



# TEORICO

## ▼ UNIDAD 1

### ▼ El proceso de Software

#### ▼ Definicion

1. Un conjunto de actividades, métodos, prácticas, y transformaciones que la gente usa para desarrollar o mantener software y sus productos asociados
2. Conjunto estructurado de actividades/tareas de cierta manera, para desarrollar un sistema de software. Estas actividades varían dependiendo de la organización y el tipo de sistema que debe desarrollarse

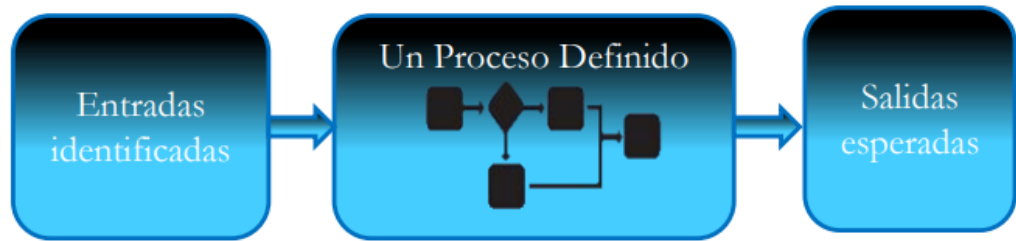
#### ▼ Compuesto por

- Personas
- Artefactos (salida)
- Herramientas (automatizan el proceso)
- Procedimientos

#### ▼ Tipos

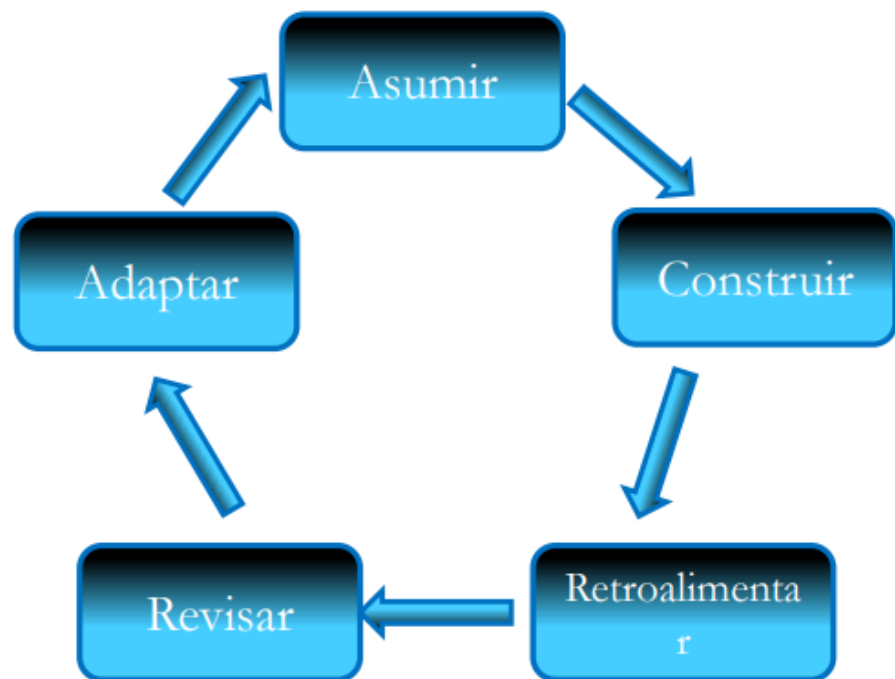
##### ▼ Definidos

- Esta inspirado en la linea de produccion y Asume que podemos repetir el mismo proceso una y otra vez, indefinidamente, y obtener los mismos resultados. En los procesos definidos, busco definir como realizar las actividades, lo cual nos va a ayudar en la administracion y control de la predictibilidad del proceso definido
- Contra —> parte con la base de que se predecir la salida, y esto en el software no siempre es asi



#### ▼ Empiricos

- Asume procesos complicados con variables cambiantes. Cuando se repite el proceso, se pueden llegar a obtener resultados diferentes. La administracion y control se hace atravez de inspecciones frecuentes y adaptaciones. Es por esto tambien que se dice que osn procesos en los que se busca ir aprendiendo a media que va avanzando el preceso.
- Patron de conocimiento en los procesos empiricos:



#### ▼ Ciclo de vida

- Se puede decir que es la manera u orden en la que ejecutamos las actividades
- Serie de pasos a través de los cuales el producto o proyecto progresa.

#### ▼ Ciclo de vida del proyecto

- Un ciclo de vida de un proyecto software es una representación de un proceso. Grafica una descripción del proceso desde una perspectiva particular

#### ▼ PROYECTO

- Es una instanciación del proceso
- Son únicos, cuentan con un inicio y un fin y tienen un objetivo que guía al proyecto, este no debe ser ambiguo y debe ser claro y alcanzable
- Están dirigidos a obtener resultados, y ellos se reflejan en los objetivos
- Tiene tareas que están relacionadas entre sí, son ejecutadas con recursos y son basadas en el esfuerzo asociado (se mide en horas dedicadas)

#### ▼ Equipo de proyecto

Un grupo de personas comprometidas en alcanzar un conjunto de objetivos de los cuales se sienten mutuamente responsables.

Se caracterizan por:

1. Diversos conocimientos y habilidades
2. Posibilidad de trabajar juntos efectivamente
3. Usualmente es un grupo pequeño
4. Tienen sentido de responsabilidad como una unidad

## ▼ Plan de proyecto

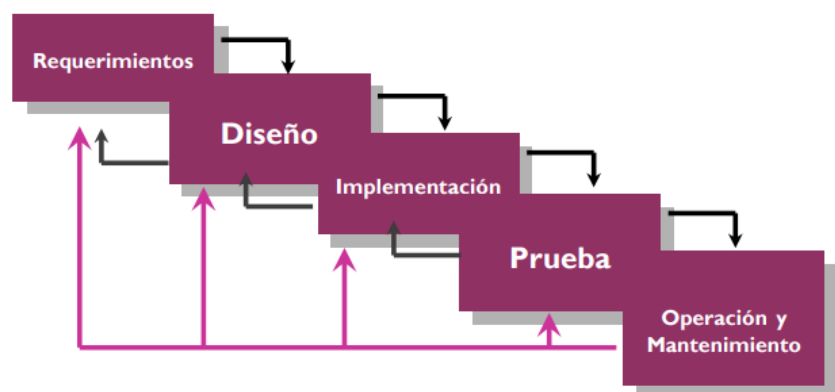
- Se puede decir que una hoja de ruta a seguir para alcanzar el objetivo cumpliendo de la mejor manera posible la triple restriccion. Nos dice el :
  1. Que es lo que hacemos? —> alcance del proyecto
  2. Cuando lo hacemos? —> tiempo
  3. Como lo hacemos? —> actividades / recursos
  4. Quien lo va a hacer? —> roles

Implica

### 1. Definición del Alcance

- a. Del producto (se mide contra la especificacion de requerimientos) —> Son todas las características que pueden incluirse en un producto o servicio
- b. Del proyecto (se mide contra el plan del proyecto) —> Es todo el trabajo y solo el trabajo que debe hacerse para entregar el producto o servicio con todas las características y funciones especificadas

### 2. Definición de Proceso y Ciclo de Vida

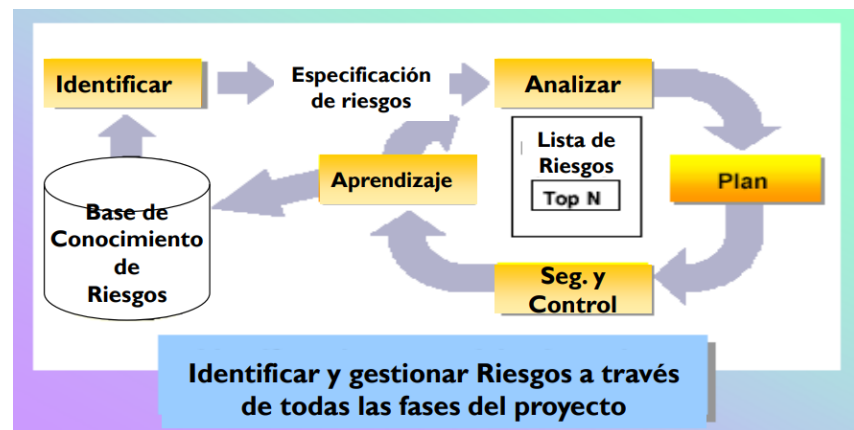


### 3. Estimación

- a. Tamaño
- b. Esfuerzo
- c. Calendario
- d. Costo
- e. Recursos Criticos

#### 4. Gestión de Riesgos

Evento que podría comprometer el éxito del proyecto



- 5. Asignación de Recursos
- 6. Programación de Proyectos
- 7. Definición de Controles
- 8. Definición de Métricas
  - a. De proceso
  - b. De proyecto
  - c. De producto

#### ▼ Ciclo de vida del producto

#### ▼ Clasificación

- Secuencial
- Iterativo

- Recursivo

#### ▼ Administracion de proyectos

- Es la administración de proyectos es la aplicación de conocimientos, habilidades, herramientas y técnicas a las actividades del proyecto para satisfacer los requerimientos del proyecto.
- Basicamente se podria decir que es tener el trabajo hecho, en tiempo, con el presupuesto acordado y habiendo satisfecho las especificaciones o requerimientos.

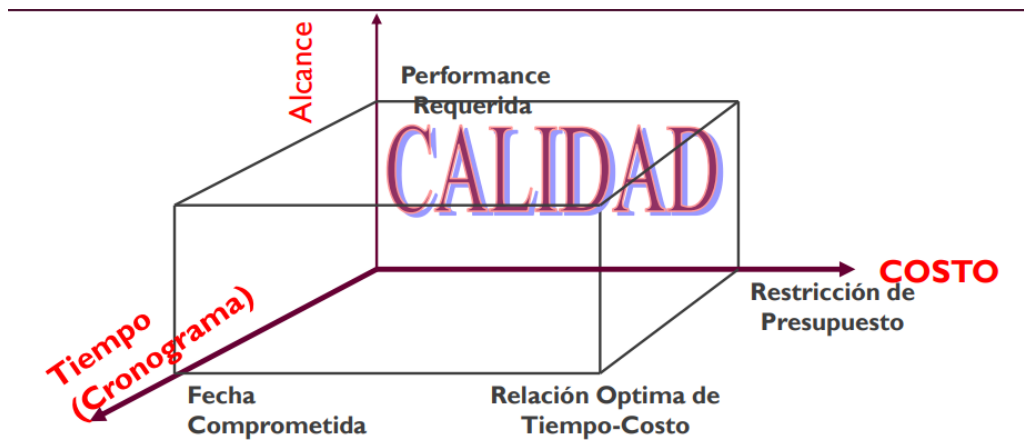
Incluye

1. Identificar los requerimientos
2. Establecer objetivos claros y alcanzables
3. Adaptar las especificaciones, planes y el enfoque a los diferentes intereses de los involucrados

#### ▼ Triple restriccion

1. Alcance —> Objetivos de proyecto
2. Costo —> cuánto debería costar?
3. Tiempo —> cuánto tiempo debería llevar completarlo?

El balance de estos tres factores afecta directamente la calidad del proyecto y es responsabilidad del Líder de proyecto balancear estos tres objetivos. Se dice que “proyectos de alta calidad entregan el producto requerido, el servicio o resultado, satisfaciendo los objetivos en el tiempo estipulado y con el presupuesto planificado.”



### ▼ Filosofía Ágil Manifiesto Ágil

#### ▼ Requerimientos Agiles/Desarrollo agil de software

Busca aliviar el proceso de desarrollo y tienen una fuerte relación con los procesos empíricos. En cuanto al manifiesto ágil, se dice que es un acuerdo firmado y que cuenta con 4 valores ágiles y 12 principios. Los cuales nos guían a una forma de pensar en términos organizacionales.

#### ▼ Los 12 principios

1. Satisfacer al cliente
2. Aceptar que los requerimientos cambien
3. Entregar software funcional frecuentemente
4. Todos deben trabajar juntos día a día
5. Individuos motivados
6. Conversaciones cara a cara
7. La principal medida del éxito es el software funcionando
8. Desarrollo sostenible
9. Mejorar la agilidad teniendo en cuenta el buen diseño y la excelencia técnica
10. La simplicidad es esencial

11. Equipos auto-organizados

12. Ajustar comportamiento segun la efectividad

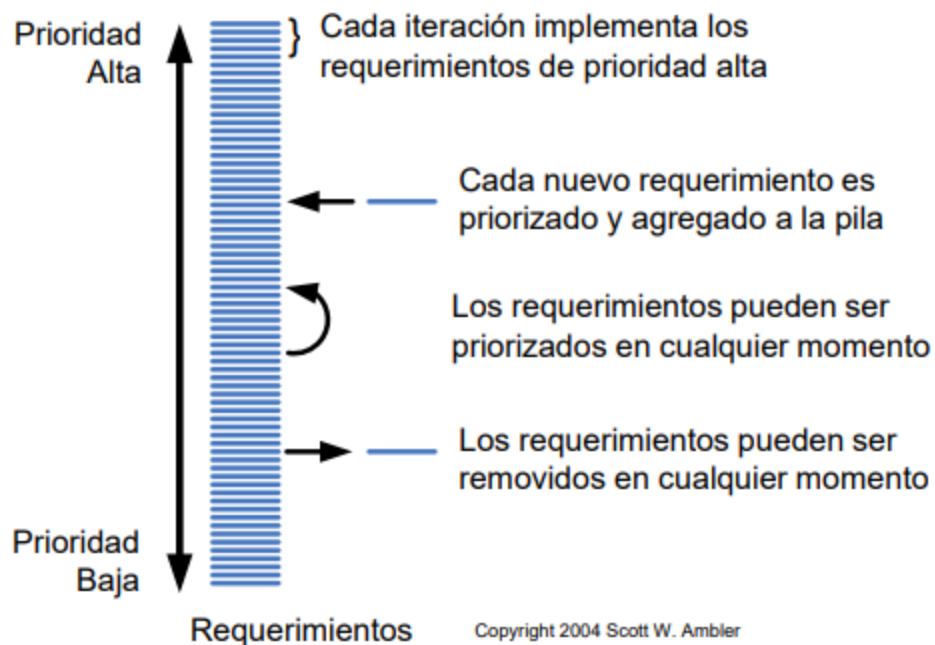
▼ Agil

- Es una ideología con un conjunto definido de principios que guían el desarrollo del producto
- Balance entre ningún proceso y demasiado proceso.

▼ Tecnicas

1. Dailys
2. Prioritized backlogs
3. Short iterations
4. Retrospectives
5. Iteration planning

▼ Gestión agil de los requerimientos de software



Debemos plantear cual es el valor, de los requerimientos para el cliente y de esta forma no realizar cosas inesesarias. Para esto tambien es



necesario pensar en el objetivo del cliente y de esta forma entregarle un software que le sirva. En otras palabras podríamos decir que debemos priorizar los requerimientos según el valor que tienen para el cliente.

VALOR = lo asociamos a la utilidad, beneficio o satisfacción que le ofrecemos a los usuarios finales por cada funcionalidad completa que les entregamos.

#### ▼ Tipos de requerimientos

- Requerimiento de Negocio
- Requerimiento de Usuario
- Requerimiento Funcional
- Requerimiento No funcional
- Requerimiento de Implementación

#### ▼ Técnicas

##### ▼ User Story

Es una definición de una funcionalidad que el software debe cumplir, expresada en forma de historia y desde la perspectiva del usuario.

##### ▼ PARTES


- Tarjeta
- Conversación
- Confirmación


#### ▼ UNIDAD 2


##### ▼ Estimaciones de software


- No son precisas → tienen un componente de incertidumbre
- Utilizamos herramientas para disminuir esa incertidumbre
- Aplicación de técnicas para predecir lo que pasará
- Lo más beneficioso de la estimación es el proceso de hacerlo

- Puede servir como una respuesta temprana sobre si el trabajo planificado es factible o no
- Puede servir como una protección para el equipo → si el plan es distinto a lo que se estimó
- Estimar NO es lo mismo que planear
  - Las estimaciones son la base de los planes, pero los planes no tienen que ser lo mismo que lo estimado
  - Para armar un plan debemos contar con estas estimaciones
  - Si al planificar ya tengo las estimaciones, disminuyo la incertidumbre
  - Al planificar tenemos más en cuenta aspectos del negocio
  - El compromiso se logra a través del plan
  - A veces en el plan definimos más tiempo que lo estimado por conveniencia de negocio

 Si las estimaciones se utilizan como compromisos son muy peligrosas y perjudiciales para cualquier organización.

 Lo más beneficioso en las estimaciones es el “proceso de hacerlas”.

 La estimación podría servir como una gran respuesta temprana sobre si el trabajo planificado es factible o no.

 La estimación puede servir como una gran protección para el equipo.

- ¿Para qué sirve estimar?
  - Hacer que el compromiso que asumamos se acerque lo más posible a la realidad
- Al comienzo del proyecto las estimaciones son más distintas a la realidad que al final del proyecto
- Debemos ir analizando las estimaciones durante todo el proyecto
- Causas de los errores de estimación

- No conocer el tamaño del sistema
- Subestimar el esfuerzo
- Requerimientos faltantes
- Omitir actividades necesarias
- No tener en cuenta los riesgos
- Pensar que algo es muy pesimista y cambiarlo
- Orden de estimación
  1. Tamaño del sistema
    - Cantidad de casos de uso por complejidad
    - Cantidad de user stories
    - Cantidad de features
  2. Esfuerzo
    - Cantidad de horas lineales que me llevarán hacer el trabajo necesario
    - Tener en cuenta
      - Días feriados
      - Dependencias entre actividades
      - Capacidad del equipo
- Uso de buffer → se realizan estimaciones necesarias por la existencia de un riesgo
- ▼ Técnicas de estimación
  - ▼ Contar
    - Features
    - Stories
    - Casos de uso
  - ▼ Basados en la experiencia

#### ▼ Datos históricos

- Proyectos similares
- Base de conocimiento
- Es difícil estimar con datos porque todas las situaciones son distintas

#### ▼ Juicio experto

##### ▼ Puro

- Un experto estudia las especificaciones y hace su estimación
- Se basa en los conocimientos de la persona
- Si desaparece el experto, se deja de estimar
- Es demasiado subjetivo
- Es el más utilizado en la práctica
- Método de “optimista, pesimista y habitual”
- Fórmula =  $(o + 4h + p)/6 \rightarrow$  da algo intermedio

##### ▼ Wideband Delphi

- Un grupo de personas son informadas y tratan de adivinar lo que costará el desarrollo tanto en esfuerzo, como en duración
- No es un único experto
- Conocen el negocio
- Normalmente son las personas que forman parte del equipo de desarrollo
- La estimación en grupo es mejor que la individual

##### ▼ Pasos

1. Se dan las especificaciones a un grupo
2. Se discute del producto y estimación

3. Remiten sus estimaciones individuales al coordinador
4. Cada uno tiene un feedback de su estimación y las ajenas pero anónimamente
5. Se reúnen de nuevo para discutir
6. Cada uno revisa su propia estimación y la reenvía
7. Se repite el proceso hasta que convergen las estimaciones

▼ Basados exclusivamente en los recursos

▼ Basado exclusivamente en el mercado

- Buscar proyectos del mismo negocio

▼ Basados en los componentes del producto o en el proceso de desarrollo

▼ Métodos algorítmicos

▼ Estimaciones ágiles

▼ Story points (SP)

- Es una unidad de medida específica (del equipo) de, complejidad, riesgo y esfuerzo
- No es una medida basada en el tiempo
- No es una medida absoluta
- Medida relativa
- Incluye:
  - Tamaño/Complejidad
  - Esfuerzo que le lleva implementar esa feature a ese equipo particular
  - Riesgo
- Tamaño  $\neq$  Esfuerzo
- Se puede usar la serie de fibonacci

- La complejidad tiende a incrementarse de manera exponencial

#### ▼ Estimación relativa

- Las personas no saben estimar en términos absolutos
- Comparar es más rápido
- Se obtiene una mejor dinámica grupal y pensamiento de equipo

#### ▼ Poker estimation

- Derivado del Wideband Delphi
- Las personas más competentes en resolver una tarea deben ser quienes las estimen
- Se van comparando las features y las ordenamos
- Cada uno estima y dan las razones
- Tiene la última palabra el que da la feature

#### ▼ Escala de estimaciones (agregar)

#### ▼ Velocidad

- Métrica del progreso
- ¿Cuántos story points completamos en una iteración?
- Se cuentan los story points de las Users Storys que están completas, no parcialmente completas.
- La velocidad corrige los errores de estimación

#### ▼ Gestión de Productos

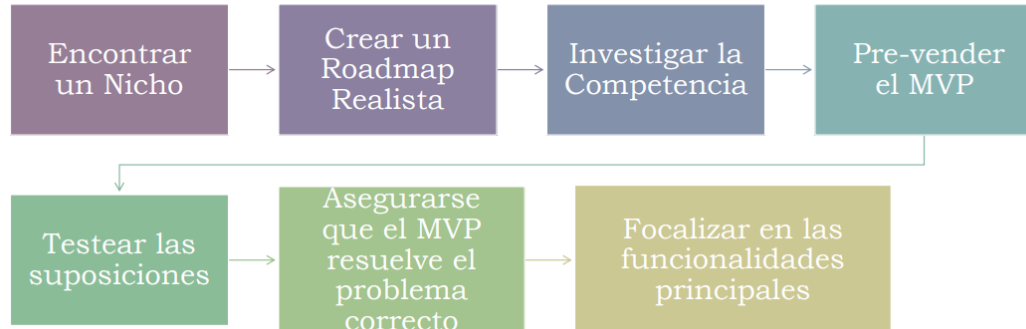
¿Por qué creamos productos?

- Para satisfacer a los clientes
- Para tener muchos usuarios logeados
- Realizar una gran vision y cambiar el mundo

#### ▼ Evolución de un producto de software

1. Comprender que un producto nuevo tiene una hipótesis de valor único, una propuesta de valor única (UVP)
2. Debemos crear un producto mínimo viable (MVP) para probar su hipótesis.
3. Luego debemos ver mediante nuestro MVP el mercado al cual vamos a “ingresar”
  - Se muestra el caso en el que cada cliente potencial con el que interactúas te dice "Esto es genial, pero para poder usarlo necesito X" y X es diferente para cada cliente / usuario. Esto muestra que aún no se encuentra un mercado para el producto
  - Si por el contrario ves cada vez más respuestas apuntando al “MISMO X” entonces tiene sentido revisar la hipótesis de Cliente/Problema/Solución. Se podría decir que estamos construyendo un MVP2
4. Luego seguimos con la construcción de nuestra característica mínima caracterizable (MMF). Cuyo objetivo es aportar valor. Debemos tener en cuenta la importancia de dividir una característica grande en MMF, que es principalmente el tiempo de comercialización (Time to market) y la capacidad de aportar valor en muchas áreas.
5. Como conclusión podríamos decir que la forma de afrontarlo es crear una función "pionera": MVF (Característica Mínima Viable). La característica mínima que aún puede ser viable para uso real y aprendizaje de los usuarios reales. Si la MVF resulta exitosa (hit gold), puede desarrollar más MMF en esa área para tomar ventaja (si eso tiene sentido). Si no es así, puede cambiar a otro enfoque hacia esa área de características, o en algún momento buscar una ruta de crecimiento alternativa. Esencialmente, el MVF es una versión mini del MVP.

# Preparar un MVP



## ▼ Gestión de Configuración de Software}

### ▼ Contexto

Decimos que un software va a estar compuesto por PERSONAS, estas personas se van a incorporar a un PROYECTO al cual se le adaptan PROCESOS que para poder ser realizados se automatizan con HERRAMIENTAS. Como resultado de este proyecto obtenemos el PRODUCTO.





También podemos decir que el software es un conjunto de:

- Programas
- Procedimientos
- Reglas
- Documentación
- Datos

El software evoluciona debido a múltiples factores, como por ejemplo los cambios de requerimientos.

La gestión de configuraciones nos permite que el software evolucione de manera organizada. El objetivo de la gestión de la configuración de software es mantener la integridad del producto de software que estamos construyendo:

Los cambios tienen su origen en:

- Cambios de negocio y nuevos requerimientos
- Soporte de cambios de productos asociados
- Reorganización de las prioridades
- Cambios en el presupuesto
- Defectos a corregir
- Oportunidades de mejora

## ▼ SCM

### ▼ Definición

Es una disciplina de soporte que ayuda a mantener la integridad del producto de software a lo largo de todo su ciclo de vida. Es una actividad “paragüas”, transversal a todo el proyecto. Lo que busca es que mediante el control de calidad de proceso, control de calidad del producto y pruebas de software, lograr un aseguramiento de calidad del software

- Una disciplina que aplica dirección y monitoreo administrativo y técnico a: identificar y documentar las características funcionales y

técnicas de los ítems de configuración, controlar los cambios de esas características, registrar y reportar los cambios y su estado de implementación y verificar correspondencia con los requerimientos.

▼ Integridad del producto

1. Satisfacer las necesidades del usuario, requerimientos funcionales y no funcionales
2. Puede ser fácil y completamente rastreado durante su ciclo de vida  
→ Versionado del producto y sus features. Esto puede traer algunos de sus problemas, por ejemplo:
  - a. Pérdidas de componentes
  - b. Pérdidas de cambios
  - c. Sincronía fuente-objeto-ejecutable
  - d. Regresión de fallas
  - e. Doble mantenimiento
  - f. Superposición de cambios
3. Satisface criterios de performance
4. Cumple con sus expectativas de costo

▼ Conceptos claves

▼ Item de Configuración de Software

- son todos y cada uno de los artefactos que forman parte del producto o del proyecto, que pueden sufrir cambios o necesitan ser compartidos entre los miembros del equipo y sobre los cuales necesitamos conocer su estado y evolución
  - Plan de CM
  - Propuestas de cambio
  - Visión
  - Riesgos
  - Plan de desarrollo

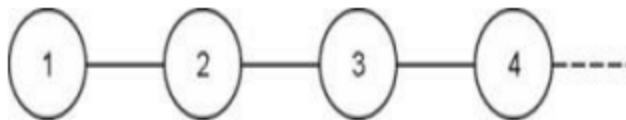
- Prototipos de interfaz
- Guía de estilo
- Manual de usuario
- Requerimientos
- Plan de calidad
- Arquitectura
- Plan de integración
- Planes de iteración

#### ▼ Versión

Una versión se define, desde el punto de vista de la evolución como la forma particular de un artefacto en un instante o contexto dado

El control de versiones se refiere a la evolución de un único ítem

La evolución puede representarse gráficamente en forma de grafo.

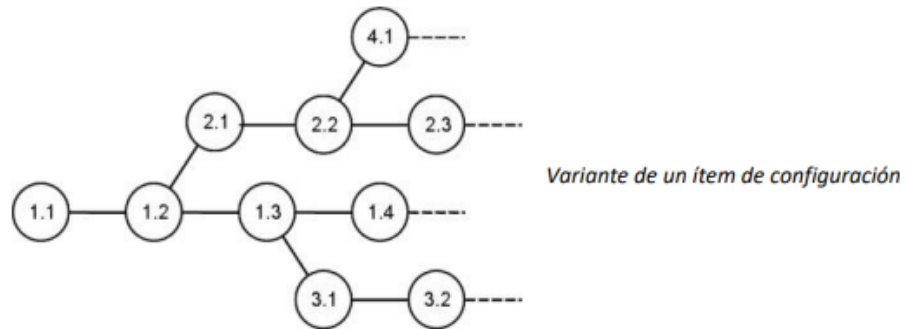


#### ▼ Variante

Una variante es una versión de un ítem de configuración (o de la configuración) que evoluciona por separado

Las variantes representan configuraciones alternativas.

Un producto de software puede adoptar diferentes variaciones. Ej: configuración para iPhone y para Android



#### ▼ La configuración del software

Es un conjunto de ítems de configuración con su correspondiente versión en un momento determinado

#### ▼ Repositorio

Un repositorio es donde tenemos la información que contiene los ítems de configuración.

##### ▼ Centralizado

Un servidor contiene todos los archivos con sus versiones

Falla el servidor y "estamos al horno"

##### ▼ Descentralizado

Cada cliente tiene una copia exactamente igual del repositorio completo.

Si un servidor falla sólo es cuestión de "copiar y pegar"

#### ▼ Identificación de Línea base

La línea base es una configuración que ha sido revisada formalmente y sobre la que se ha llegado a un acuerdo.

Sirve como base para desarrollos posteriores y puede cambiarse sólo a través de un procedimiento formal de control de cambios

Permiten ir atrás en el tiempo y reproducir el entorno de desarrollo en un momento dado el proyecto.

#### ▼ Ramas

- Existe una rama principal (trunk, master)

- Su utilización sirve para bifurcar el desarrollo, experimentar con diferentes configuraciones y demás

## ▼ Actividades fundamentales de la gestión de configuración de software

### ▼ 1) Identificar ítems de configuración

Identificación unívoca de cada ítem de configuración

- Unicidad para identificar el TIPO de ítem de configuración
- En algunos ítems es conveniente incluir, por ejemplo, la fecha

Definición de convenciones y reglas de nombrado

Definición de la estructura del repositorio

Ubicación dentro de la estructura del repositorio

### ▼ 2) Control de cambios

Para poder introducir cambios en una línea base, necesito que esos cambios sean validados y/o aceptados por un comité de control de cambios.

Un comité de control de cambios está integrado por diferentes roles, de mínima algún referente de cada una de las áreas interesadas.

Implica la evaluación del riesgo o impacto del cambio

### ▼ 3) Auditorías de gestión de configuración

Dos tipos:

- Auditoría funcional de configuración: valida que el producto que se está construyendo sea consistente con los requerimientos identificados. Se utiliza una herramienta llamada matriz de rastreabilidad para trazar cuáles son los modelos que se involucran en cada requerimiento hasta su cumplimiento.

- Auditoría de configuración física: verifica que todo lo que definimos para la gestión de configuración en el plan de gestión de configuración se esté siguiendo de la manera adecuada

Tienen que ser realizadas por alguien externo al proyecto, no necesariamente ajena a la organización.

- Validación se corresponde con la auditoría funcional. Construir el producto correcto
- Verificación se corresponde con la auditoría de gestión física. Construir el producto correctamente

#### ▼ 4) Informes de estado

Se ocupa de mantener los registros de la evolución del sistema e incluye reportes de rastreabilidad de todos los cambios realizados a las líneas base durante el ciclo de vida.

#### ▼ Plan de gestión de configuración

El plan incluye:

- Reglas de nombrado
- Herramientas a utilizar para SCM
- Roles e integrantes del comité
- Procedimiento formal de cambios
- Plantillas de formularios
- Procesos de auditoría

El plan de gestión de configuración es parte del plan de proyecto y para ser útil al proyecto debe ser actualizado y revisado constantemente.

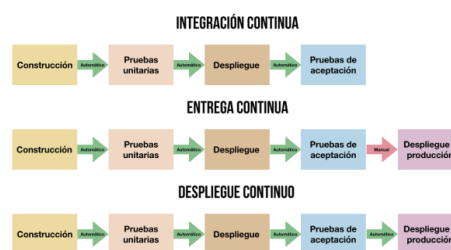
#### ▼ Evolución de la Gestión de Configuración de Software

##### ▼ Continuo

- Integración continua: permite a raíz de una automatización del testing que los build que se van construyendo se vayan incorporando al producto de software. Se ejecutan pruebas de

manera automatizada que en caso de arrojar un resultado exitoso la integramos a nuestro producto.

- Entrega continua: Es una mejora que se incorpora sobre la integración continua que deja lista nuestra versión para su despliegue en producción. El despliegue final se realiza de manera manual.
- Despliegue continuo: Funciona como la entrega continua pero el despliegue ya es automático



#### ▼ Gestión de la configuración en Metodologías Ágiles

- Hace seguimiento y coordina el desarrollo en lugar de controlar a los desarrolladores.
- Busca esforzarse por ser transparente y "sin fricción", automatizando tanto como sea posible.
- Eliminar el desperdicio - no agregar nada más que valor.
- Feedback continuo y visible sobre calidad, estabilidad e integridad