Software

¿Qué es?

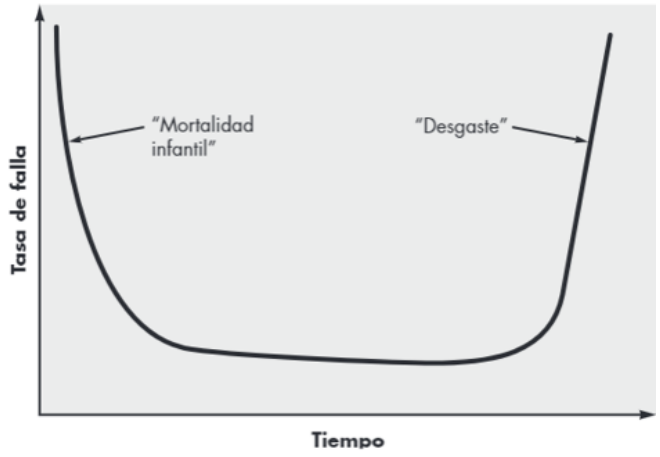
Instrucciones (programas de cómputo), procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación *(IEEE)*.

Características del software

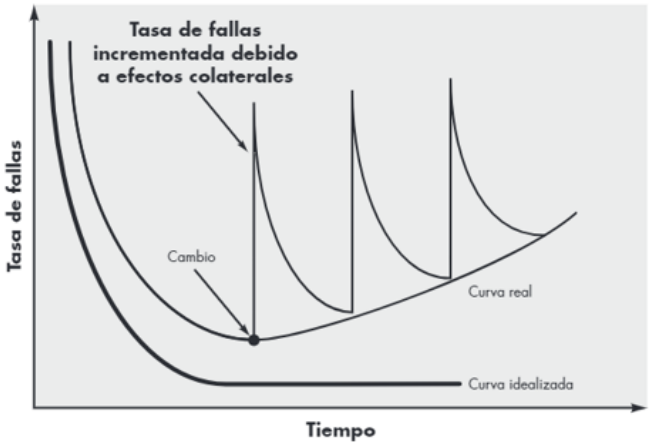
El software empezó a desarrollarse como se desarrollaba el hardware; pero el hardware es “duro”, una vez construido no se puede modificar, mientras que el software es “blando”, haciendo que muchas de las metodologías de desarrollo aplicadas al hardware fracasen al aplicarse en el software. Además, a diferencia del hardware, el software no se desgasta por el uso, y se desarrolla, no se fabrica.

Los problemas en el software no están vinculados con el tiempo de operación (como ocurre con el hardware), sino en los cambios que se introducen.

Envejecimiento del hardware



Envejecimiento del software



Tipos de producto de software

Cada vez más sistemas son construidos por un producto genérico como base que luego es adaptado a los requerimientos del cliente.

- **Genéricos:** sistemas aislados producidos por organizaciones desarrolladoras de software y que se venden en un mercado abierto. No están desarrollados para un usuario o uso específico.
- **Personalizados:** Sistemas requeridos por un cliente en particular.

Software libre

El término software libre se refiere a libertad, tal como fue concebido por Richard Stallman en su definición. En concreto, se refiere a cuatro libertades:

1. Libertad para **ejecutar** el programa en cualquier sitio, con cualquier propósito y para siempre.
2. Libertad para **estudiarlo** y adaptarlo a nuestras necesidades. Esto exige el acceso al código fuente.
3. Libertad de **redistribución**, de modo que se nos permita colaborar con vecinos y amigos.
4. Libertad para **mejorar** el programa y publicar las mejoras. También exige el código fuente.

Clasificación

- **De sistemas:** sirve a otros programas.
- **De aplicación:** resuelve las necesidades de un cliente. Lo usa un usuario final.
- **Científicos y de ingeniería:** software específico para usos en estos ámbitos.
- **Empotrados:** software específico para alguna máquina y que tiene restricciones de tamaño, velocidad de procesamiento, etc.
- **De línea de productos:** similar al software genérico que después se puede personalizar para ciertos clientes.
- **Integrados**
- **Aplicaciones web/móviles**
- **De inteligencia artificial**

Retos

Software heredado: es un software, normalmente de baja calidad, que no ha sido preparado para cambios (lo que genera muchos errores). Usualmente, cuenta con poca o nula documentación, lo que lo hace difícil de rehacer.

A medida que los usuarios adoptan nuevas tecnologías, parte del trabajo consiste en integrar los sistemas tradicionales con los nuevos para asegurar un contexto útil.

Ingeniería de software

¿Qué es la ingeniería de software?

Disciplina de la ingeniería que comprende todos los aspectos de la producción de software, desde las etapas iniciales de la especificación del sistema e incluyendo la evolución de este, luego que se comienza a ejecutar. No solo comprende los procesos técnicos del desarrollo de software, sino también se realizan actividades como la gestión de proyectos y el desarrollo de herramientas, métodos y teorías de apoyo a la producción de software.

El IEEE define a la ingeniería de software como:

1. El uso de métodos sistemáticos, disciplinados y cuantificables para el desarrollo, operación y mantenimiento de software.
2. El estudio de técnicas relacionadas con 1.

Usa métodos sistemáticos cuantificables, dentro de tiempos y costos estimados, para el desarrollo, operación y mantenimiento.

Un ingeniero de software debe saber lo que el usuario quiere, cómo lo quiere, cuándo y por qué. La **comunicación** es la base para la obtención de las necesidades del cliente, un *error* en los requerimientos se arrastra a lo largo de todo el desarrollo del software. Al hablar de necesidades, en términos más técnicos, estamos hablando de **requerimientos.**

Requerimientos

Un **requerimiento** es una característica del sistema o una descripción de algo que el sistema debe ser capaz de hacer con el objetivo de satisfacer su propósito.

Definición IEEE-STD-610

1. Condición o capacidad que necesita el usuario para resolver un problema o alcanzar un objetivo.
2. Condición o capacidad que debe satisfacer o poseer un sistema o una componente de un sistema para satisfacer un contrato, un estándar, una especificación u otro documento formalmente impuesto.
3. Representación documentada de una condición o capacidad como en 1 o 2.

Tipos de requerimientos

Requerimientos funcionales

- Describen una interacción entre el sistema y su ambiente. Cómo debe comportarse el sistema ante determinado estímulo.
- Describen qué hace y qué no hace el sistema. Esto último sirve para acotar las funcionalidades y el alcance del sistema.
- Son independientes de la implementación de la solución.

Requerimientos no funcionales

- Describen una **restricción** sobre el sistema que limita nuestras elecciones en la construcción de una solución al problema.
- Son limitaciones en relación con la construcción del sistema. Por ejemplo: seguridad, tecnologías específicas, accesibilidad, portabilidad, etc.
- Generan calidad al sistema.
- No modifican a los requerimientos funcionales.

Tipos de requerimientos no funcionales

- **Requerimientos del producto:** especifican el comportamiento del producto (usabilidad, eficiencia, rendimiento, espacio, fiabilidad, portabilidad, etc.).
- **Requerimientos organizacionales:** se derivan de las políticas y procedimientos existentes en la organización del cliente y en la del desarrollador (entrega, implementación, estándares, etc.).
- **Requerimientos externos:** interoperabilidad, legales, privacidad, seguridad, éticos, etc.).

Ingeniería de requerimientos

Es el proceso por el cual se transforman los requerimientos a especificaciones precisas, no ambiguas, consistentes y completas del comportamiento del sistema que servirán como base para acuerdos. Esto queda en un documento de especificación de requerimientos que incluye el punto de vista del programador y del cliente.

El documento de **especificación de requerimientos** de software IEEE STD-830-1998 (**SRS**) tiene como objetivo brindar una colección de buenas prácticas para escribir especificaciones de software donde se describen los contenidos y las cualidades de una buena especificación de requerimientos.

La ingeniería de requerimientos es, también, el proceso mediante el cual se intercambian diferentes puntos de vista para recopilar y modelar lo que el sistema va a realizar.

Especificación de requerimientos

Propiedades de los requerimientos

- **Necesario:** su omisión provoca una deficiencia.
- **Conciso:** Fácil de leer y entender
- **Completo:** No necesita ampliarse
- **Consistente:** No contradictorio con otro
- **No ambiguo:** Tiene una sola implementación
- **Verificable:** Puede testearse a través de inspecciones, pruebas, etc.

Objetivos

- **Permitir** que los desarrolladores expliquen cómo han entendido lo que el cliente pretende del sistema.
- **Indicar** a los diseñadores qué funcionalidad y características va a tener el sistema resultante
- **Indicar** al equipo de pruebas qué demostraciones llevar a cabo para convencer al cliente de que el sistema que se le entrega es lo que había pedido.

Fuentes de requerimientos

Stakeholder

Es cualquier persona o grupo que se verá afectado por el sistema, directa o indirectamente. Como: usuarios finales, ingenieros, gerentes, expertos del dominio.

Puntos de vista

Existen tres tipos genéricos de puntos de vista:

- **Interactuadores:** personas y/o sistemas que interactúan directamente con el sistema.
- **Indirecto:** stakeholders que no utilizan el sistema pero que influyen en sus requerimientos.
- **Dominio:** de que trata el software, incluye las características y restricciones del dominio. Entre otras cosas, incluye a las regulaciones (leyes) y estándares a aplicar.

Elicitación de requerimientos

Es el proceso de adquirir todo el conocimiento relevante necesario para producir un modelo de los requerimientos de un dominio del problema. La elicitación de requerimientos es más una actividad de carácter social que tecnológico.

Objetivos:

- Conocer el dominio del problema para poder comunicarse con clientes y usuarios y entender sus necesidades.
- Conocer el sistema actual (manual o informatizado).
- Identificar las necesidades, tanto explícitas como implícitas, de clientes y usuarios, y sus expectativas sobre el sistema a desarrollar.

Técnicas de elicitación de requerimientos

Métodos discretos

Los métodos discretos son menos perturbadores que otras formas de averiguar los requerimientos, pero se consideran insuficientes para recopilar información. Sin embargo, son una buena forma de formar un panorama más completo de los requerimientos.

1. **Muestreo de la documentación, los formularios y los datos existentes.**
   Recolección de hechos a partir de la documentación existente. Permiten conocer el historial que origina el proyecto.

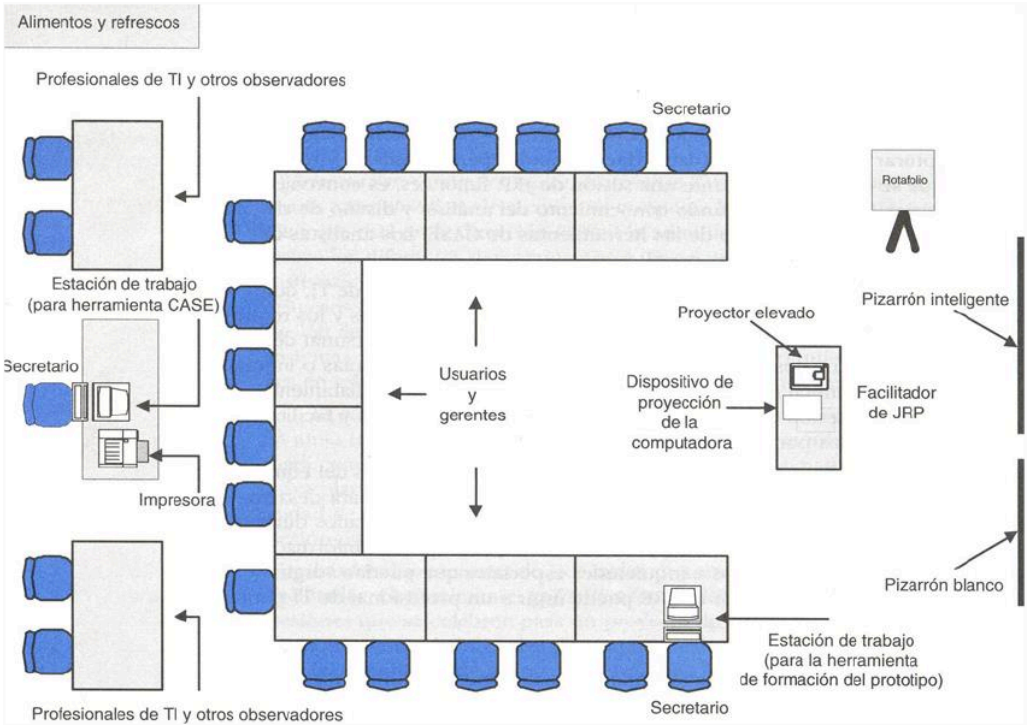
2. **Investigación y visitas al lugar.**
   Investigar el dominio, consultar a otras organizaciones con problemas similares, revistas especializadas, buscar problemas similares en internet, etc.

3. **Observación del ambiente de trabajo.**
   El analista observa a las personas y actividades del sistema con el objeto de aprender acerca del sistema. Permite obtener datos confiables al observar exactamente lo que se hace.

Métodos interactivos

Hay métodos interactivos que pueden usarse para obtener los requerimientos de los miembros de la organización. La base es hablar con las personas en la organización y escuchar para comprender.

1. **Planeación conjunta de requerimientos (JRP)**
   Proceso mediante el cual se conducen reuniones de grupo altamente estructuradas con el propósito de analizar problemas y definir requerimientos.



Patrocinador: es el responsable del proyecto, toma las decisiones finales.
Equipos de TI: escuchan y toman notas de los requerimientos.
Usuarios y gerentes: Los *usuarios* comunican los requerimientos y los *gerentes* los aprueban.
Secretarios: llevan el registro de la sesión y van publicando los resultados realizados.
Facilitador: dirige las sesiones y tiene amplias habilidades de comunicación y negociación.

2. **Brainstorming (lluvia de ideas)**

   Se alienta a los participantes para que ofrezcan tantas ideas como sea posible en un corto tiempo, sin ningún análisis hasta que se hayan agotado las ideas.

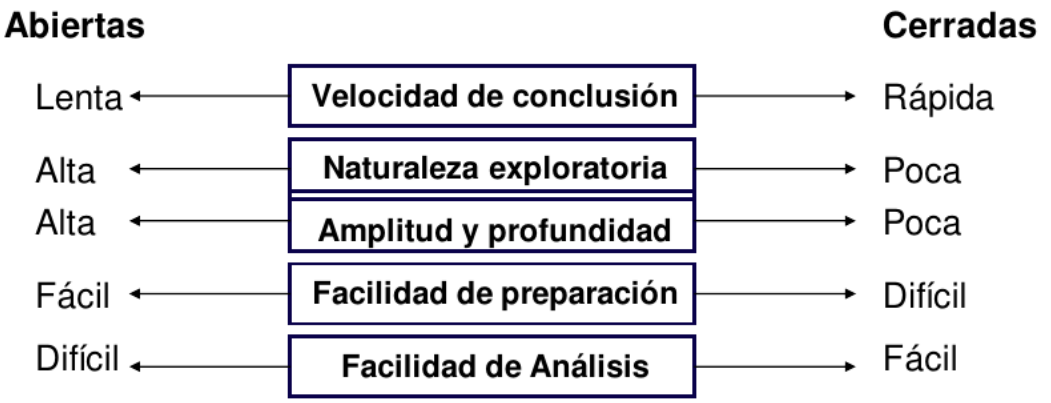
   Cuantas más ideas se sugieren, mejores resultados se conseguirán. La producción de ideas en grupos puede ser más efectiva que la individual. Las ideas de una persona pueden hacer que aparezcan otras por "contagio. A veces las mejores ideas aparecen tarde. Es mejor elegir sobre una variedad de soluciones.

3. **Cuestionarios**

   Es un documento que permite al analista recabar información y opiniones de los encuestados.

   Tipos de preguntas:

   - **Abiertas:** son las que dejan abiertas todas las opciones de respuesta.
   - **Cerradas:** limitan o cierran las opciones de respuestas disponibles



Tipo de información obtenida:

- **Actitud:** lo que las personas dicen que quieren.
- **Creencias:** lo que las personas creen que es verdad.
- **Comportamiento:** lo que realmente hacen.
- **Características:** de las personas o cosas.

4. **Entrevistas**

Es una conversación con un propósito específico, que se basa en u formato de preguntas y respuestas en general. Permite conocer opiniones y sentimientos del entrevistado

Tipos de entrevistas:

- **Estructuradas (cerradas):** conjunto de preguntas específico sobre un requerimiento puntual. No permite adquirir un amplio conocimiento del dominio.
- **No estructuradas (abiertas):** el encuestador lleva a un tema en general. Sin preparación de preguntas específicas. Iniciar con preguntas que no dependen del contexto, para conocer el problema, la gente involucrada, etc.

Tipos de preguntas:

- **Abiertas:** permite al encuestado responder de cualquier manera.
- **Cerradas:** las respuestas son directas, cortas o de selección específica.
- **Sondeo:** permite obtener más detalle sobre un tema puntual.

Organización de una entrevista:

- **Piramidal (inductivo):** cerradas → abiertas.
- **Embudo (deductivo):** abiertas → cerradas.
- **Diamante:** cerradas → abiertas → cerradas.

Estudio de viabilidad

A partir de una descripción resumida del sistema se elabora un informe que recomienda la realización o no del proceso de desarrollo. Se realiza principalmente para sistemas nuevos.

Responde a las siguientes preguntas:

- ¿El sistema contribuye a los objetivos generales de la organización? (Si no contribuye, entonces no tiene un valor real en el negocio)
- ¿El sistema se puede implementar con la tecnología actual?
- ¿El sistema se puede implementar con las restricciones de costo y tiempo?
- ¿El sistema puede integrarse a otros que existen en la organización?

Una vez que se ha recopilado la información necesaria para contestar las preguntas anteriores, se redacta un informe donde debería hacerse una recomendación sobre si debe continuar o no el desarrollo del sistema.

Validación de requerimientos

Es el proceso de certificar la corrección del modelo de requerimientos contra las intenciones del usuario. Trata de mostrar que los requerimientos definidos son los que estipula el sistema, describiendo el ambiente en el que debe operar el sistema. Es importante porque los errores en los requerimientos pueden conducir a grandes costos si se descubren más tarde.

Según el IEEE:

- **Validación:** al final del desarrollo, evaluar el software para asegurar que cumple con los requerimientos.
- **Verificación:** el software cumple los requerimientos correctamente.

Técnicas de especificación de requerimientos

- **Estáticas:** se describe el sistema a través de *entidades* u *objetos*, sus atributos y sus relaciones con otros. No describen cómo las relaciones cambian con el tiempo.
- **Dinámicas:** se describe el sistema en *función de los cambios* que ocurren a lo largo del tiempo. Se considera que el sistema está en un estado particular hasta que un estímulo lo obliga a cambiar su estado.

HU - Historias de usuario

Una historia de usuario es una descripción corta y simple de un requerimiento de un sistema, que se escribe en lenguaje común del usuario y desde su perspectiva.

CU - Casos de uso

Modelado de las funcionalidades del sistema en término de los eventos que interactúan entre los usuarios y el sistema.

DTE - Diagramas de transición de estados

Máquinas de estado finito

Describe al sistema como un conjunto de estados donde el sistema reacciona a ciertos eventos posibles (eternos o internos).

`f(Si, Cj) =` Sk: al estar en el estado `Si`, la ocurrencia de la condición `Cj` hace que el sistema cambie al estado Sk.

5-tupla (S, Σ, T, s, A) donde:

- Σ es un alfabeto: todos los eventos o transiciones que pueden pasar.
- S es un conjunto de estados
- T es la función de transición
- s es el estado inicial
- A es un conjunto de estados de aceptación o finales.

RP - Redes de Petri

Utilizadas para especificar sistemas de tiempo real en los que son necesarios representar aspectos de concurrencia.

Tablas de decisión

Describe el sistema como un conjunto de: posibles **condiciones** satisfechas por el sistema en un momento dado, **reglas** para reaccionar ante los estímulos que ocurren cuando se reúnen determinado conjunto de condiciones y **acciones** a ser tomadas como un resultado.

Se construye una tabla con condiciones y acciones simples. Las condiciones tomas valores de verdadero o falso y hay 2^N reglas donde N es la cantidad de condiciones.

Análisis estructurado

La técnica de **análisis estructurado** hace énfasis en el procesamiento o la transformación de datos conforme estos pasan por distintos procesos.

Diagrama de flujo de datos (DFD)

Representa la transformación de entradas a salidas y es también llamado diagrama de burbujas. Permite visualizar un sistema como una red de procesos funcionales conectados entre sí por "conductos" y almacenamiento de datos.

Se utiliza un rectángulo para representar una ***entidad externa,*** esto es, un elemento del sistema (por ejemplo, un elemento hardware, una persona, otro programa) u otro sistema que produce información para ser transformada por el software, o recibe información producida por el software.

Un círculo (también llamado burbuja) representa un ***proceso*** o *transformación* que es aplicado a los datos (o al control) y los modifica.

Una flecha representa uno o más ***elementos de datos*** (objetos de dato).

Un rectángulo abierto (lado izquierdo y derecho) que representa un **almacén de datos**

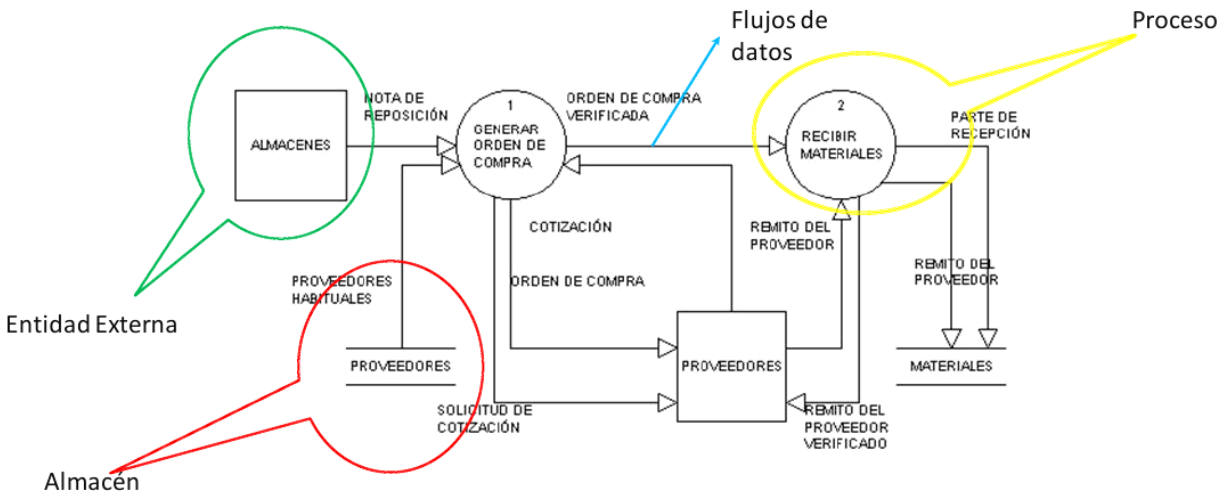
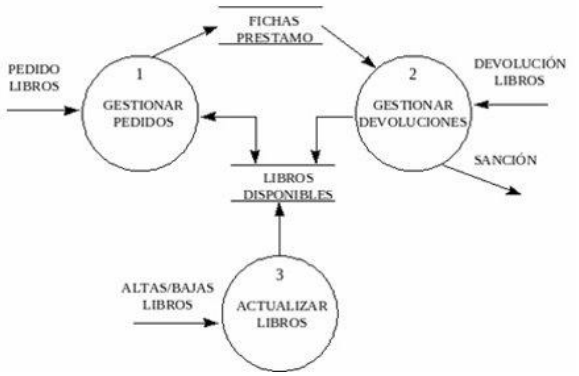


Diagrama de contextos

Se muestra un panorama global que muestre las entradas básicas y las salidas. Es el nivel más alto en un DFD y contiene un solo proceso que representa a todo el sistema.

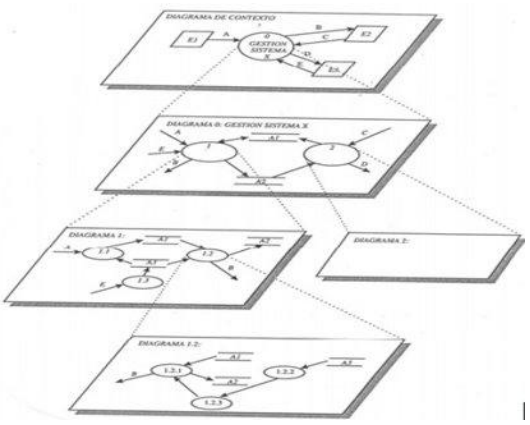
Nivel 0

Es la ampliación del diagrama de contexto. Las entradas y salidas permanecen, pero se amplía para incluir hasta 9 procesos (como máximo) y mostrar los almacenes de datos y nuevos flujos.



Nivelación de un DFD

Cada proceso se puede ampliar para crear un diagrama hijo más detallado. Las entradas y salidas del proceso padre permanecen y pueden aparecer nuevos almacenes de datos y nuevos flujos.



Modelos de proceso

Un **proceso** es un conjunto ordenado de tareas que nos marcan diferentes formas de desarrollo de software

¿Qué es un proceso de software?

Un **proceso de software** es un conjunto de actividades y resultados asociados que producen un producto de software. Está entre las necesidades y requisitos del cliente, y el producto final.

Actividades fundamentales de los procesos de software

- **Especificación:** consiste en el proceso de comprender y definir que servicios se requieren del sistema, así como la identificación de las restricciones sobre la operación y desarrollo del sistema. También llamada *ingeniería de requerimientos.*
- **Desarrollo:** corresponde al proceso de convertir una especificación en un sistema ejecutable. Incluye los procesos de diseño y programación del software.
- **Validación:** verificar que el sistema cumpla con sus especificaciones y con las expectativas del cliente.
- **Evolución:** como se lleva adelante el mantenimiento del software una vez puesto en producción.

¿Qué es un modelo de proceso de software?

Es una representación simplificada que nos presenta una visión de ese proceso de software que queremos llevar adelante. Estos modelos pueden incluir actividades que son partes de los procesos y productos de software, y el papel de las personas involucradas.

> *Marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software. Abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso.*
>
> *Norma ISO 12207-1*

Términos

- **Ciclo de vida:** proceso que implica la construcción de un producto.
- **Ciclo de vida del software:** describe la vida del producto de software desde su concepción hasta su implementación, entrega, utilización y mantenimiento.
- **Modelos de proceso de software:** representación abstracta de un proceso del software
    - Modelo de proceso, paradigma de software y ciclo de vida del software son términos equivalentes.

Modelos de procesos

- **Modelos prescriptivos:** prescriben un conjunto de elementos del proceso. Cada modelo de proceso prescribe también un flujo de trabajo, es decir, de qué forma los elementos del proceso se interrelacionan entre sí.
- **Modelos descriptivos:** describen la forma en que se realizan los procesos en la realidad, pero no hay una prescripción.

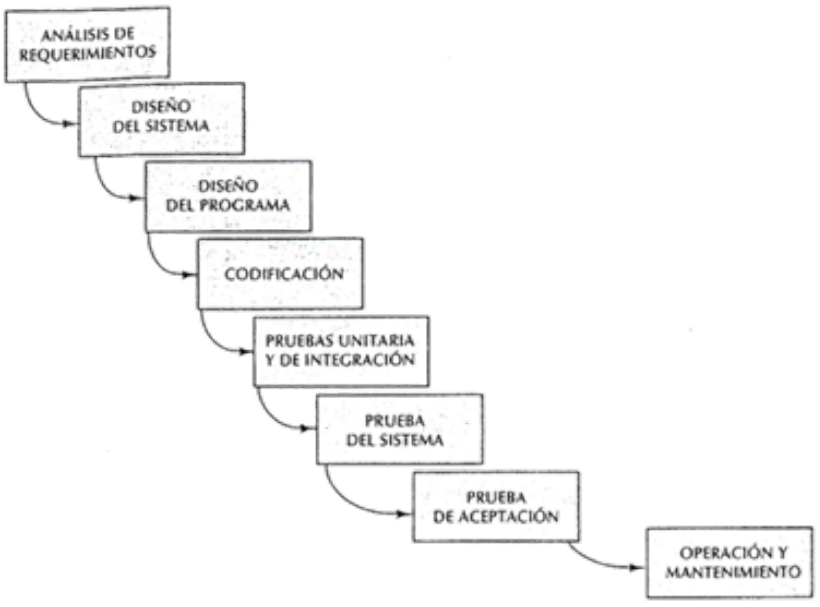
Ambos modelos deberían ser iguales (tanto lo que prescriben como lo que describen).

Tipos

- **Modelos tradicionales:** formados por un conjunto de fases o actividades en las que no tienen en cuenta la naturaleza evolutiva del software. Por ejemplo: en cascada, modelo en V, basado en prototipos.
- **Modelos evolutivos:** son los modelos que se adaptan a la evolución que sufren los requisitos del sistema en función del tiempo. Por ejemplo: en espiral, evolutivo, incremental.
- **Procesos ágiles:** scrum, xp, etc.

Modelo en cascada

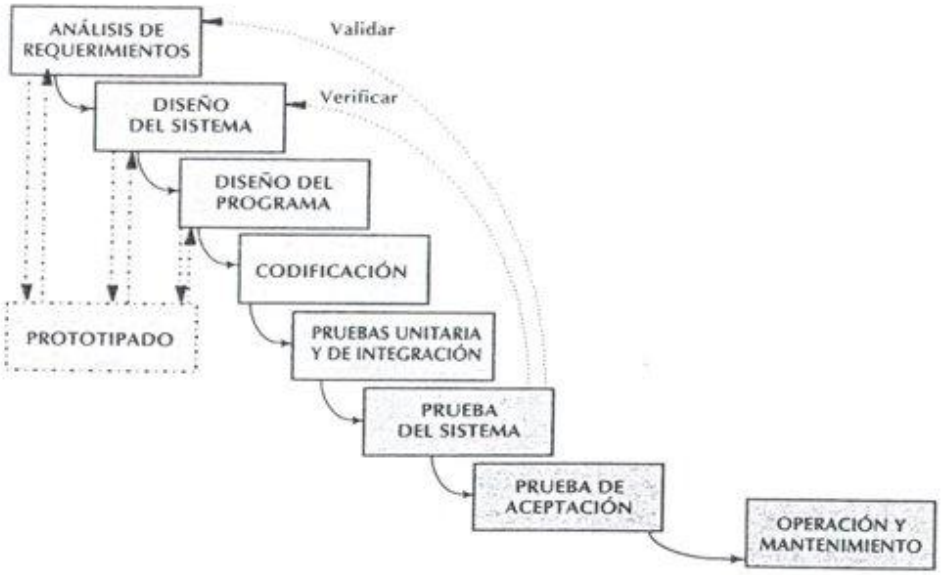
Las etapas se representan cayendo en cascada. Cada etapa de desarrollo se debe completar antes de que comience la siguiente.



El problema de este modelo es que no se adapta a la realidad del desarrollo de software. No responde bien a los cambios y a la detección de errores.

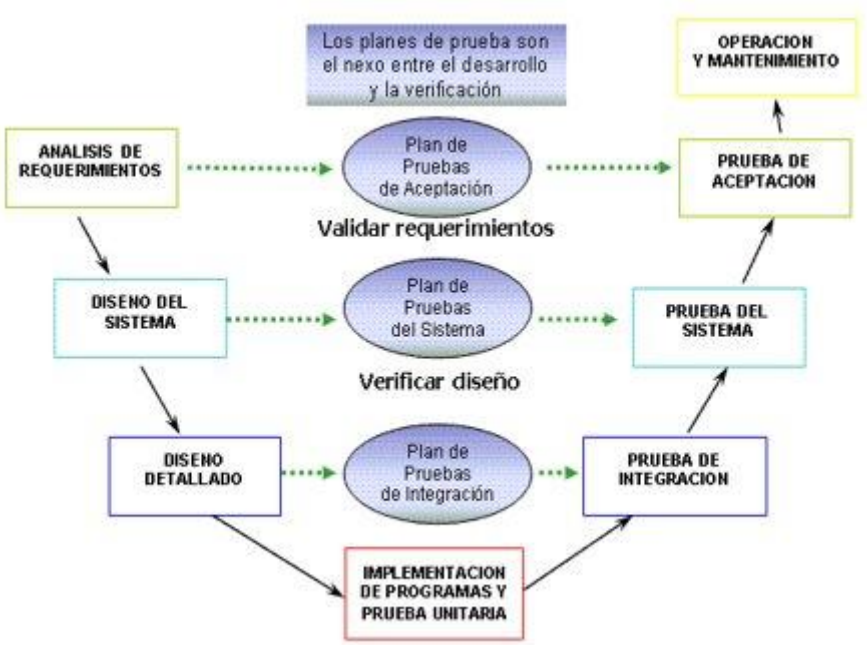
Modelo en cascada con prototipo

Se incorporan etapas tempranas de prototipado que nos permiten ir y volver entre los pasos, sobre todo en la etapa de especificación de requerimientos.



Modelo en V

Relaciona las actividades de prueba con las de análisis y diseño. La vinculación entre el lado derecho e izquierdo implica que, si se encuentran problemas durante la verificación y la validación, entonces el lado izquierdo puede ser ejecutado nuevamente para solucionar el problema.



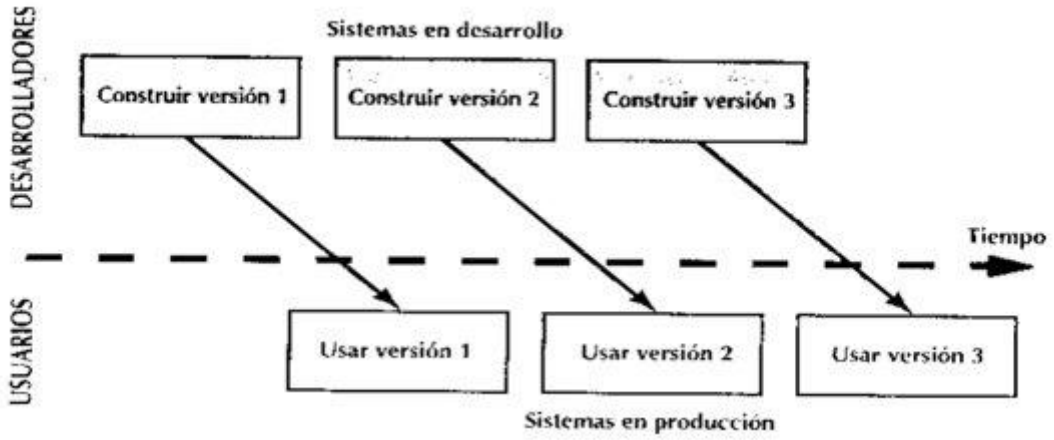
Modelo de prototipos

Un **prototipo** es un producto parcialmente desarrollado que permite a clientes y desarrolladores examinar algunos aspectos del sistema propuesto y decidan si este es adecuado para el producto terminado. Existen dos tipos:

	Descartable	Evolutivo
Descripción	No tiene funcionalidad. Se utilizan herramientas de modelado	Empieza a trabajarse en un desarrollo que terminará siendo el sistema a entregar
Enfoque de desarrollo	Rápido y sin rigor	Riguroso
Que construir	Solo las partes problemáticas	Primero las partes bien entendidas. Sobre una base sólida
Objetivo último	Desecharlo	Lograr el sistema

Modelo de desarrollo por fases

El sistema se desarrolla de manera tal que puede ser entregado en piezas. Esto implica que existen **dos sistemas funcionando** en paralelo: el sistema **operacional** y el sistema en **desarrollo**.



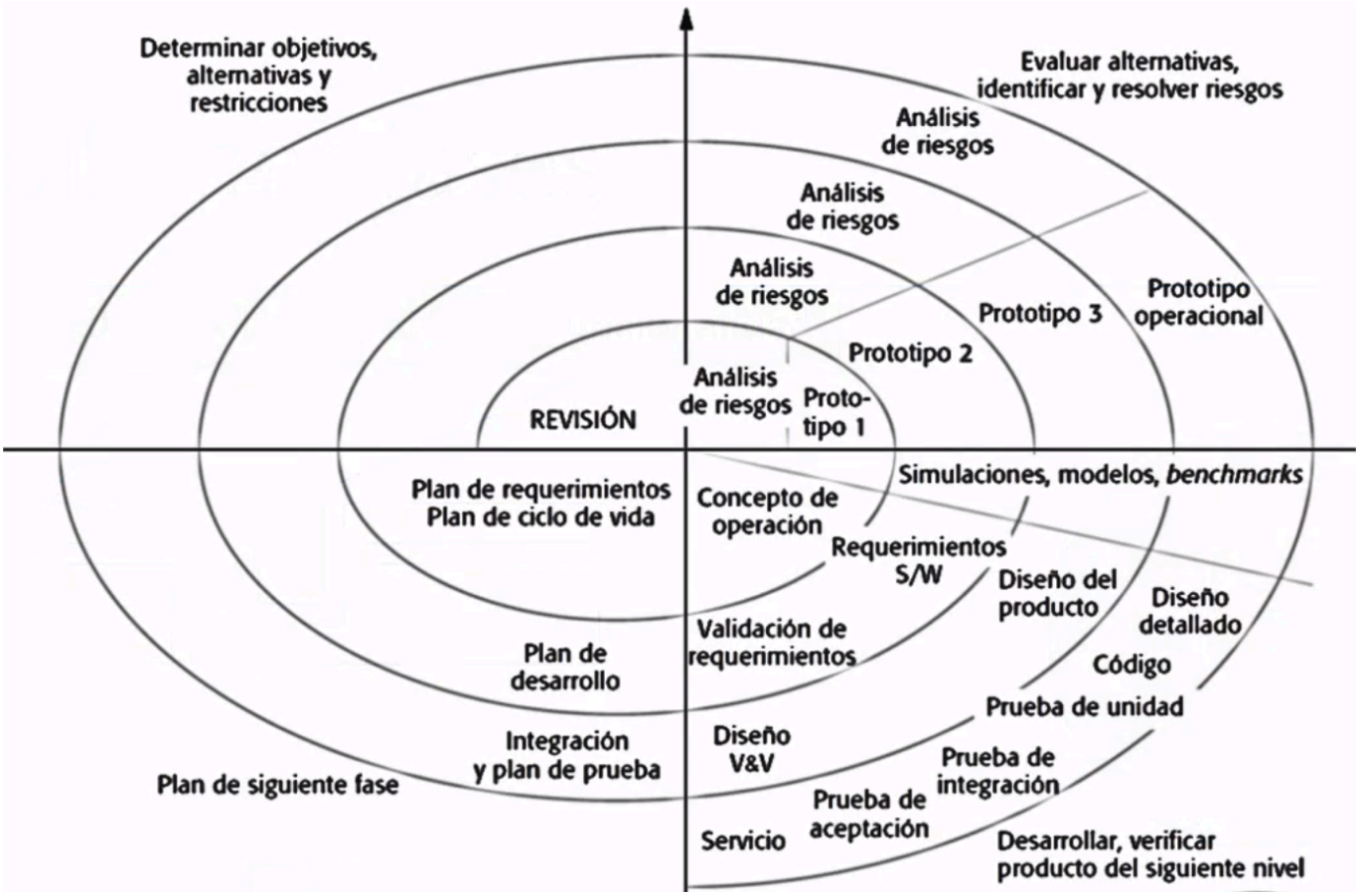
Puede ser de dos tipos:

- **Incremental:** el sistema es particionado en subsistemas de acuerdo con su funcionalidad, donde cada entrega agrega un subsistema.
- **Iterativo:** entrega un sistema completo desde el principio y luego aumenta la funcionalidad de cada subsistema con las nuevas versiones.

Modelo en espiral (Boehm)

Combina las actividades de desarrollo con la gestión del riesgo. Cada ciclo del espiral se divide en cuatro sectores:

1. **Establecimiento de objetivos:** se identifican restricciones, se traza un plan de gestión y se identifican riesgos.
2. **Valoración y reducción del riesgo:** se analiza cada riesgo identificado y se determinan acciones.
3. **Desarrollo y validación:** se determina el modelo de desarrollo.
4. **Planeación:** el proyecto se revisa y se toman decisiones para la siguiente fase.



Metodologías ágiles

El éxito de un desarrollo está dado por la metodología empleada. Generalmente, esta metodología lleva asociado un marcado énfasis en el control del proceso; definiendo roles, actividades, herramientas y documentación detallada.

Este enfoque no resulta muy adecuado para proyectos actuales, donde el entorno del sistema es muy cambiante y se exige una reducción de tiempo. En este contexto, las **metodologías ágiles** emergen como una posible solución.

Las metodologías ágiles tienen un enfoque *iterativo* e *incremental* del desarrollo de software.

- **Iterativo:** estrategia de reproceso en la que el tiempo se separa para revisar y mejorar partes del sistema.
- **Incremental:** es una estrategia programada y en etapas, en la que las diferentes partes de los sistemas desarrollan en diferentes momentos o velocidades, y se integran a medida que se completan.

Una **metodología ágil** es aquella en las que se da prioridad a las tareas que dan resultados directos y que reducen la burocracia tanto como sea posible.

Valores

- **Individuos e interacciones** más que procesos y herramientas.
- **Software operantes** más que documentaciones completas.
- **Colaboración con el cliente** más que negociaciones contractuales.
- **Respuesta al cambio** más que apegarse a una rigurosa planificación.

Principios

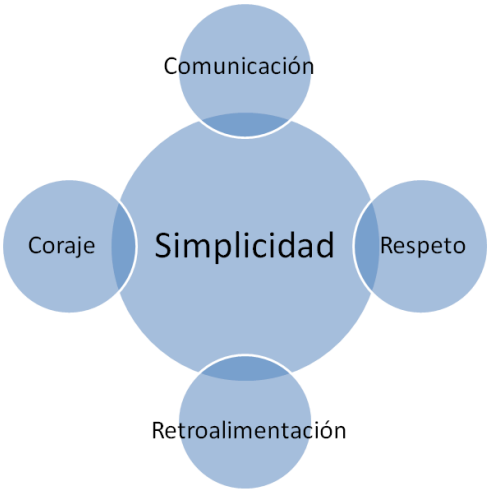
1. Nuestra mayor prioridad es satisfacer al cliente a través de fáciles y continuas entregas de software valuable.
2. Los cambios de requerimientos son bienvenidos, aún tardíos, en el desarrollo. Los procesos ágiles capturan los cambios para que el cliente obtenga ventajas competitivas.
3. Entregas frecuentes de software, desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre una entrega y la siguiente.
4. Usuarios y desarrolladores deben trabajar juntos durante todo el proyecto.
5. Construir proyectos alrededor de motivaciones individuales.
6. Darles el ambiente y el soporte que ellos necesitan y confiar el trabajo dado. El diálogo cara a cara es el método más eficiente y efectivo de intercambiar información entre el equipo de desarrolladores.
7. El software que funciona es la medida clave de progreso.

8. Los procesos ágiles promueven un desarrollo sostenible. Los stakeholders, desarrolladores y usuarios deberían ser capaces de mantener un paso constante indefinidamente.
9. Atención continua a la excelencia técnica y buen diseño incrementa la agilidad.
10. Simplicidad (el arte de maximizar la cantidad de trabajo no dado) es esencial.
11. Las mejores arquitecturas, requerimientos y diseños surgen de la propia organización de los equipos.
12. A intervalos regulares, el equipo reflexiona sobre cómo volverse más efectivo, entonces afina y ajusta su comportamiento en consecuencia.

Principales metodologías ágiles

XP - Extreme Programming

Se basa en los valores de sencillez, comunicación, retroalimentación, coraje y respeto.



Su acción consiste en llevar al equipo a prácticas simples, con suficiente información para ver dónde están y para ajustar las prácticas a su situación particular.

Las características esenciales:

- Historias de usuarios
- Roles
- Proceso
- Prácticas

Roles: programador *(programmer)*, jefe de proyecto *(manager)*, cliente *(customer)*, entrenador *(coach)*, encargado de pruebas *(tester)*, rastreador *(tracker)*.

Proceso

1. **Exploración**
    a. Los clientes plantean las historias de usuario que son de interés para la primera entrega del producto.
    b. El equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto.
    c. Se construye un prototipo.
2. **Planificación**
    a. El cliente establece la prioridad de cada historia de usuario.
    b. Los programadores realizan una estimación del esfuerzo.
    c. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente.

3. **Iteraciones**
    a. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas.
    b. El cliente es quien decide qué historias se implementarán en cada iteración.
    c. Al final de la última iteración el sistema estará listo para entrar en producción.
4. **Producción**
    a. Esta fase requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente.
    b. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.
5. **Mantenimiento**
    a. Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones.
    b. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.
6. **Muerte**
    a. Es cuando el cliente no tiene más historias para ser incluidas en el sistema.
    b. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura.
    c. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

Prácticas

- Testing
- Refactoring
- Programación de a pares
- Propiedad colectiva del código
- Integración continua

    Cada pieza de código es integrada en el sistema una vez que esté lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día. Reduce la fragmentación de los esfuerzos de los desarrolladores por falta de comunicación sobre lo que puede ser reutilizado o compartido.
- Semana de 40 horas
- Cliente en el lugar de desarrollo
- Estándares de codificación

Scrum

En Scrum se realizan entregas parciales y regulares del proyecto, priorizadas por el beneficio que aportan al receptor del mismo.

Principios

- **Eliminar el desperdicio:** no generar artefactos, ni perder el tiempo haciendo cosas que no le suman valor al cliente.
- **Construir la calidad con el producto:** la idea es inyectar la calidad directamente en el código desde el inicio.
- **Crear conocimiento:** En la práctica no se puede tener el conocimiento antes de empezar el desarrollo.
- **Diferir las decisiones:** tomar las decisiones en el momento adecuado, esperar hasta ese momento, ya que uno tiene más información a medida que va pasando el tiempo. Si se puede esperar, mejor.
- **Entregar rápido:** Debe ser una de las ventajas competitivas más importantes.
- **Respetar a las personas:** la gente trabaja mejor cuando se encuentra en un ambiente que la motive y se sienta respetada.
- **Optimizar el todo:** optimizar todo el proceso, ya que el proceso es una unidad, y para lograr tener éxito y avanzar, hay que tratarlo como tal.

Proceso

Scrum es un proceso iterativo e incremental donde se busca poder atacar todos los problemas que surgen durante el desarrollo del proyecto. Durante los ***Sprints*** las fases de desarrollo se solapan.

1. **Product backlog:** lista de funcionalidades ordenada por prioridad desde la perspectiva del cliente.
2. **Sprint planning meeting:** Se planifica la entrega en el tiempo acordado para la duración del sprint.
3. **Sprint backlog:** las funcionalidades del *product backlog* que se desarrollará durante el sprint.
4. **Daily scrum:** reunión ágil, corta, de 15 a 30 minutos, donde se da visibilidad a los avances diarios
5. **Finished work**
    a. **Sprint review:** se lleva a cabo al final del sprint y se muestra lo implementeado en dicha iteración.
    b. **Sprint retrosprective:** se toma en cuenta los resultados del sprint, discutidos en el *sprint review.*

Kanban

Es un enfoque lean de desarrollo de software ágil. La filosofía Lean hace hincapié en eliminar cualquier actividad que no agregue valor a través de la mejora continua para agilizar las operaciones. Está centrada en el cliente y en ofrecer una mayor calidad, reducir el tiempo de ciclo y reducir los costos.

A diferencia de otras metodologías, no define ningún rol ni ninguna secuencia de pasos para llevar a cabo el desarrollo del proyecto.

Un **tablero Kanban** permiten visualizar las tareas *a realizar*, las *realizadas* y las *pendientes*, y permiten visualizar explícitamente el proceso de desarrollo. Está compuesto por una serie de columnas que representan los diversos estados que atraviesa un requerimiento durante el proceso de desarrollo. Las tarjetas se mueven de un estado a otro, mostrando la evolución, hasta que hayan sido aceptadas por el cliente. Es habitual combinar estos tableros en las prácticas de otras metodologías ágiles.

Desarrollo de software basado en modelos (MBD)

Este paradigma asigna a los modelos un rol central y activo: son al menos tan importantes como el código fuente. La construcción de un sistema de software debe ser precedida por la construcción de un modelo. Un **modelo** del sistema consiste en una conceptualización del dominio del problema y actúa como una especificación precisa de los requerimientos que el sistema de software debe satisfacer.

Model Driven Development (MDD) promueve enfatizar los siguientes puntos claves:

- Mayor nivel de **abstracción en la especificación**, tanto del problema a resolver como de la solución correspondiente.
- Aumento en la **confianza en la automatización** asistida por computadora para soportar el análisis, diseño y ejecución.
- Uso de **estándares industriales** como medio para facilitar las comunicaciones, la interacción entre diferentes aplicaciones y productos, y la especialización tecnológica.
- Los **modelos son los conductores primarios** en todos los aspectos del desarrollo de software.

Modelos de MDD. PIM y PSM

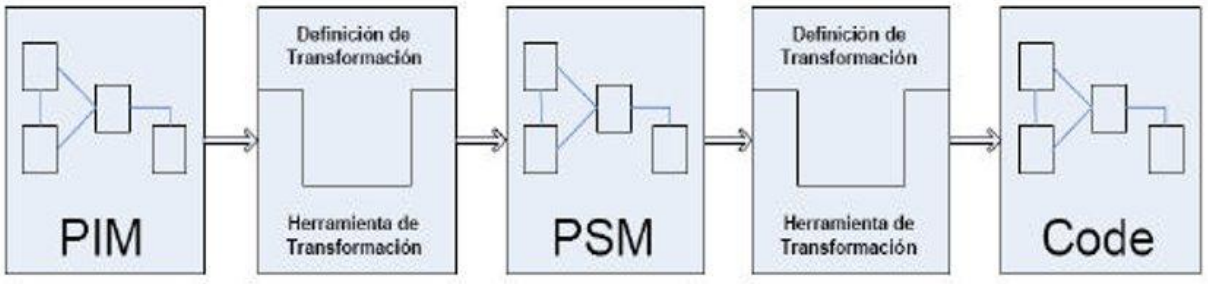
- **Platform independent model (PIM):** no contiene información sobre la plataforma y la tecnología con la que se implementará
- **Platform specific model (PSM):** contiene información sobre la plataforma y la tecnología con la que se implementará.

Transformación de modelos: proceso de conversión de un modelo a otro para el mismo sistema.

¿Qué es una transformación?

Una transformación consiste en un conjunto de reglas (especificaciones no ambiguas) de las formas en que un modelo, o parte de él, puede ser usado para crear otro modelo, o parte de él.

El patrón MDD normalmente es utilizado sucesivas veces para producir una sucesión de transformaciones.



Calidad

El significado de calidad es ambiguo y muchas veces su uso depende de lo que cada uno entiende por calidad.

Algunas definiciones formales son:

1. Propiedad o conjunto de propiedades inherentes a algo, que ***permiten juzgar su valor***.
2. Condición o requisito que se pone en un contrato.
3. ...

De esto se puede inferir que, la calidad, es un término totalmente subjetivo, que depende del juicio de la persona que intervenga en la evaluación.

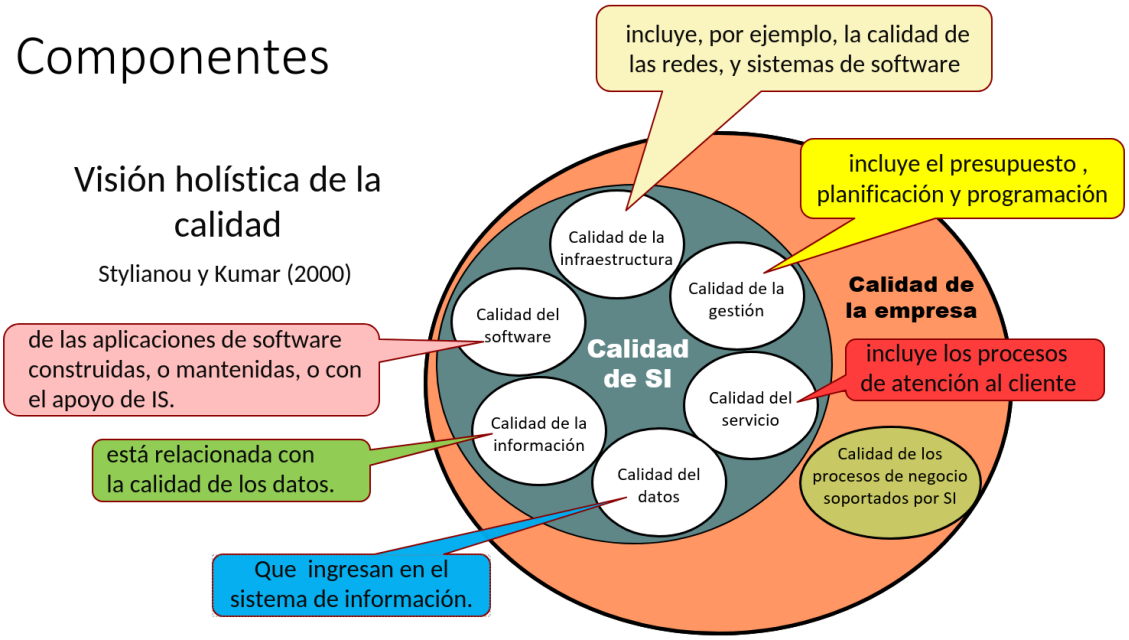
Según normas internacionales:

- **ISO 9000:** el grado en el que un conjunto de características inherentes cumple con los requisitos.
- **ISO 8402:** conjunto de propiedades o características de un producto o servicio que le confieren aptitud para satisfacer unas necesidades expresadas o implícitas.

  Una **norma** es un documento establecido por consenso y aprobado por un **organismo reconocido** que proporciona una serie de reglas, directrices o características para un uso común y repetido.

Sistemas de información

Un **sistema de información** es el conjunto de personas, datos, procesos y tecnología de información que interactúan para recopilar, procesar, guardar, y proporcionar como salida la información necesaria para brindar soporte a una organización.

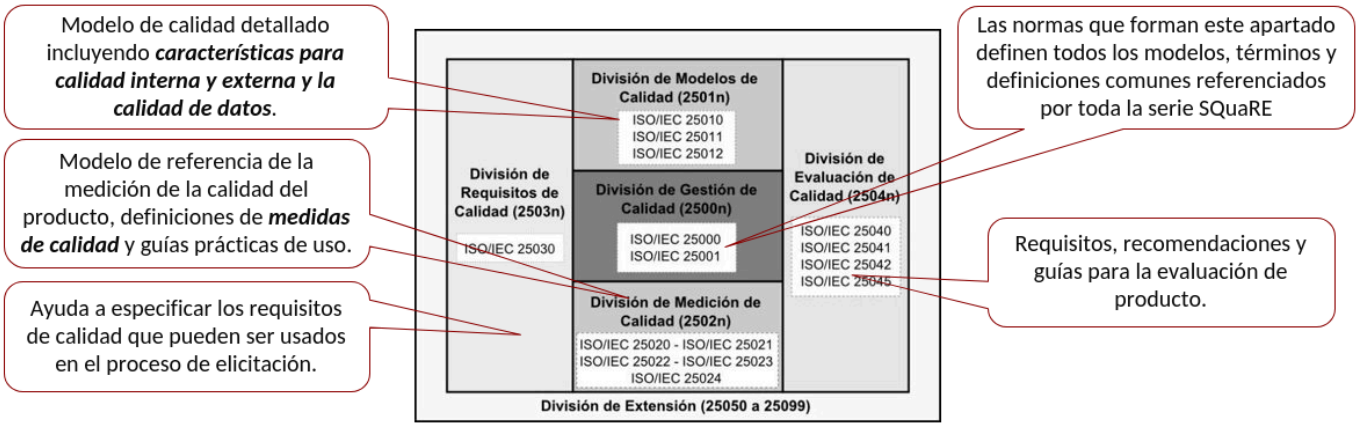


Calidad de software

Se divide en:

- **Calidad del producto obtenido:** define las propiedades que debe satisfacer el producto de software resultante.
- **Calidad del proceso de desarrollo:** define la manera de desarrollar el producto de software.

ISO/IEC 25000 SQuaRE: Software product Quality Requeriment and Evaluation:



Mejora de la calidad de los procesos de software

- **Modelo de procesos**
  - **ISO/IEC 12207:2008:** ISO/IEC 12207 establece un modelo de procesos para el ciclo de vida del software.
- **Modelo de evaluación**
  - **ISO/IEC 15504:** es una norma internacional para establecer y mejorar la capacidad y madurez de los procesos de las organizaciones en la adquisición, desarrollo, evolución, y soporte de productos y servicios.
  - **Familia de normas ISO/IEC 33000:** proporciona un marco de trabajo coherente para la evaluación de procesos de software que sustituye las diferentes partes de la norma ISO/IEC 15504.

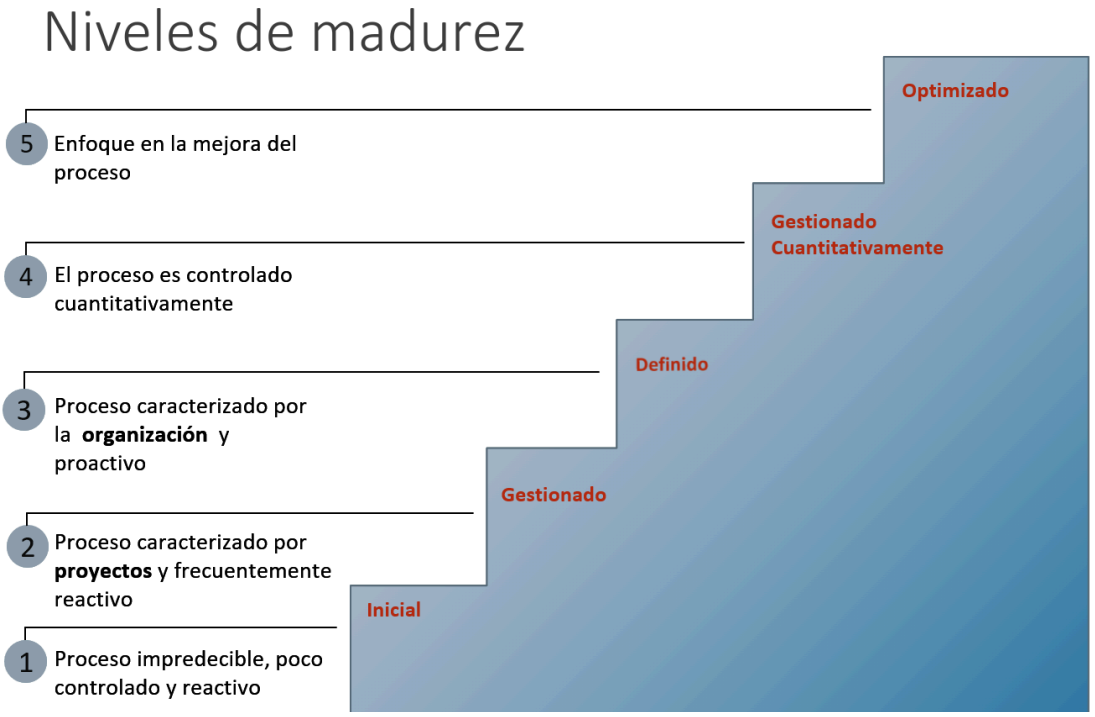
CMM (1993) - CMMI (2000)

CMM proporciona un marco estructurado para evaluar los procesos actuales de la organización, establecer prioridades de mejora e implementar esas mejorar.

CMMI posee dos vistas que permiten un enfoque diferente según las necesidades de quien vaya a implementarlo:

- **Escalonado:** centra su foco en la madurez de la organización (igual que CMM)

- **Continuo:** enfoca las actividades de mejora y evaluación en la capacidad de los diferentes procesos. Presenta 6 niveles de capacidad que indican qué tan bien se desempeña la organización en un área de proceso individual.



Familia de las ISO 9000

- **ISO – 9001:2015** - Quality management system – Requirements
- **IRAM – ISO 9001:2015** – Sistema de gestión de la calidad – Requisitos
- **ISO 90003:2004**
    - Basada ISO 9001:2000 (se espera una actualización para el próximo año)
    - Directrices para la interpretación en el proceso de software
    - Proporciona una guía para identificar la evidencias dentro del proceso de software para satisfacer los requisitos de la ISO 9001.

Beneficios de trabajar con un sistema de gestión de calidad (SGC)

- ISO 9001 asegura que su negocio cumpla con los requisitos legales y del cliente.
- Aumenta el rendimiento de su organización. El SGC, ayuda a implementar procesos simplificados y mejorar la eficiencia operacional.
- Asegura la toma de decisiones y mejora la satisfacción del cliente.
- Optimica sus operaciones para así cumplir y superar los requisitos de sus clientes.
- Mejora su rendimiento financiero.

Resumen

Calidad de producto de software	Se evalúa la calidad mediante	ISO/IEC 25000	Está compuesto por distintos modelos. Define características que pueden estar presentes o no en el producto. La norma nos permite evaluar si están presentes o no, y de qué manera evaluarlas. EJ: Seguridad, Compatibilidad, Seguridad. Etc.
Calidad de proceso de desarrollo de software	Se evalúa la calidad mediante	ISO/IEC 12207	ISO/IEC 12207 establece un modelo de procesos para el ciclo de vida del software. Define cómo debería ser el modelo de proceso para ser completo y con calidad. Actividades, tareas etc.
		ISO/IEC 15504 (reemplazada por ISO 33000)	Es una norma internacional para establecer y mejorar la capacidad y madurez de los procesos. Define que se debe tener en cuenta para evaluar el modelo de proceso y concluir si es completo y con calidad.
		ISO/IEC 90003	Proporciona una guía sobre cómo aplicar la ISO 9001 en procesos de software
		CMMI	Proporciona un marco estructurado para evaluar los procesos actuales de la organización, establecer prioridades de mejora, e implementar esas mejoras. Se utiliza para organizaciones desarrolladoras de software de medianas a grandes dimensiones
Calidad de Procesos/ Servicios en general	Se evalúa mediante	ISO 9001	La Norma ISO 9001 determina los requisitos para establecer un Sistema de Gestión de la Calidad. Forma parte de la familia ISO 9000, que es un conjunto de normas de “gestión de la calidad” aplicables a cualquier tipo de organización con el objetivo de obtener mejoras en la organización y, eventualmente arribar a una certificación, punto importante a la hora de competir en los mercados globales.