

Modelos de ciclo de vida:

Definición (ISO/IEC 12207:2008):

1. marco de referencia de los procesos y actividades relacionadas con la evolución del software (desde su concepción hasta su retirada) que pueden organizarse en etapas, las cuales actúan como una referencia común para la comunicación y la comprensión

Son definiciones generales que enumeran las fases por las que pasa el software a lo largo de su vida, así como las dependencias entre dichas fases. (“Los modelos no dicen cómo hacer las cosas, sino qué fases existen y cómo se relacionan.”)

Proceso del software (/procesos de software)

Pero se necesita definir más en detalle “qué es lo que hay que hacer” en cada una de esas fases

Definición general de proceso (ISO 9000:2005):

- conjunto de actividades interrelacionadas o que interactúan entre sí las cuales transforman entradas en salidas

Concretando al software:

- conjunto coherente de políticas, estructuras organizativas, tecnologías, procedimientos y artefactos que se necesitan para concebir, desarrollar, implantar y mantener un producto software (“Es el conjunto de tareas, políticas, herramientas y estructuras que se usan para desarrollar, implantar y mantener un software.”)

Objetivos (de definir modelos y procesos?):

- analizar los mejores modelos de organización de las tareas necesarias para “desarrollar” software,
- mediante la definición del orden y de los métodos con los que realizar las tareas se obtendrá un mejor software.

Características generales:

- no son únicos, (“No existe un único modelo válido para todos los casos”)
- se definen a nivel teórico, en cada organización se adaptan en su implantación,
- Son flexibles para cualquier ámbito, tamaño y/o tipo de software

✓ Características Generales de los Modelos de Ciclo de Vida del Software

1. No son únicos
 - Existen muchos modelos diferentes (cascada, espiral, incremental, ágil, etc.), y cada uno se adapta mejor a ciertos contextos.
2. Se definen a nivel teórico
 - Los modelos proporcionan una visión abstracta y general del proceso de desarrollo. No se aplican literalmente en la práctica.
3. Se adaptan en su implantación
 - Cada organización ajusta el modelo elegido según sus necesidades, recursos, cultura, etc.
4. Son aplicables a cualquier ámbito, tamaño y tipo de software
 - Un modelo bien definido puede usarse tanto para sistemas críticos como para apps simples, con los ajustes necesarios.

Qué definen (los procesos?): las actividades a realizar, y en qué orden.

- Las **actividades** son la unidad mínima de trabajo, la cual tiene una duración definida, está relacionada lógicamente con otras actividades (“algunas deben hacerse antes o después de otras”) y consume recursos (“(tiempo, personal, herramientas, etc.)”).

- a veces se agrupan en niveles más altos (fases) o se dividen en subactividades o tareas,

-Las actividades definen:

- qué se debe hacer (objetivo)
- quién lo hace (roles)
- productos que intervienen (entradas y salidas)

Marco de referencia, **normas IEC/ ISO 12207:2008**

Define un conjunto de procesos para el desarrollo del software.

Dos grupos de procesos:

Procesos de contexto del sistema y Procesos específicos del software.

Procesos de contexto del sistema

- No propios del software, más bien **relacionados con la organización**
- Subgrupos: contratación, gestión de proyectos, facilitadores de la organización, técnicos (relacionados con el sistema en su globalidad)

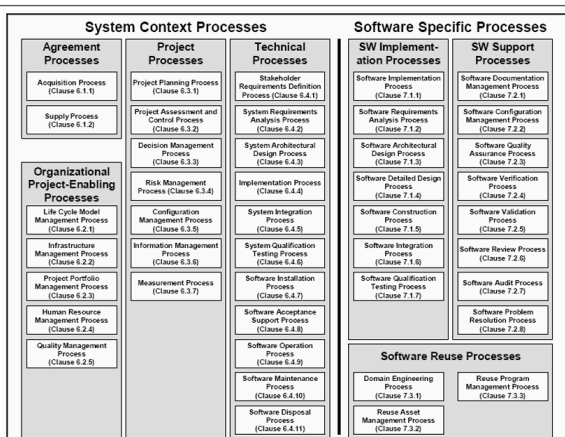
Procesos específicos del software

“Son los directamente relacionados con el desarrollo y mantenimiento del software.”

Tres subgrupos

- Implementación del software
- Soporte para el desarrollo del software
- Reutilización del software

Marco de referencia ISO/IEC 12207:2008 (y 3)



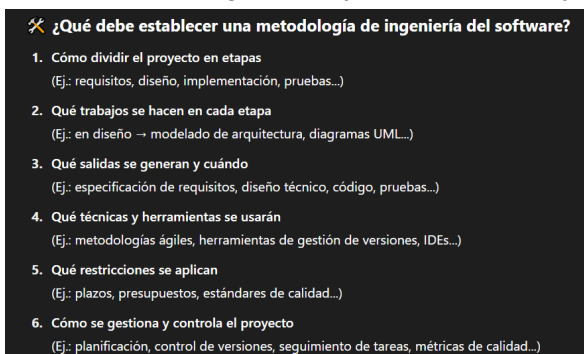
Metodologías de Ingeniería del Sw

- Definición: **conjunto integrado de técnicas y métodos que permiten obtener de forma homogénea y abierta, cada una de las actividades del ciclo de vida**

- **Métodos:** actividades llevadas a cabo para conseguir un objetivo.
- **Homogénea:** sistemática, que se utilice en todos los proyectos de una organización. **No debería definirse una nueva metodología para cada proyecto.**
- **Abierta:** a cambios y adaptación según el proyecto que se va a llevar a cabo

-Una metodología de ingeniería del sw, debe establecer:

- cómo dividir un proyecto en etapas
- qué trabajos se llevan a cabo en cada etapa.
- qué salida se produce en cada etapa, y cuando se debe producir.
- qué técnicas y herramientas se van a utilizar en cada etapa.
- qué restricciones se aplican (de tiempo, coste, objetivos, etc.)
- cómo se gestiona y controla un proyecto



Objetivos:

- General: concebir el desarrollo como un proyecto de ingeniería del sw (“Esto implica aplicar principios de la ingeniería (planificación, análisis, diseño sistemático, control de calidad, documentación, etc.) al desarrollo de software, tratándolo como una disciplina rigurosa y profesional.”).

- Específicos:

- Mejorar la calidad del SW, al estar más controlado su desarrollo (“Al seguir una metodología, el proceso está más controlado y menos propenso a errores, lo que se traduce en productos finales de mayor calidad”)
- Facilitar el control de los proyectos, al seguir estos un método sistemático.
- Facilitar la comunicación y entendimiento entre los participantes
- Mejora de la productividad en el desarrollo del sw, gracias a:
 - una mayor capacidad de adaptación a cambios a realizar sobre el sw.
 - facilidad de reutilización de productos (“documentos, código, módulos”) y de personas (“los equipos se adaptan mejor si usan una misma metodología en todos los proyectos”) si se utiliza en todos los proyectos la misma metodología.
 - Que se consiguen tiempos de desarrollo y costes aceptables (“Al planificar y seguir procesos bien definidos, se reducen retrasos, sobrecostes y retrabajo.”)
 - Facilitar el mantenimiento del sw, gracias a la documentación.

- Conseguir satisfacer a todas las personas afectadas por el sw (usuarios, clientes, directivos, auditores).

Ejemplo de metodologías

- Metodologías oficiales, que están patrocinadas por gobiernos.
 - Métrica 3 (España)
 - Merise (Francia)
 - SSADM (UK)
- Metodologías no oficiales. Las definen las universidades, las empresas, expertos, etc.
 - Yourdon (USA)
 - Unified Process (Rumbaugh, Booch, Jacobson)

Gestión de configuración

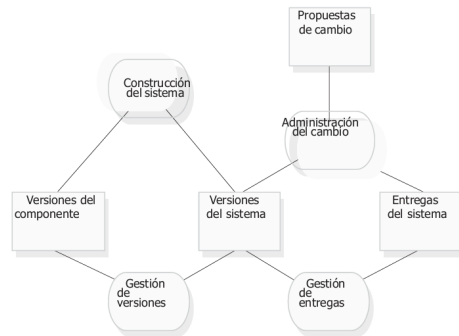
- Debido a que el software cambia con frecuencia, los sistemas pueden considerarse como un conjunto de versiones, cada una de las cuales debe ser mantenida y administrada.
- Las versiones implementan propuestas para cambios, correcciones de fallas y adaptaciones para diferentes hardware y sistemas operativos.
- La gestión de la configuración (CM) se ocupa de las **políticas, procesos y herramientas** para gestionar sistemas de software cambiantes. Necesita CM porque es fácil perder de vista qué cambios y versiones de componentes se han incorporado en cada versión del sistema (“Controlar, registrar y gestionar todos los cambios y versiones que sufre un sistema de software a lo largo del tiempo.”)

Actividades de la gestión de la configuración

- **Gestión del cambio**
 - **Realizar un seguimiento de las solicitudes de cambios** en el software de los clientes y desarrolladores, calcular los costos y el impacto de los cambios y **decidir si se deben implementar los cambios.**
- **Gestión de versiones**
 - Realizar un **seguimiento de las múltiples versiones de los componentes** del sistema y garantizar que los **cambios realizados en los componentes por diferentes desarrolladores no interfieran entre sí.**
- **Sistema de construcción**
 - **El proceso de ensamblar componentes de programa, datos y bibliotecas, luego compilarlos para crear un sistema ejecutable.**
- **Gestión de lanzamiento**
 - **Preparar el software para el lanzamiento externo y realizar un seguimiento de las versiones del sistema que se han lanzado para uso del cliente.** (“Decide qué versión del sistema se libera, con qué funcionalidades. Se asegura de que esté probada, documentada y empaquetada correctamente. Lleva un

registro de qué versión tiene cada cliente. Permite hacer rollbacks si una versión da problemas.”)

Actividades de la gestión de la configuración



?????

Terminología de la gestión de la configuración

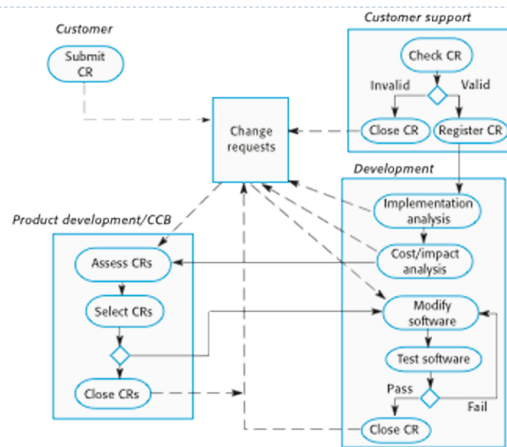
- Configuration item or software configuration item (SCI)
 - Todo lo relacionado con un proyecto de software (diseño, código, datos de prueba, documento, etc.) que se haya colocado bajo el control de configuración. A Menudo hay diferentes versiones de un elemento de configuración. Los Elementos De Configuración Tiene Un Nombre Único.
- Configuration control
 - El proceso de asegurar que las versiones los sistemas y componentes se registren y se mantengan para que los cambios se gestionen y todas las versiones de los componentes se identifiquen almacenen durante toda la vida útil del sistema..
- Version
 - Una instancia de un elemento de configuración (SCI) que difiere,de alguna manera, de otras instancias de ese elemento. Las versiones siempre tienen un identificador único,que a menudo se compone del nombre del elemento de configuración más el número de versión.
- Baseline
 - Una línea de base es una colección de versiones de componentes que conforman un sistema. Las líneas de base están controladas, lo que significa que las versiones de los componentes que forman el sistema no se pueden cambiar. Esto significa que siempre debería ser posible recrear una línea de base a partir de sus componentes constituyentes. (“Una colección controlada de versiones de SCIs que forman un sistema en un momento concreto. Una vez definida una baseline, no se puede cambiar (a menos que se cree una nueva). Permite reconstruir exactamente el sistema como estaba en ese punto.”)
- Codeline
 - Una línea de código es un conjunto de versiones de un componente de software y otros elementos de configuración de los que depende ese componente.

- Mainline
 - Una secuencia de líneas de base (baseline) que representan diferentes versiones del sistema.
- Release
 - Una versión de un sistema que se ha mandado a los clientes (u otros usuarios una organización) para su uso.
- Workspace
 - Un área de trabajo privada donde el software puede modificarse sin afectar a otros desarrolladores que pueden estar usando o modificando ese software.
- Branching
 - La creación de una nueva rama de código a partir de una versión en una rama de código existente. La nueva rama de código y la rama de código existente pueden entonces desarrollarse independientemente.
- Merging
 - La creación de una nueva versión de un componente de software mediante la fusión de versiones separadas en diferentes ramas de código. Estas ramas de código pueden haber sido creadas por una rama anterior de una de las ramas involucradas.
- System Building
 - La Creación De una versión del sistema ejecutable mediante la compilación y la vinculación de las versiones apropiadas de los componentes y las bibliotecas que conforman el sistema.

Gestión del cambio

- Las necesidades y los requisitos de la organización cambian durante la vida útil de un sistema, los errores deben ser reparados y los sistemas deben adaptarse a los cambios en su entorno.
- El objetivo de la gestión de cambios es garantizar que la evolución del sistema sea un **proceso administrado** y que se **dé prioridad a los cambios más urgentes y rentables**.
- El proceso de gestión de cambios se ocupa del análisis de los costos y beneficios de los cambios propuestos, la aprobación de aquellos cambios que valen la pena y el seguimiento de los componentes del sistema que se han modificado

Proceso de gestión del cambio



CR: Change Request

Check CR → Se revisa la solicitud:

Si es inválida → Se cierra directamente (Close CR).

Si es válida → Se registra (Register CR) y pasa al siguiente paso.

Product Development / CCB (Change Control Board):

Assess CRs → Se evalúan técnicamente.

Select CRs → Se decide cuáles se van a implementar.

Si no se van a hacer → Close CR.

Development (Desarrollo). Para cada CR aprobada:

Implementation analysis → Se analiza cómo se haría el cambio.

Cost/Impact analysis → Se estudian los costes y efectos que tendría ese cambio.

Modify software → Se realiza el cambio en el código.

Test software → Se prueba el software actualizado.

Si pasa las pruebas → se cierra la CR (Close CR).

Si falla → el ciclo puede repetirse o reconsiderarse.

Factores a considerar para un cambio

- Las consecuencias de no hacer el cambio.
- Los beneficios del cambio.
- El número de usuarios afectados por el cambio. (Cambios que afectan a muchos usuarios suelen tener más prioridad. Ejemplo: si un error afecta solo a un cliente aislado, puede esperar; si afecta a todos, es urgente.)
- Los costes de hacer el cambio.
- El ciclo de lanzamiento del producto (¿En qué etapa del desarrollo o mantenimiento está el producto? Puede que no sea conveniente hacer cambios justo antes de una entrega importante o durante una fase crítica.)

Gestión del cambio en métodos ágiles

- En algunos métodos ágiles, **los clientes están directamente involucrados en la gestión del cambio. Proponen un cambio** en los requisitos y trabajan con el equipo

para **evaluar su impacto y decidir si el cambio debe tener prioridad sobre las características planificadas para el próximo incremento** del sistema.

- Los cambios para mejorar la mejora del software son decididos por los programadores que trabajan en el sistema. (“No todos los cambios vienen del cliente. Los desarrolladores también proponen mejoras internas, como reorganizar código, simplificar lógica o mejorar rendimiento. Ejemplo: un desarrollador detecta que una función es redundante y decide mejorarla para hacer el código más limpio.”)
- La **refactorización**, donde el software se mejora continuamente, no se ve como una sobrecarga, sino como una parte necesaria del proceso de desarrollo (“La refactorización consiste en mejorar el código sin cambiar su funcionalidad, hacerlo más limpio, más eficiente, más mantenible. En los métodos ágiles, no se considera una "pérdida de tiempo" ni un esfuerzo extra. Se ve como una parte fundamental del desarrollo continuo y de la calidad del producto.”)

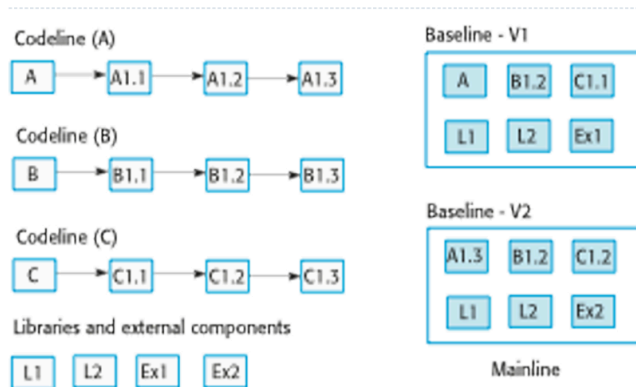
Gestión de versión

- La administración de versiones (VM) es el proceso de seguimiento de diferentes versiones de componentes de software o elementos de configuración y los sistemas en los que se utilizan estos componentes.
- También implica asegurar que los cambios realizados por diferentes desarrolladores en estas versiones no interfieran entre sí.
- Por lo tanto, la administración de versiones se puede considerar como el proceso de administración de líneas de código y líneas de base.

Línea (rama) de código y línea base (codeline y baseline)

- Una **línea de código** es una secuencia de versiones del código fuente (de un componente!). Cada versión en la secuencia deriva de versiones anteriores.
 - Las líneas de código normalmente se aplican a los componentes de los sistemas, de modo que hay diferentes versiones de cada componente.
- Una **línea de base** es una definición de un sistema específico. Por lo tanto, la línea de base especifica las versiones de los componentes que se incluyen en el sistema más una especificación de las bibliotecas utilizadas, los archivos de configuración, etc

Línea (rama) de código y línea base



25

Módulo 6. Procesos

Líneas base

- Las líneas de base se pueden especificar utilizando un **lenguaje de configuración**, que le permite definir qué componentes se incluyen en una versión de un sistema en particular.
- “Lenguaje de configuración: Es un lenguaje o formato específico (a menudo declarativo o estructurado, como XML, JSON, YAML o scripts) que se utiliza para describir qué versiones de componentes forman parte de una línea base. Este lenguaje permite:
 - Declarar explícitamente qué archivos, bibliotecas, configuraciones y versiones forman el sistema.
 - Automatizar la reconstrucción exacta de una versión del sistema.
 - Evitar errores humanos en la selección de versiones.”
- Las líneas de base son importantes porque a menudo hay que recrear una versión específica de un sistema completo.
- Por ejemplo, se puede crear una instancia de una línea de productos para que haya versiones individuales del sistema para diferentes clientes. Es posible que tenga que volver a crear la versión entregada a un cliente específico si, por ejemplo, ese cliente reporta errores en su sistema que deben ser reparados.

Sistemas de gestión de versiones

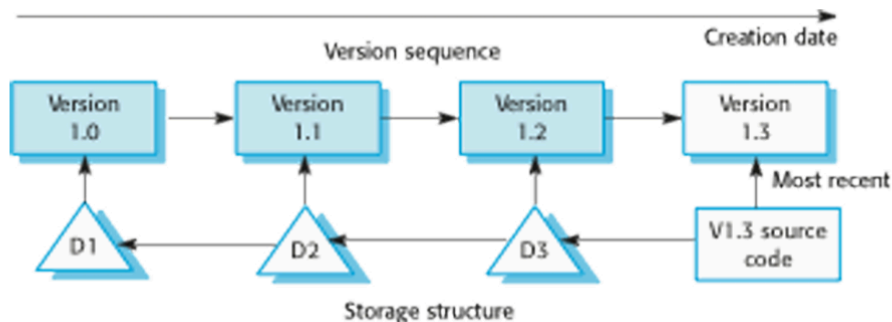
“Es un sistema que registra, organiza y controla todas las versiones de archivos (código, documentación, etc.) a lo largo del tiempo”

- Identificación de versión
 - A las versiones administradas se les asignan identificadores cuando se envían al sistema.
- Administración de almacenamiento
 - Para reducir el espacio de almacenamiento requerido por las múltiples versiones de componentes que difieren sólo ligeramente, los sistemas de

administración de versiones generalmente proporcionan servicios de administración de almacenamiento. (“Para ahorrar espacio, se almacenan solo los cambios (deltas) entre versiones, no copias completas.”)

- Historial de cambios
 - Todos los cambios realizados en el código de un sistema o componente se registran y se enumeran.
- Desarrollo independiente sin interferencias
 - El sistema de gestión de versiones realiza un seguimiento de los componentes que se han revisado para su edición y garantiza que los cambios realizados en un componente por diferentes desarrolladores no interfieran.
- Versiones compartidas por varios proyectos
 - Un sistema de gestión de versiones puede soportar el desarrollo de varios proyectos, que comparten componentes

Gestión de almacenamiento usando incrementos



“En lugar de guardar cada versión completa, se almacena:

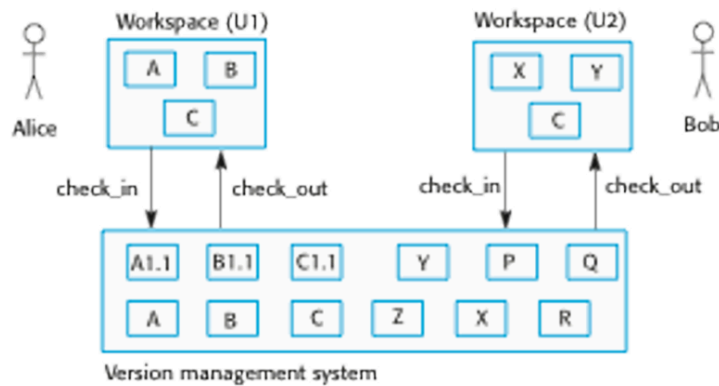
Version 1.0 como versión completa

D2 = diferencias entre 1.0 y 1.1

D3 = diferencias entre 1.1 y 1.2

V1.3 source code = versión completa más reciente (para acceso rápido)”

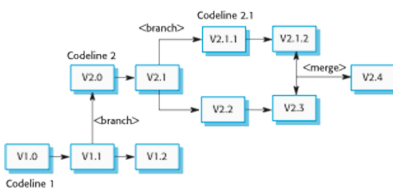
Registro de entrada y salida de un repositorio de versiones



Ramas de código

- En lugar de una secuencia lineal de versiones que reflejan cambios en el componente a lo largo del tiempo, puede haber varias secuencias independientes.
- Esto es normal en el desarrollo de sistemas, donde diferentes desarrolladores trabajan independientemente en diferentes versiones del código fuente y, por lo tanto, lo cambian de diferentes maneras.
- En algún momento, puede ser necesario combinar las ramas de la línea de código para crear una nueva versión de un componente que incluya todos los cambios que se hayan realizado.
- Si los cambios realizados involucran diferentes partes del código, las versiones de los componentes pueden fusionarse automáticamente combinando los deltas que se aplican al código

Ramificación y confluencia



Puntos Clave (resumen)

- La gestión de la configuración es la gestión de un sistema de software en evolución. Al mantener un sistema, se establece un equipo de CM para garantizar que los cambios se incorporen al sistema de forma controlada y que los registros se mantengan con detalles de los cambios que se han implementado.
- Los principales procesos de gestión de la configuración son la gestión de cambios, la gestión de versiones y la creación de sistemas.

- La gestión del cambio implica evaluar las propuestas de cambio de los clientes del sistema y otras partes interesadas y decidir si es rentable implementarlas en una nueva versión de un sistema.
- La administración de versiones implica realizar un seguimiento de las diferentes versiones de los componentes de software a medida que se realizan cambios en ellos

Construcción del sistema

- La creación del sistema es el proceso de crear un sistema completo y ejecutable mediante la compilación y la vinculación de los componentes del sistema, bibliotecas externas, archivos de configuración, etc.
 - “Es el proceso de ensamblar todas las partes del software para generar una versión ejecutable del sistema.”
- Las herramientas de creación de sistemas y las herramientas de administración de versiones deben comunicarse, ya que el proceso de creación implica verificar las versiones de componentes del repositorio administrado por el sistema de administración de versiones.

(“La herramienta de construcción debe estar conectada al sistema de versiones porque:

 - Necesita descargar las versiones correctas de todos los componentes desde el repositorio.
 - Tiene que usar exactamente los archivos definidos en una línea base (baseline), sin errores ni versiones equivocadas.”)
- La **descripción de la configuración** utilizada para identificar una línea de base también es utilizada por la herramienta de construcción del sistema. (“La descripción de configuración es como un archivo de instrucciones que dice: Qué componentes usar, Qué versiones exactas, Qué dependencias externas instalar, Cómo debe compilarse todo. Este archivo es utilizado tanto para: Definir una línea base (baseline) del sistema, como para Indicarle a la herramienta de construcción qué debe ensamblar y cómo. EJ MAKEFILE”)

Plataformas de Construcción

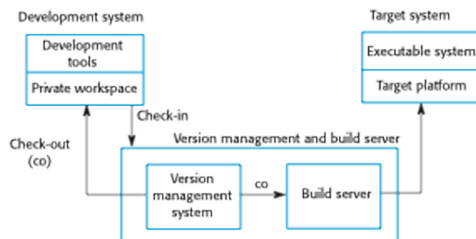
“Es el conjunto de herramientas, procesos y entornos que permite generar versiones ejecutables de un sistema de software a partir del código fuente y otros componentes. incluye:”

- El **sistema de desarrollo** incluye herramientas de desarrollo como compiladores, editores de código fuente, etc.
 - Los desarrolladores verifican el código del **sistema de administración de versiones** en un espacio de trabajo privado antes de realizar cambios en el sistema. (“Los desarrolladores trabajan en su espacio de trabajo privado (workspace): Hacen cambios localmente, Prueban el código, y luego lo suben al repositorio con un check-in”)
- El **servidor de compilación**, que se utiliza para compilar versiones definitivas y ejecutables del sistema (“generar automáticamente versiones compiladas y ejecutables del software.”).

- Los desarrolladores ingresan el código de entrada al sistema de administración de versiones antes de que se construya. **La compilación del sistema puede depender de bibliotecas externas que no están incluidas en el sistema de administración de versiones.**

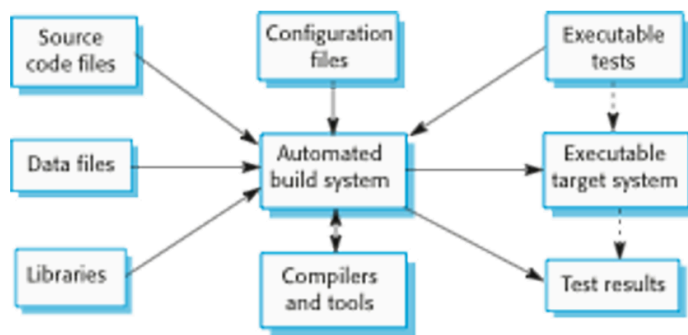
- El entorno de destino, que es la plataforma en la que se ejecuta el sistema (servidor web, android, aws...)

Desarrollo, construcción y plataformas de destino.



??

Construcción del sistema



“Automated Build System:

Es una herramienta que toma todos los elementos necesarios del software y los compila, ensambla, configura y prueba automáticamente para generar:

Un sistema ejecutable, Tests automáticos Y los resultados de esas pruebas.

Ejemplos: Jenkins, Maven, Gradle, Make, GitHub Actions. “

“Source code files: Archivos de código fuente (.java, .py, .cpp, etc.).

Data files: Archivos necesarios en tiempo de ejecución (configuraciones, bases de datos, imágenes, etc.).

Libraries: Dependencias externas necesarias para compilar o ejecutar el sistema (por ejemplo, numpy, log4j, etc.).

Configuration files: Archivos que indican cómo debe construirse el sistema:

Ej.: pom.xml, build.gradle, Makefile, Dockerfile.

Compilers and tools: Herramientas necesarias para compilar y empaquetar (como gcc, javac, tsc...). “

“A la derecha están los resultados del proceso automatizado:

Executable target system: El sistema final generado: una app, un servicio web, un programa de escritorio, etc.

Executable tests: Pruebas que se compilan y ejecutan automáticamente para comprobar que todo funciona.

Test results: Resultados de las pruebas (éxito, fallos, errores), que pueden guardarse en reportes o dashboards.”

Funcionalidades de la construcción del sistema

- **Generar script de construcción** (“Crear un archivo que describa cómo construir el sistema paso a paso.”)
- Integración del sistema de gestión de versiones.
- Re-compilación mínima (“Solo vuelve a compilar los archivos que han cambiado, no todo el sistema.”)
 - Las herramientas para apoyar la construcción del sistema generalmente están diseñadas para minimizar la cantidad de compilación que se requiere.
 - Lo hacen comprobando si hay disponible una versión compilada de un componente. Si es así, no hay necesidad de volver a compilar ese componente. (“si el archivo no ha cambiado desde la última vez, no lo recompila”)
 - Una firma única (*1) identifica cada fuente y la versión del código objeto y se cambia cuando se edita el código fuente. Al comparar las firmas en los archivos de código fuente y objeto, es posible decidir si el código fuente se usó para generar el componente de código objeto
- Creación de sistema ejecutable.
- Automatización de pruebas (“Ejecutar pruebas automáticas después de compilar para asegurar que todo sigue funcionando.”)
- Notificación de la construcción (“Informar a los desarrolladores del estado del proceso de construcción.”)
- Generación de documentación (“Crear documentación automáticamente a partir del código fuente o comentarios. javadoc...”)

Identificación de ficheros (*1)

- Marcas de tiempo de modificación

La firma en el archivo de código fuente es la fecha y hora en que se modificó ese archivo. Si el archivo de código fuente de un componente se ha modificado después del archivo de código objeto relacionado, entonces el sistema asume que es necesaria la recompilación para crear un nuevo archivo de código objeto.

- Sumas de comprobación de código fuente

La firma en el archivo de código fuente es una suma de comprobación calculada a partir de los datos en el archivo. Una función de suma de comprobación calcula un número único utilizando el texto de origen como entrada. Si cambia el código fuente (incluso por 1 carácter), esto generará una suma de comprobación diferente. Por lo tanto, puede estar seguro de que los archivos de código fuente con diferentes sumas de comprobación son realmente diferentes

(“Cada archivo fuente tiene una firma (checksum o fecha de modificación). Esa firma también se guarda en el archivo objeto correspondiente. Si el código fuente no ha cambiado, su firma es igual a la de la versión compilada anterior.”)

Gestión de lanzamientos

- Una versión de lanzamiento es una versión de un sistema de software que se distribuye a los clientes.
- **Para el software de mercado masivo**, generalmente es posible identificar dos tipos de lanzamientos: **los lanzamientos principales** que ofrecen una funcionalidad nueva e importante, y los **lanzamientos menores**, que reparan errores y solucionan los problemas de los clientes que se han reportado.
- **Para software personalizado** o líneas de productos de software, las **versiones del sistema deben producirse para cada cliente** y los **clientes individuales pueden ejecutar varias versiones diferentes del sistema al mismo tiempo**.

Reproducción de lanzamientos

“Es la capacidad de volver a construir exactamente la misma versión del software que se lanzó, incluso tiempo después.”

- Para documentar un lanzamiento, hay que registrar las versiones específicas de los componentes del código fuente que se utilizaron para crear el código ejecutable.
- Debe conservarse las copias de los archivos de código fuente, los archivos ejecutables correspondientes y todos los archivos de datos y configuración.
- También hay que registrar las versiones del sistema operativo, las bibliotecas, los compiladores y otras herramientas utilizadas para compilar el software (“debes haber guardado todo lo que había en la línea base en ese momento.”)

Planificación de lanzamientos.

- Además del trabajo técnico relacionado con la creación de una distribución de lanzamiento, se debe preparar material publicitario y se deben implementar estrategias de marketing para convencer a los clientes de que compren el nuevo lanzamiento del sistema.
- Tiempo de liberación
 - Si las versiones son demasiado frecuentes o requieren actualizaciones de hardware, los clientes no pueden pasar a la nueva versión, especialmente si tienen que pagar por ello.

- Si las versiones del sistema son demasiado infrecuentes, la cuota de mercado puede perderse a medida que los clientes pasan a sistemas alternativos.

Componentes de un lanzamiento

Además del código ejecutable del sistema, una versión también puede incluir:

- archivos de configuración que definen cómo se debe configurar el lanzamiento para instalaciones particulares;
- archivos de datos, como los archivos de mensajes de error, que son necesarios para el funcionamiento exitoso del sistema;
- un programa de instalación que se utiliza para ayudar a instalar el sistema en el hardware de destino;
- documentación electrónica y en papel que describe el sistema;
- Empaquetados y publicidad asociada que han sido diseñados para ese lanzamiento

Factores que afectan a la planificación de lanzamientos

- Calidad técnica del sistema
 - Si se notifican fallos graves del sistema que afectan la forma que muchos clientes usan el sistema, puede ser necesario emitir una versión de reparación de fallos. Los fallos menores del sistema se pueden reparar mediante la publicación de parches (generalmente distribuidos a través de Internet) que se pueden aplicar a la versión actual del sistema.
- Cambios de plataforma
 - Es posible que se deba crear una nueva versión de aplicación de software cuando se lance una nueva versión de la plataforma del sistema operativo.
- Ley quinta de Lehman
 - Esta "ley" sugiere que si se agregan muchas funcionalidades nuevas a un sistema también se introducirán errores que limitarán la cantidad de funcionalidad que puede incluirse en la próxima versión. Por lo tanto, una versión del sistema con una funcionalidad nueva e importante puede ser seguida por una versión que se centra en reparar problemas y mejorar el rendimiento
- Competencia
 - Para el software mercado masivo, puede ser necesaria una nueva versión del sistema porque un producto de la competencia ha introducido nuevas características y la participación en el mercado puede perderse si no se proporcionan a los clientes existentes.
- Requerimientos del marketing
 - El Departamento De Marketing de Una Organización Puede haberse comprometido a que los lanzamientos estén disponibles en una fecha determinada.
- Propuestas De cambios del cliente
 - Para sistemas personalizados, los clientes pueden haber realizado y pagado un conjunto específico de propuestas de cambio de sistema, y esperan una versión del sistema tan pronto como se hayan implementado.

Puntos clave

- La creación del sistema es el proceso de ensamblar componentes del sistema en un programa ejecutable para ejecutarse en un sistema informático de destino.
- El software se debe reconstruir y probar con frecuencia inmediatamente después de que se haya construido una nueva versión. Esto facilita la detección de errores y problemas que se han introducido desde la última compilación.
- Las versiones del sistema incluyen código ejecutable, archivos de datos, archivos de configuración y documentación. La administración de lanzamientos implica tomar decisiones sobre las fechas de lanzamiento del sistema, preparar toda la información para su distribución y documentar cada lanzamiento del sistema

Ética profesional en la Ing. Software

- La ética profesional pretende regular las actividades que se realizan en el marco de una profesión.
- En 1999 las organizaciones ACM y IEEE Computer Society publicaron el “Código de ética y práctica profesional de la Ingeniería del Software”
 - Define: “Son Ingenieros de Software quienes contribuyen, mediante participación directa o enseñanza, al análisis, la especificación, el diseño, el desarrollo, la certificación, el mantenimiento y pruebas de sistemas de software”

Los principios identifican las diferentes relaciones en las que los individuos, grupos y organizaciones participan, y las principales obligaciones de tales relaciones. El código contiene **8 principios clave** (desglosados en varias cláusulas cada uno):

- **Sociedad** “Los ingenieros de software deberán actuar de manera coherente con el interés general”
- **Cliente y empresario:** “Los ingenieros de software deberán actuar de tal modo que se sirvan los mejores intereses para sus clientes y empresarios, y consecuentemente con el interés general”
- **Producto:** “Los ingenieros de software deberán garantizar que sus productos y las modificaciones relacionadas con ellos cumplen los estándares profesionales de mayor nivel que sea posible”
- **Juicio:** “Los ingenieros de software deberán mantener integridad e independencia en su valoración profesional”. (“Emitir opiniones técnicas objetivas y fundamentadas, aunque no sean las que el cliente quiere oír. No dejarse influenciar por intereses externos.”)
- **Gestión:** “Los gestores y líderes en ingeniería de software suscribirán y promoverán un enfoque ético a la gestión del desarrollo y el mantenimiento del software” (“fomentar entornos justos, transparentes y con prácticas éticas en todo el equipo”)
- **Profesión:** “Los ingenieros de software deberán progresar en la integridad y la reputación de la profesión, coherentemente con el interés general”.
- **Compañeros:** “Los ingenieros de software serán justos y apoyarán a sus compañeros”.

- **Persona:** “Los ingenieros de software deberán participar en el aprendizaje continuo de la práctica de su profesión y promoverán un enfoque ético en ella”.

Tema 7: CALIDAD

Gestión de la calidad del software

Para garantizar que se alcance el nivel de calidad requerido en un producto de software hay tres actuaciones principales:

- A nivel **organizativo**, la gestión de la calidad se ocupa del **establecimiento de un marco de procesos y estándares organizativos que conduzcan a un software de alta calidad**. (“Se define el marco general de calidad para toda la empresa. Establece políticas, procesos y estándares que toda la organización debe seguir.”)
- A nivel **de proyecto**, la gestión de la calidad también se preocupa por establecer un **plan de calidad** para un proyecto. El plan de calidad debe establecer los objetivos de calidad para el proyecto y definir qué procesos y estándares se utilizarán.
- A nivel **de proyecto**, la gestión de la calidad implica la aplicación de procesos de calidad específicos y la **verificación de que se han seguido estos procesos** planificados.

Actividades de la Gestión de la calidad del sw

- La gestión de la calidad proporciona una verificación independiente del proceso de desarrollo del software. (“La gestión de calidad no participa directamente en el desarrollo, sino que lo supervisa de forma externa e imparcial.”)
- El proceso de gestión de la calidad verifica los entregables del proyecto para garantizar que sean coherentes con los estándares y objetivos de la organización (“y los requisitos del proyecto(? ”).
- El **equipo de calidad debe ser independiente del equipo de desarrollo** para que puedan tener una visión objetiva del software. Esto les permite informar sobre la calidad del software sin ser influenciados por problemas de desarrollo de software.

Actividades de la Gestión de la calidad del sw



Plan de calidad del sw

- Un plan de calidad establece las **cualidades deseadas del producto y cómo se evalúan y define los atributos de calidad más significativos.**
- El plan de calidad debe **definir el proceso de evaluación de la calidad.**
- Debe establecer **qué estándares de organización deben aplicarse** y, cuando sea necesario, **definir nuevos estándares** que se utilizarán
 - (“Es un documento formal que define: Qué significa “calidad” para el proyecto en concreto, Cómo se va a medir y verificar esa calidad, Qué normas o estándares deben seguirse, Qué riesgos pueden afectar la calidad y cómo se gestionarán.”)

Estructura del plan de calidad

- **Introducción del producto;** Describe brevemente el sistema o software a desarrollar.
- **Planes de productos;** Define qué entregables se generarán y en qué fases del proyecto. También puede incluir cómo se almacenarán, entregarán y verificarán.
- **Descripciones de procesos;** Especifica qué procesos de calidad se seguirán. Revisiones de código, Pruebas unitarias, de integración, etc. Validación con el cliente...+ que pruebas se harán respecto a los atributos de calidad y con qué criterios se considerarán aceptables...)
- **Objetivos de calidad;** Define las metas de calidad medibles que el software debe alcanzar.
- **Riesgos y gestión de riesgos.** Identifica los riesgos que podrían afectar la calidad y cómo se gestionarán.

Los planes de calidad deben ser documentos cortos y concisos. Si son demasiado largos, nadie los leerá.

Alcance del Plan de Calidad

- La gestión de la calidad es particularmente importante para sistemas grandes y complejos. La documentación de calidad es un registro del progreso y respalda la continuidad del desarrollo a medida que cambia el equipo de desarrollo.
 - (“Tienen muchas partes, equipos y fases. Errores son costosos o críticos. El equipo de desarrollo puede cambiar con el tiempo, así que se necesita una base documental clara para garantizar continuidad..”)
- Para sistemas más pequeños, la gestión de calidad necesita menos documentación y debe centrarse en establecer una cultura de calidad.

Calidad del software

- Calidad, de manera simplista, significa que un producto debe cumplir con sus especificaciones. Esto es problemático para los sistemas de software.
 - Existe una tensión entre los requisitos de calidad del cliente (eficiencia, confiabilidad, etc.) y los requisitos de calidad del desarrollador (capacidad de mantenimiento, reutilización, etc.);
 - Algunos requisitos de calidad son difíciles de especificar de manera no ambigua;

- Las especificaciones del software suelen ser incompletas y, a menudo, inconsistentes.
- El enfoque puede ser "**idoneidad para el propósito**" en lugar de conformidad con las especificaciones.

Idoneidad para el propósito del software

¿Se han seguido los estándares de programación y documentación en el proceso de desarrollo?

¿Se ha probado correctamente el software?

¿Es el software suficientemente confiable para ser puesto en uso?

¿El rendimiento del software es aceptable para el uso normal?

¿Se puede usar el software?

¿El software está bien estructurado y es comprensible?

Conflictos en la calidad

No es posible optimizar ningún sistema para todos los atributos de calidad; por ejemplo, mejorar la robustez puede llevar a la pérdida de rendimiento. Por lo tanto, **el plan de calidad debe definir los atributos de calidad más importantes** (en objetivos de calidad) para el software que se está desarrollando.

El plan también debe incluir una **definición del proceso de evaluación de la calidad** (en descripciones de procesos), una forma acordada de evaluar si alguna calidad, como la capacidad de mantenimiento o la solidez, está presente en el producto

Calidad del producto y del proceso

- **La calidad de un producto desarrollado está influenciada por la calidad del proceso de producción.**
- Esto es importante en el desarrollo de software ya que algunos atributos de calidad del producto son difíciles de evaluar. ("si no puedes medir bien ciertos atributos del producto final, entonces necesitas confiar más en la calidad del proceso (revisiones, pruebas, buenas prácticas) para asegurar indirectamente la calidad del producto." ??)
- Sin embargo, existe una relación muy compleja y poco comprendida entre los procesos de software y la calidad del producto.
- La aplicación de habilidades individuales y experiencia es particularmente importante en el desarrollo de software;
- Los factores externos, como la novedad de una aplicación (crear algo totalmente nuevo) o la necesidad de un programa de desarrollo acelerado, pueden afectar la calidad del producto.

Uso de standards(normativas) de software

- Las normas definen los atributos requeridos de un producto o proceso. Juegan un papel importante en la gestión de la calidad.

- Los estándares pueden ser estándares internacionales, nacionales, organizacionales o de proyecto (ej ISO, UNE, reglas internas de una empresa o reglas definidas para x proyecto)
- Los estándares del producto definen las características que todos los componentes de software deben exhibir, por ejemplo. Un estilo de programación común.
- Los estándares de proceso definen cómo se debe promulgar el proceso de software (definición de fases de ciclo de vida, política de ctrl de versiones...) .

Importancia de los standards

- La implantación de mejores prácticas evita la repetición de errores pasados (“Los estándares recogen las mejores prácticas aprendidas con el tiempo”)
- Son un marco para definir qué significa la calidad en un entorno particular, es decir, la visión de calidad de esa organización; establece criterios medibles y valores compartidos.
- Brindan continuidad: el personal nuevo puede entender a la organización al comprender los estándares que se utilizan.

Problemas con los standards

- Es posible que los ingenieros de software no los vean como relevantes y actualizados.
- A menudo implican demasiado relleno burocrático (“Escribir documentos que nadie lee o usa, solo porque “lo exige el estándar”, para cada entrega, hay que rellenar plantillas de revisión, informes de validación, aprobaciones formales.....”)
- Si las herramientas de software no las admiten, a menudo se requiere un tedioso trabajo de llenado de formularios para mantener la documentación asociada con los estándares. (“ej: Un estándar exige cierta estructura en los comentarios del código, pero el IDE no tiene plantillas automáticas → el programador tiene que escribirlo a mano cada vez”)

Costes de la calidad

Costes de prevención:

- Controlar que el proceso cumpla los criterios de calidad establecidos y prevenir la aparición de defectos.
 - Incluye: Planificación actividades de calidad, formación de nuevos integrantes del equipo de desarrolladores, realización de informes del producto desarrollado.

Costes de evaluación de calidad:

- Asociados con actividades de verificación de la calidad.
 - Incluye: Verificar que el producto obtenido cumple los requisitos establecidos, inspección del producto entregado, mantenimiento de equipos de pruebas,

Costes fallos internos:

- Producidos por tareas de detección y reparación de defectos internos del software.
 - Incluye: coste de revisar el producto reparado, corrección de errores para hacer el producto utilizable, ...

Costes fallos externos:

- Derivados de corregir fallos encontrados en el software después de la entrega al cliente.
 - Incluye: análisis del software, restitución o reemplazo del producto, servicios y reparaciones durante período de garantía, ...

Calidad del producto. Modelo IEC/ ISO 9126

- Objetivo principal: Proporcionar una especificación de la calidad de productos software y un modelo de evaluación. Para conseguirlo define un lenguaje común para **especificar requisitos de calidad**. ("Permite que los equipos hablen el mismo lenguaje al definir requisitos de calidad.")
- Otros objetivos:
 - Establecer **medidas objetivas de calidad**.
 - **Evaluación de la calidad reproducible y sistemática**

Requisitos principales:

Calidad del producto. Modelo ISO/IEC 9126



Revisiones e inspecciones

- Un grupo **examina parte o la totalidad de un proceso o sistema y su documentación para encontrar problemas** potenciales. Código, diseños, especificaciones, planes de prueba, normas, etc., todos pueden ser revisados.
- El software o los documentos pueden ser "aprobados" **en una revisión** que signifique que el **progreso a la siguiente etapa de desarrollo ha sido aprobado** por la gerencia.
- Existen diferentes tipos de revisión con diferentes objetivos:
 - Inspecciones para la eliminación de defectos (producto);
 - Revisiones para la evaluación del progreso (producto y proceso);
 - Revisiones de calidad (producto y normas).

Proceso de revision del software



Inspecciones del programa

- Estas son revisiones por pares (desarrolladores examinan trabajo de otro desarrollador) donde los ingenieros examinan la fuente de un sistema con el objetivo de descubrir anomalías y defectos.
- Las inspecciones **NO requieren la ejecución** de un sistema, por lo que se pueden usar antes de la implementación.
 - Pueden **aplicarse a cualquier representación del sistema** (requisitos, diseño, datos de configuración, datos de prueba, etc.).
- Se ha demostrado que son una **técnica eficaz para descubrir errores** de programa.

VER PAGINAS DE REVISIONES E INSPECCIONES!! no entiendo diferencia

Tipo	Objetivo principal	Participantes típicos	En qué etapa se aplica	Enfoque	
Revisión (general)	Detectar errores, validar avance	Desarrolladores, líderes, QA	En cualquier etapa	General	
Inspección	Detectar defectos técnicos en el producto	Ingenieros pares	Antes o durante el desarrollo	Técnica, detallada	
Revisión de calidad	Verificar cumplimiento de normas/estándares	QA, gerencia, usuarios, otros roles	Antes de pasar de fase (revisión formal)	Normativa, evaluadora	

Listas de verificación de inspección

“Es un conjunto de preguntas o puntos de control que los ingenieros deben revisar durante una inspección de código o documentación, con el objetivo de detectar defectos comunes.????”

- Se debe utilizar la lista de verificación de errores comunes para conducir la inspección.
- Las listas de verificación de errores **dependen del lenguaje de programación** y reflejan los **errores característicos que probablemente surjan en el lenguaje**.
- En general, **cuanto más baja sea la comprobación de tipo, mayor será la lista de verificación**.
- Ejemplos: Inicialización, nombre de constantes, terminación de bucle, límites de matriz, etc.

Medida (métrica) de la calidad del software


- La medición de software se refiere a la obtención de un valor numérico para un atributo de un producto o proceso de software.
- Esto permite comparaciones objetivas entre técnicas y procesos (“Su objetivo es evaluar, comparar, controlar o mejorar la calidad del software de forma objetiva y sistemática.”)
- Aunque algunas compañías han introducido programas de medición, la mayoría de las organizaciones todavía no hacen un uso sistemático de la medición de software.
- Hay pocos estándares establecidos en esta área.

Métricas de software

- Cualquier tipo de medida relacionada con un sistema de software, proceso o documentación relacionada.
 - Ej: Líneas de código en un programa, el índice de niebla, la cantidad de días por persona requeridos para desarrollar un componente.
- Permitir que el software y el proceso del software se cuantifiquen.
- Puede usarse para predecir atributos del producto o para controlar el proceso del software.
- Las métricas del producto se pueden usar para predicciones generales o para identificar componentes anómalos ("Predecir esfuerzo, tamaño, complejidad o costo, o Detectar anomalías en componentes o en la productividad")



Diferencia principal

Aspecto	Medidas de calidad del software	Métricas de software (general)	
Enfoque	Evaluar la calidad del producto	Cuantificar todo el ciclo de vida del software	
Relación con estándares	Basado en modelos como ISO/IEC 9126 o 25010	No necesariamente, puede ser más operativo	
Aplicación principal	Control y mejora de la calidad	Predicción, control, análisis, estimación	
Ejemplos	Fiabilidad, eficiencia, mantenibilidad	LOC, esfuerzo, legibilidad, velocidad de desarrollo	

??? diferenciaaaa

Uso de métricas

- Para asignar un valor a los atributos de calidad del sistema
 - Al medir las características de los componentes del sistema, como su complejidad ciclomática, y luego agregar estas mediciones, puede evaluar los atributos de calidad del sistema, como la capacidad de mantenimiento.
 - Identificar los componentes del sistema cuya calidad es inferior a la estándar.
- Las mediciones pueden identificar componentes individuales con características que se desvían de la norma.
 - Por ejemplo, puede medir componentes para descubrir aquellos con la mayor complejidad. Es más probable que estos contengan errores porque la complejidad los hace más difíciles de entender.

Supuestos en las métricas

- Una propiedad de software puede ser medida.
- La relación existe entre lo que podemos medir y lo que queremos saber (ej: complejidad ciclomática alta → más difícil de mantener → menos calidad.)

- Solo podemos medir los atributos internos, pero a menudo estamos más interesados en los atributos de software externos. Puede ser difícil relacionar lo que se puede medir con atributos de calidad externa deseables. (“usabilidad, satisfacción por el cliente... medimos atributos internos y deducimos la calidad externa a partir de eso.”)
 - Esta relación ha sido formalizada y validada.: “se refiere a que la conexión entre lo que medimos (atributos internos del software) y lo que queremos evaluar (atributos de calidad externa, como mantenibilidad, fiabilidad, usabilidad, etc.) ha sido estructurada en modelos teóricos, y en algunos casos, comprobada empíricamente.”

Problemas con la medición en la industria.

- Es imposible cuantificar el retorno de la inversión al introducir un programa de métricas en las empresas.
- No hay estándares para métricas de software o procesos estandarizados para medición y análisis. (“A diferencia de otras disciplinas de ingeniería, en software no hay normas ampliamente aplicadas para definir: Qué métricas usar, Cómo recolectarlas, Cómo analizarlas o interpretarlas”)
- En muchas empresas, los procesos de software no están estandarizados y están mal definidos y controlados. (“Si no hay un proceso claro, es difícil saber dónde y qué medir.”)
- La mayoría del trabajo en medición de software se ha centrado en métricas basadas en código y procesos de desarrollo impulsados por planes.
- La introducción de medidas agrega una sobrecarga adicional a los procesos (tiempo, herramientas...).

Métricas de productos

- Una métrica de calidad debe ser un indicador de la calidad del producto. (“Son aquellas métricas que evalúan directamente el software como producto, es decir, su código, comportamiento en ejecución, estructura interna, rendimiento...”)
- Clases de métricas de producto:
 - **Métricas dinámicas** que son recolectadas por mediciones hechas de un programa en ejecución.
 - Ayudan a evaluar la eficiencia y confiabilidad.
 - están estrechamente relacionadas con los atributos de calidad del software.
 - Es relativamente fácil medir el tiempo de respuesta de un sistema (atributo de rendimiento) o el número de fallos (atributo de confiabilidad)
 - **Métricas estáticas** que se recopilan mediante mediciones hechas de las representaciones del sistema analizando su estructura (código fuente, documentación, diseño...).
 - Ayudan a evaluar la complejidad, la comprensibilidad y la capacidad de mantenimiento
 - relación indirecta con los atributos de calidad.

- Debe intentar y derivar una relación entre estas métricas y propiedades como la complejidad, la comprensibilidad y la capacidad de mantenimiento.
- “Evaluar su impacto requiere interpretación o inferencia. Ejemplo: una clase con complejidad ciclomática de 20 puede ser difícil de mantener, pero no siempre.”

“Métrica de producto vs de software en general: las métricas de software abarcan todo lo que puede medirse en un proyecto de software, no solo el código o el producto; Todo el proceso, el equipo, el producto y el proyecto.”

Métricas estáticas:

- **Fan-in** es una medida del número de funciones o métodos que llaman a otra función o método (por ejemplo, X). Un valor alto para fan-in significa que X está estrechamente acoplado al resto del diseño y los cambios a X tendrán amplios efectos de activación.
- **Fan-out** es el número de funciones a las que llama la función X. Un valor alto para el abanico sugiere que la complejidad general de X puede ser alta debido a la complejidad de la lógica de control necesaria para coordinar los componentes llamados.
- **Longitud del código:** Esta es una medida del tamaño de un programa. En general, cuanto mayor sea el tamaño del código de un componente, más complejo y propenso a errores será ese componente. Se ha demostrado que la longitud del código es una de las métricas más confiables para predecir la propensión a errores en los componentes
- **Complejidad ciclomática:** Esta es una medida de la complejidad de control de un programa. Esta complejidad de control puede estar relacionada con la comprensibilidad del programa
- **Longitud de los identificadores:** Esta es una medida de la longitud promedio de los identificadores (nombres de variables, clases, métodos, etc.) en un programa. Cuanto más largos sean los identificadores, mayor será la probabilidad de que sean significativos y, por lo tanto, más comprensible el programa.
- **Profundidad en los anidamientos:** Esta es una medida de la profundidad de anidamiento de sentencias if en un programa. Las declaraciones if profundamente anidadas son difíciles de entender y potencialmente propensas a errores.
- **Índice de niebla:** Esta es una medida de la longitud promedio de las palabras y oraciones en los documentos. Cuanto mayor sea el valor del índice de niebla de un documento, más difícil será comprenderlo.

Sorpresas con las métricas

- Reducir el número de fallas en un programa lleva a un mayor número de llamadas al servicio de asistencia
- El programa ahora se considera más confiable y, por lo tanto, tiene un mercado más amplio y diverso. **El porcentaje de usuarios que llaman a la mesa de ayuda puede haber disminuido pero el total puede aumentar** (hay más clientes en total).
- Un sistema más confiable se usa de una manera diferente a un sistema donde los usuarios resuelven las fallas (“Las mejoras técnicas pueden cambiar el patrón de uso

del software, lo que puede activar nuevos retos operativos; Los usuarios ahora confían más en el software y lo usan para tareas más críticas o complejas..”). Esto lleva a más llamadas a la mesa de ayuda

Puntos clave

- Las revisiones de los entregables del proceso del software involucran a un equipo de personas que verifican que se cumplan los estándares de calidad.
- En una inspección del programa o revisión por pares (peers), un pequeño equipo comprueba sistemáticamente el código.
- Leen el código en detalle y buscan posibles errores y omisiones.
- La medición de software puede utilizarse para recopilar datos sobre software y procesos de software.
- Las métricas de calidad del producto son particularmente útiles para resaltar componentes anómalos que pueden tener problemas de calidad.

Mejora del Proceso

- Muchas compañías de software han recurrido a la mejora de los procesos de software como una forma de mejorar la calidad de su software, reducir costos o acelerar sus procesos de desarrollo.
- La mejora de procesos significa comprender los procesos existentes y cambiar estos procesos para aumentar la calidad del producto y / o reducir los costos y el tiempo de desarrollo

Métodos para mejorar

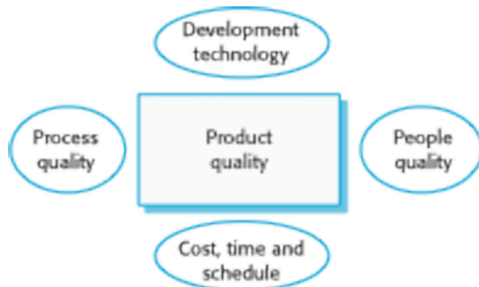
- El método de la madurez del proceso, que se enfoca en mejorar la gestión de procesos y proyectos e introducir buenas prácticas de ingeniería de software.
 - El nivel de madurez del proceso refleja la medida en que se han adoptado buenas prácticas técnicas y de gestión en los procesos de desarrollo de software de la organización.
- El método ágil, que se centra en el desarrollo iterativo y la reducción de gastos generales en el proceso de software.
 - Las características principales de los métodos ágiles son la entrega rápida de
- la funcionalidad y la capacidad de respuesta a los cambios en los requisitos del cliente

Proceso y calidad del producto

- La calidad del proceso y la calidad del producto están estrechamente relacionadas y los beneficios de mejora del proceso surgen debido a que la calidad del producto depende de su proceso de desarrollo.
- Generalmente se requiere un buen proceso para producir un buen producto.

- Para los productos manufacturados, el proceso es el principal determinante de la calidad.
- Para las actividades basadas en el diseño, también intervienen otros factores, especialmente las capacidades de los diseñadores

Factores que afectan a la calidad del producto software



Factores de calidad

- Para proyectos grandes con (desarrolladores con) capacidades "promedio", el proceso de desarrollo determina la calidad del producto.
- Para proyectos pequeños, las capacidades de los desarrolladores son el principal determinante.
- La tecnología de desarrollo es particularmente significativa para pequeños proyectos ("elección de herramientas que automaticen tareas, lenguajes...")
- En todos los casos, si se impone un programa poco realista, la calidad del producto se verá afectada.

Proceso de mejora de procesos

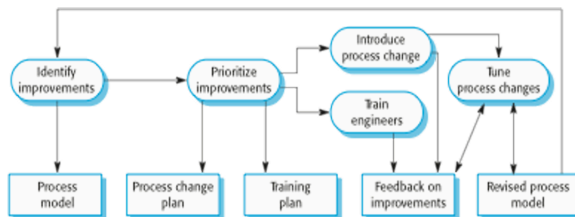
- No existe un proceso de software "ideal" o "estándar" que sea aplicable en todas las organizaciones o para todos los productos de software de un tipo en particular.
- Rara vez tendrá éxito en la introducción de mejoras en el proceso si simplemente intenta cambiar el proceso a uno que se utiliza en otros lugares.
- Siempre debe considerar el entorno y la cultura locales y cómo esto puede verse afectado por las propuestas de cambio de proceso.
- **Cada empresa debe desarrollar su propio proceso** en función de su tamaño, los antecedentes y las habilidades de su personal, el tipo de software que se está desarrollando, los requisitos del cliente y del mercado, y la cultura de la empresa

Cambio del proceso

- Implica hacer modificaciones a los procesos existentes.
- Esto puede implicar:
 - Introducción de nuevas prácticas, métodos o procesos;

- Cambiar el orden de las actividades de proceso;
- Introducción o eliminación de entregables;
- Introduciendo nuevos roles o responsabilidades.
- El cambio debe ser impulsado por objetivos medibles

Proceso de mejora de procesos



Etapas en el cambio del proceso

Identificación de mejora

- Esta etapa tiene que ver con el uso de los resultados del análisis de procesos para identificar formas de abordar problemas de calidad, programar cuellos de botella o ineficiencias de costos que se han identificado durante el análisis de procesos.

Mejora de prioridades

- Cuando se han identificado muchos cambios posibles, generalmente es imposible introducirlos todos a la vez, y hay que decidir cuáles son los más importantes.

Introducción al cambio de proceso

- La introducción del cambio de proceso significa establecer nuevos procedimientos, métodos y herramientas e integrarlos con otras actividades de proceso

Proceso de cambio de entrenamiento

- Sin capacitación, no es posible obtener todos los beneficios de los cambios en los procesos. Los ingenieros involucrados deben comprender los cambios que se han propuesto y cómo realizar los procesos nuevos y modificados.

Ajuste del cambio

- Los cambios de proceso propuestos nunca serán completamente efectivos tan pronto como se presenten. Necesita una fase de ajuste donde se puedan descubrir problemas menores y se pueden proponer e introducir modificaciones al proceso

Problemas en el cambio de proceso

- Resistencia al cambio
 - Los miembros del equipo o los gerentes de proyecto pueden resistir la introducción de cambios en el proceso y proponer razones por las cuales los cambios no funcionarán, o retrasar la introducción de cambios. En algunos casos, pueden obstruir deliberadamente cambios en el proceso e interpretar datos para mostrar la ineficacia del cambio de proceso propuesto.
- Cambio de persistencia

- Si bien es posible que inicialmente se introduzcan cambios en el proceso, es común que las innovaciones del proceso se descarten después de un breve período de tiempo y que los procesos vuelvan a su estado anterior
- Los gerentes de proyecto a menudo se resisten al cambio de proceso porque cualquier innovación tiene riesgos desconocidos asociados.
 - Los gerentes de proyecto son juzgados según si su proyecto produce software a tiempo y según el presupuesto. Es posible que prefieran un proceso ineficiente pero predecible a un proceso mejorado que tenga beneficios organizativos, pero que conlleve riesgos a corto plazo.
- Los ingenieros pueden resistir la introducción de nuevos procesos por razones similares, o porque consideran que estos procesos amenazan su profesionalismo.
 - Es decir, pueden sentir que el nuevo proceso predefinido les da menos discreción y no reconoce el valor de sus habilidades y experiencia (“ej: Si el proceso está muy definido, pueden sentir que no se valora su criterio o experiencia.”)

Persistencia del cambio

El problema de los cambios que se introducen y luego se descartan es un problema común. Los cambios pueden ser propuestos por un "evangelista" que cree firmemente que los cambios llevarán a una mejora. Él o ella puede trabajar duro para asegurar que los cambios sean efectivos y que se acepte el nuevo proceso.

Si el "evangelista" se va, entonces las personas involucradas pueden simplemente volver a las formas anteriores de hacer las cosas.

Cambiar la institucionalización es importante

- Esto significa que el cambio de proceso no depende de los individuos, sino que los cambios se convierten en parte de la práctica estándar en la empresa, con apoyo y capacitación en toda la empresa.

Calidad del proceso. Modelo CMMI. Capability Maturity Model Integration:

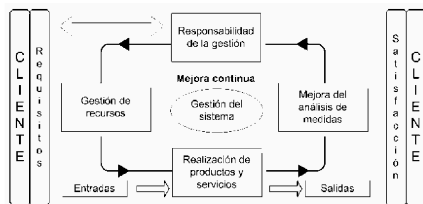
- Evaluación de CMM desarrollado por SEI.
- **No es un proceso de desarrollo de software.**
- Es una **guía que describe las características que hacen efectivo a un proceso.**
- Ideas aportadas utilizadas como conjunto de buenas prácticas.
- Modelo más prestigioso y difundido.
- Boeing, Nokia, Motorola, BMW, Intel, IBM, Nasa, Reuters.
- Es un **modelo de madurez**:
 - (“Es un esquema por niveles, donde cada nivel indica cuán avanzada está una organización en su forma de trabajar.”)
 - Conjunto de características que describen ciertos aspectos de equilibrio, experiencia y formalidad en una organización.
 - Se emplean como referencia para la comparación o mejora.

Calidad del proceso. ISO 9000

Conjunto de estándares para sistemas de calidad:

- Creado en 1980.
- ISO 9000. Fundamentos de los sistemas de gestión de calidad (define los principios básicos de la gestión de la calidad) y define los términos relacionados con la misma/vocabulario común a utilizar (proceso, producto..).
- ISO 9001. Especifica los requisitos de un sistema de gestión de la calidad dentro de una organización. ("especifica los requisitos concretos que debe cumplir una organización para establecer, mantener y mejorar un Sistema de Gestión de la Calidad (SGC).")
- Tienen como meta ayudar a las organizaciones a definir y mantener sistemas de calidad.
- No tiene nada que ver con los programas de aseguramiento de calidad (testing...)

Calidad del proceso. ISO 9000



ISO 9000. Principios:

- Orientación al cliente: entendiendo y satisfaciendo las necesidades del cliente.
- Liderazgo: Los líderes deben establecer unidad de propósito y dirección clara. Fomentan una cultura organizacional que motive a todos hacia los objetivos de calidad.
- Implicación de los empleados: La participación activa de los empleados mejora el rendimiento, la innovación y la responsabilidad
- Modelos de procesos. Las actividades deben verse como procesos interrelacionados que se gestionan como un sistema.???
- Modelo de gestión orientado a sistemas. Gestionar los procesos como un sistema cohesivo contribuye a la eficacia y eficiencia de la organización.??
- Mejora continua.
- Enfoque a la toma de decisiones objetiva.
- Relaciones con los proveedores mutuamente interdependientes.

Standards ISO 9001

- Un conjunto internacional de estándares que puede utilizarse como base para desarrollar sistemas de gestión de calidad.
- ISO 9001, el más general de estos estándares, se aplica a las **organizaciones que diseñan, desarrollan y mantienen productos, incluido el software.**
- El estándar ISO 9001 es un marco para el desarrollo de estándares de software.

- Establece **principios generales de calidad**, describe los procesos de calidad en general y establece los estándares y procedimientos de la organización que deben definirse. Estos deben ser documentados en un **manual de calidad organizacional**
 - Un organismo externo puede certificar que el manual de calidad de una organización cumple con las normas ISO 9000.
 - Algunos clientes requieren que los proveedores cuenten con la certificación ISO 9000, aunque aquí se reconoce cada vez más la necesidad de flexibilidad

Calidad del proceso. Otros modelos, estándares y especificaciones

Proceso software personal (PSP)

Watt Humphrey.

- Controlar, gestionar y mejorar forma de trabajo individual.
- Introducir disciplina en el proceso de desarrollo de software de cada individuo.

Principios:

- Un proceso definido y estructurado mejora la eficiencia del trabajo.
- El proceso personal definido debe alinearse con las habilidades y preferencias del individuo.
- Cada persona se debe involucrar en la definición de su proceso.
- El proceso de cada persona debe evolucionar según evolucionan sus habilidades y capacidades.
- La mejora continua del proceso se consigue si existe una retroalimentación rápida y explícita