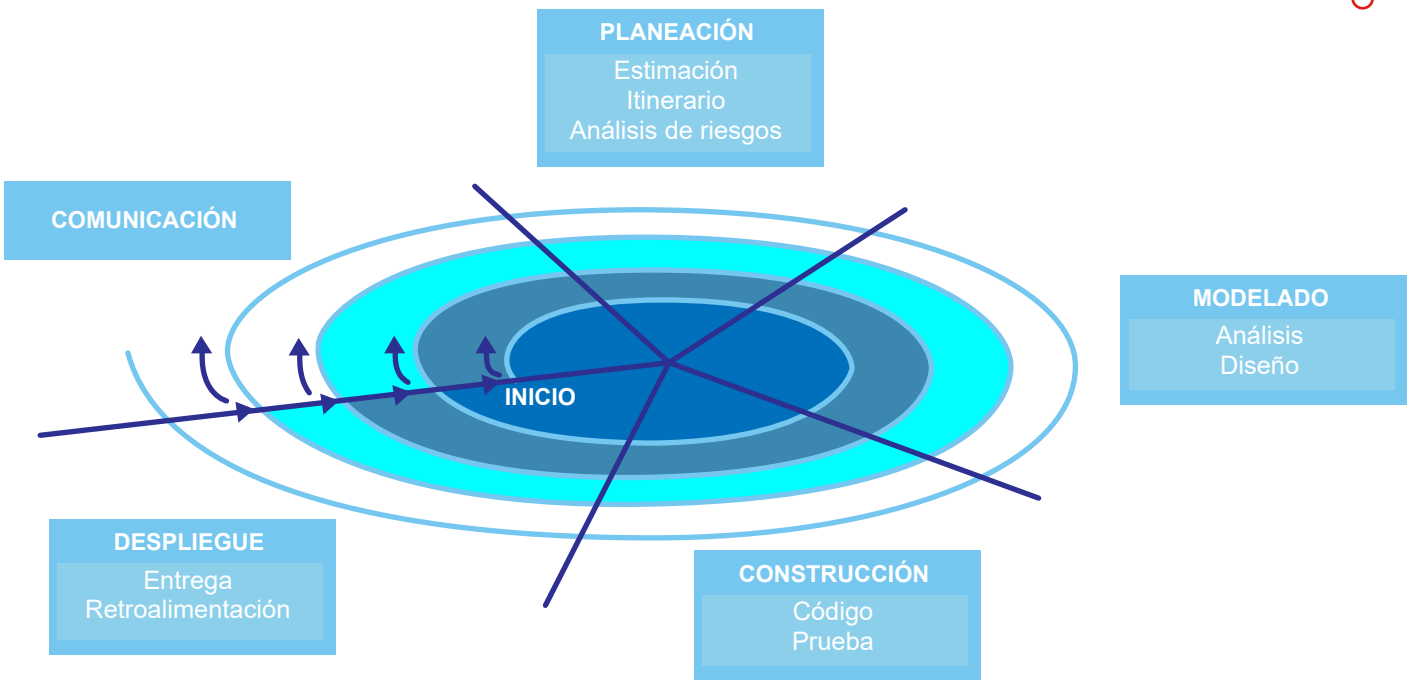


Figura 4. Modelo en espiral Fuente: Pressman (2010)

La ventaja es que se cuenta con un producto básico pero funcional en un tiempo menor comparado con el modelo en cascada.

La desventaja es que algunos clientes temen que este proceso evolutivo de entregas se salga de control. (PRESSMAN, 2010).

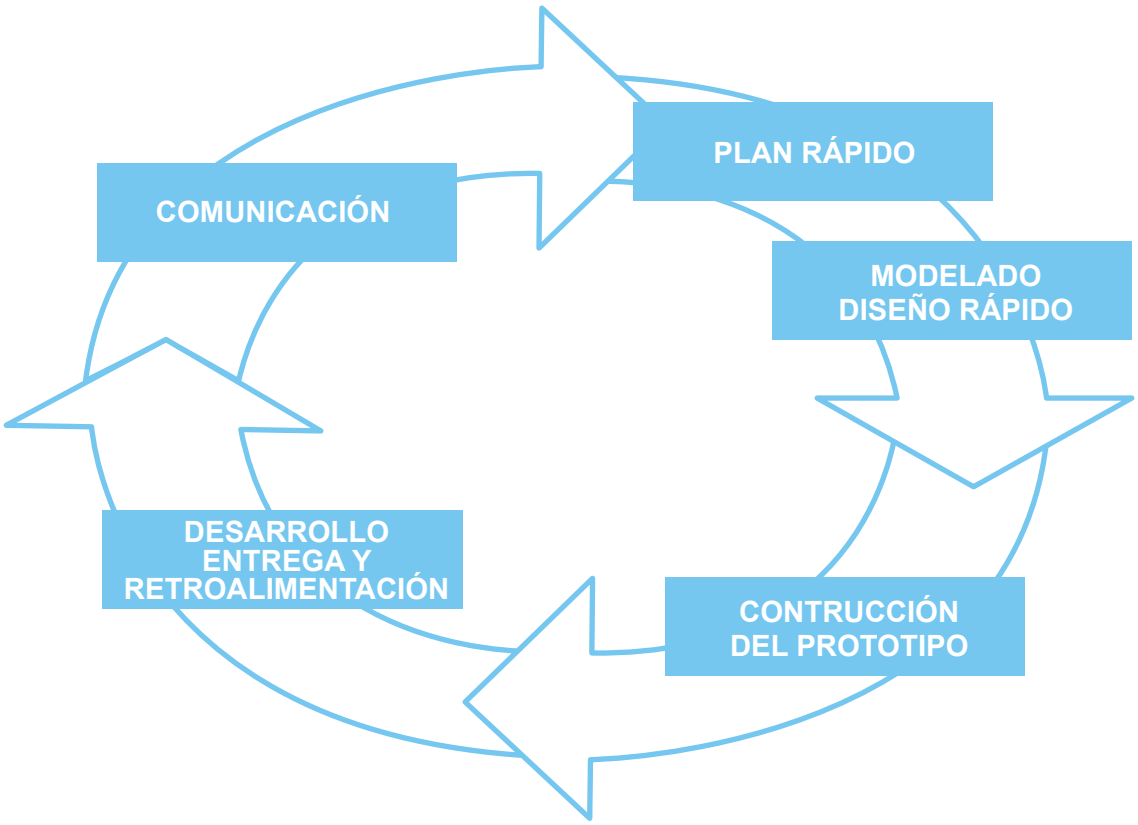
3.3. Modelo de evolución de prototipos

Este modelo se apoya en los llamados prototipos o versiones primarias del producto de software al cual se quiere llegar. Estos prototipos evolucionan o se mejoran cada vez que se aplican las actividades del proceso de software: especificación, planeación, modelado, desarrollo, implantación. Lo anterior es llamado una iteración (Figura 5).

La diferencia con el modelo incremental es que un prototipo se puede descartar por completo e iniciar con uno nuevo en cualquier etapa del proceso evolutivo o iterativo.

Su ventaja es que desde el primer prototipo ya se puede tener la realimentación del cliente y se pueden corregir en etapas muy tempranas cualquier falencia que se presente.

Su desventaja es que no se puede estimar fácilmente el tiempo de finalización del proyecto.



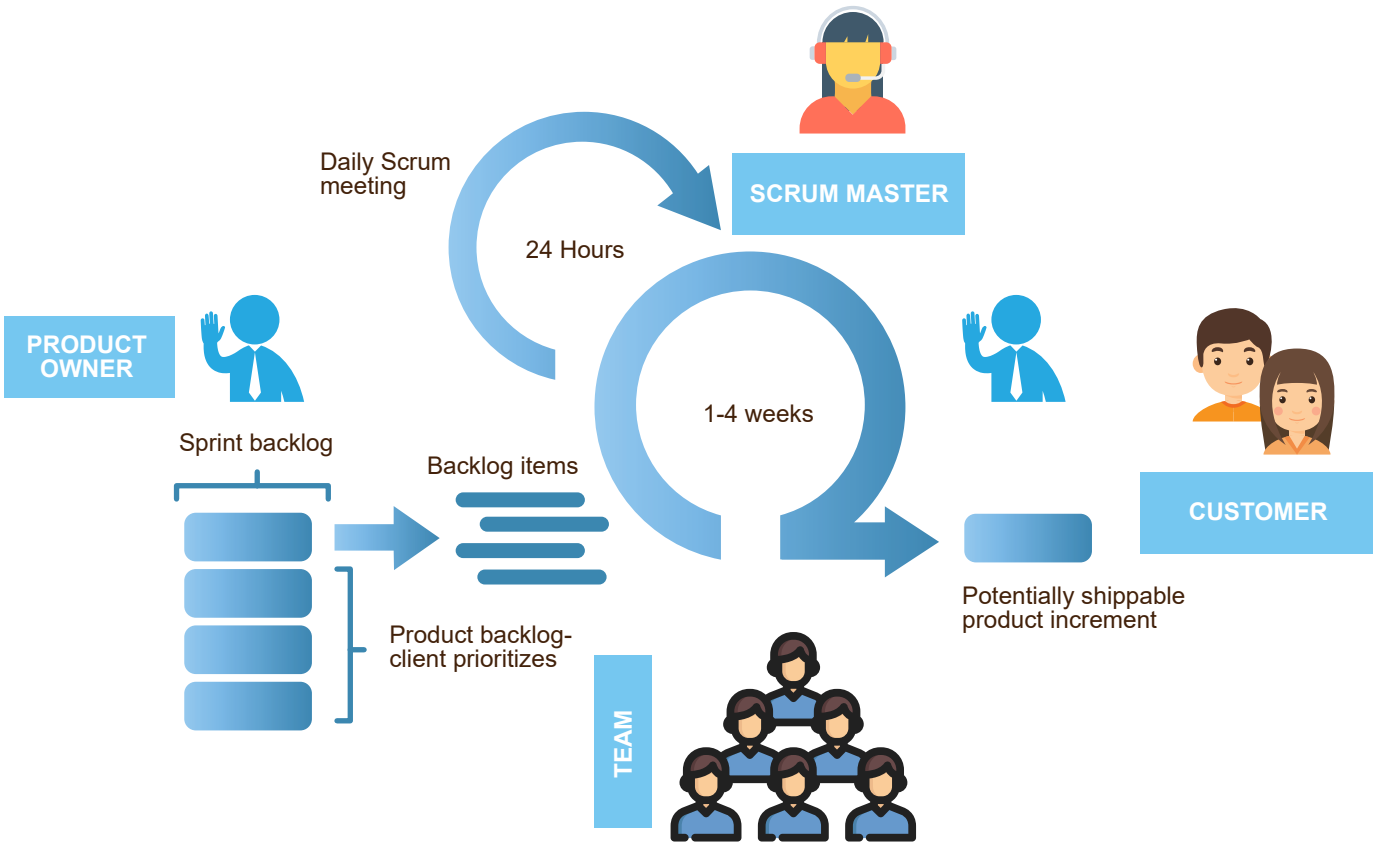
3.4. Modelos ágiles

Recientemente se han popularizado los modelos ágiles que combinan las estrategias de los modelos en cascada, incremental y prototipos.

Un ejemplo es el modelo SCRUM que está basado en el modelo incremental pero plantea, entre otros, unas técnicas o criterios para determinar cuando pasar a una siguiente versión del producto de software.

Estos modelos son apropiados cuando se anticipa que van a surgir cambios en los requerimientos a lo largo del proyecto y pretenden disminuir este impacto por un lado haciendo más liviano o ligero la formalización de las etapas del proceso del software y por otro haciendo énfasis en equipos de trabajo bastante cohesionados y motivados. (PRESSMAN, 2010).

La desventaja es que estas metodologías requieren un entrenamiento y disciplina en los equipos de trabajo que no es accesible para todas las empresas.



3.5 Modelo basado en componentes.

Este modelo se apoya en software previamente desarrollado que se puede incorporar, parametrizar o configurar al proyecto en desarrollo y de este forma disminuir la cantidad de código a producir.

Algunos softwares modernos como SAP, ORACLE, SIESA, entre otros, suministran componentes para una amplia gama de procesos de negocio (contabilidad, nómina, inventarios, entre otros) los cuales se configuran de acuerdo a las necesidades específicas del cliente. Lo anterior evita que la actividad de desarrollo se haga desde cero y los desarrollos se realizan para llenar las brechas o "gaps" entre la funcionalidad que requiere el cliente y lo ofrecido por el proveedor.

Los componentes también pueden ser proporcionados a través de "frameworks" o librerías que también son código ya elaborado y que puede ser configurado, utilizado y en algunos casos modificado en esta fase de desarrollo. Ejemplo de estos frameworks son J2EE, Microsoft .Net, Laravel (PHP), AngularJS, entre otros.

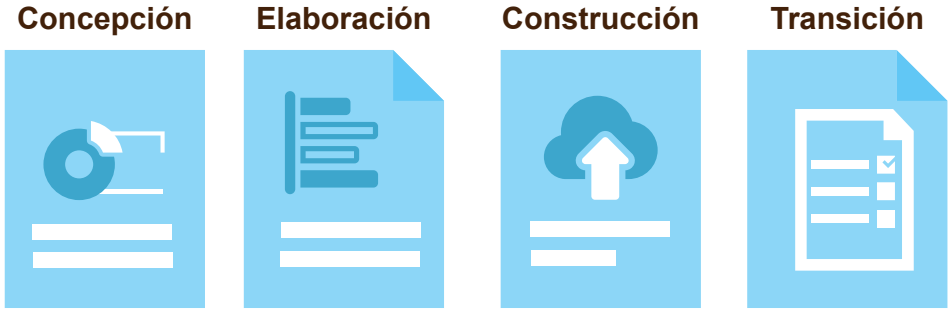
Una desventaja es que no siempre el componente o módulo cumple con todos los requerimientos del cliente y en algunos casos las empresas limitan o disminuyen los requerimientos en favor de una mayor prontitud en la entrega del producto de software especificado.

3.6. El proceso unificado.

Este proceso se basa en los modelos tradicionales ya expuestos como son cascada, prototipos e incrementales y hace énfasis en la utilización de casos de uso apoyados en el lenguaje UML para modelar los requerimientos del cliente (PRESMAN, 2010).

Está compuesto por cuatro fases: Concepción, elaboración, construcción y transición.

Al igual que los modelos ágiles el proceso unificado también afronta el desafío del cambio constante de los requerimientos y requiere de un entrenamiento al equipo de proyecto que no todas las empresas pueden acceder.



3.7. Comentarios sobre los modelos de procesos de software.

No todos los modelos son aplicables a cualquier tipo de proyecto. El ingeniero de software deberá tener en cuenta variables como: tamaño del proyecto, claridad en los requerimientos, disponibilidad y entrenamiento del recurso humano, entre otros, para proponer el modelo o combinación de modelos que le permitan llegar al producto de software especificado.

En las empresas colombianas se presenta a menudo el modelo basado en componentes, siendo estos componentes los softwares preconfigurados suministrados por empresas como SAP, ORACLE, SIESA o COMPUTERWARE (FEDESOFT, 2015).

Estos componentes son suministrados en un estado inicial, o línea base, que posteriormente es llevado a un estado deseado a través de actividades de configuración, desarrollo e integración con otros sistemas.

Glosario

Algoritmo: conjunto de pasos que se llevan a cabo para dar solución a un problema.

Case: computer Aided Software Engineering. Creación de software asistido por computador.

Componente: producto o módulo de software parametrizable que puede ser usado para construir otros programas más complejos.

Framework: conjunto de componentes o módulos de software que apoyan una o más etapas del proceso de desarrollo de software.

IDE: Integrated Development Environment. Entorno de desarrollo integrado. Conjunto de herramientas que apoyan el ciclo de desarollo de software.

IEEE: Institute of Electrical and Electronic Engineers. Asociación de ingenieros electrónicos y electricos de los Estados Unidos de América.

PMI: Project Management Institute. Asociación que agrupa a los practicantes de la gerencia de proyectos basados en el PMBOK (r).

POS: Point Of Sale. Sistema de punto de venta.

STAKEHOLDERS: interesados o participantes en un proyecto.

Bibliografía

NATO (1968). *Introducción a la ingeniería del software.* Recuperado de http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF

Pressman, R. (2010). *Ingeniería del software, un enfoque práctico* (Séptima edición). Bogotá: McGraw-Hill.

P. Bourque y R.E. Fairley, (eds) (2014), Guide to the Software Engineering Body of Knowledge, Versión 3.0, IEEE Computer Society.

Sommerville, I, (2005). *Ingeniería del software.* (Séptima Edición) Madrid: Pearson. Bourque.

Control del documento

CONSTRUCCIÓN
OBJETO DE
APRENDIZAJE

SENA

EL **PROCESO DE SOFTWARE**	
Centro Industrial de Mantenimiento Integral - CIMI Regional Santander	
Líder línea de producción:	Santiago Lozada Garcés
Asesores pedagógicos:	Rosa Elvia Quintero Guasca Claudia Milena Hernández Naranjo
Líder expertos temáticos:	Nelson Mauricio Silva M. (V1)
Experto temático:	Edgar Eduardo Vega Arango
Diseño multimedia:	Catalina Martínez Ávila
Programador:	Francisco José Lizcano Reyes
Producción de audio:	Víctor Hugo Tabares Carreño

® Registered trademark