RESUMEN ISW 1ER PARCIAL

TEÓRICO

Software

- Es un conjunto de programas, código y la documentación (para desarrolladores y usuarios) que lo acompaña
- Puede contener varios programas independientes dentro suyo
- Incluye la configuración del SW
- Su calidad depende del equipo y el proyecto de desarrollo
- Puede ser de uso general o extremadamente específico
- Hay ciertos factores para el desarrollo de un SW de calidad (relación del usuario con el equipo, definición clara de los requerimientos, buen planeamiento, equipo competente)
- Y también hay ciertos factores que resultan en un SW de mala calidad (requerimientos incompletos, falta de relación con el usuario, falta de recursos, expectativas fallidas)
- Se clasifica en:
  - De sistema: es un sistema (conjunto) de software, como los sistemas operativos.
  - Utilitarios: proveen una utilidad/funcionalidad específica, pero no son construidos teniendo en cuenta una necesidad especifica de negocio, no son personalizados para cada cliente (Word, Excel)
  - De aplicación
- No se debe comparar al SW con otros productos de manufacturacion:
  - No se produce en masa, cada SW es muy específico, no se encuentran 2 exactamente iguales
  - No se gasta con su uso
  - No es predecible
  - No todos sus fallos son errores de producción
  - No se aplican leyes físicas a los mismos

Ingeniería de SW

- Se basa en la construcción de software de múltiples versiones, por parte de múltiples personas
- Es una disciplina de la ingeniería
- Se preocupa de todos los aspectos de la producción de un software, desde las primeras etapas de la especificación hasta el mantenimiento del sistema después que se pone en operación.
- Es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software
- Cubre también el estudio de estos enfoques, es decir, el estudio de las aplicaciones de la ingeniería al software.
- Hace uso de 3 disciplinas:
  - Disciplinas técnicas: se basa en la construcción del producto, desde un enfoque técnico, siguiendo un plan con etapas claras y definidas (requerimientos, análisis, diseño, construcción, prueba, despliegue)
  - Disciplinas de Gestión: se basa en el aspecto de planificación, control y monitoreo de la ingeniería de SW (planificación, monitoreo y control de proyectos)
  - Disciplinas de Soporte: se basa en la configuración del sistema, del aseguramiento de su correcta funcionalidad y una alta calidad (SCM, métricas y calidad)

SEBOK (Systems Engineering Body of Knowledge)

- Es una autoridad que guía el desarrollo de SW a través de un conjunto de conocimientos interrelacionados (15 áreas) necesarios para su aplicación.
- Es un cuerpo de conocimiento de la ISW

Proceso de Software

- Es el conjunto estructurado de las actividades que se llevan a cabo para el desarrollo del producto y componentes asociados, incluyendo los recursos, requerimientos, métodos, protocolos, información y personas relacionadas, definiendo el orden de estas etapas a través de un ciclo de vida del proyecto
- Tiene un objetivo asociado
- En cada paso se define que personas/actividades/herramientas tendrá asociado
- Varía con cada proyecto, organización y/o equipo de trabajo
- Si se planea controlar el proceso, éste debe estar completamente modelado/detallado
- Principalmente se divide en dos tipos de procesos: procesos definidos y procesos empíricos

Proceso Definidos

- También se conocen como proceso tradicional, y se basa en las cadenas de producción
- Esta completamente detallado el plan de acción, de manera que se podría repetir nuevamente el proyecto y obtener los mismos resultados
- Es un proceso completamente controlado y planificado, su predictibilidad esta dada desde el principio, y se actuá en base a un plan y procedimientos rigurosos y detallados
- Sus etapas están claramente definidas, y se apega a un plan fijo y definido completamente
- Nada se deja a la suerte o a la subjetividad de cada persona, se actuá en base a estándares fijos
- La administración y el control se aseguran de que se este siguiendo correctamente el plan
- Se basa en un enfoque definido y secuencial
- Se determina desde el principio las etapas/fases por las que pasara el proyecto para llegar a obtener el producto final
- Gran planificación que demanda mucho tiempo la principio
- Especial atención a la eficiencia del proceso
- Se sigue una distribución de taras y un cronograma estrictos
- Se documentan todos los aspectos del proyecto, se cuenta con alta documentación al finalizar
- Busca todos los requerimientos del proyecto al principio del mismo, y presenta dificultad para alterarlos/modificarlos una vez avanzado el proyecto
- Solo se pasa a la etapa siguiente si se cumplió correctamente con la anterior
- Alto nivel de control/revisión
- Se recomienda para proyectos grandes, en los que se tiene un conocimiento pleno y completo de los requerimientos del cliente

Procesos empíricos

- Se basa en el empirismo, el aprendizaje a través de la experiencia
- Se basa en la experiencia del equipo de desarrollo
- Ante el mismo proyecto, dos equipos pueden obtener resultado completamente distintos
- Es muy especifico para cada equipo de desarrollo en cada proyecto, y no es posible repetirlo en otro proyecto/equipo
- Las tomas de decisiones se basan en lo que ya se conoce
- Las decisiones tomadas en base a la experiencia son a su vez la base de conocimientos para las próximas decisiones
- Se tiene mucha incertidumbre en el comienzo del proyecto, que va disminuyendo a medida que el equipo desarrolla el producto
- Se comienza la captura de requerimientos con la practica del equipo a medida que se avanza
- Su documentación es poca, y no es el objetivo principal del equipo documentar cada proceso/resultado/actividad
- Presenta facilidad para la modificación/agregación de requerimientos en cualquier etapa
- Es un proceso adaptativo, ya que esta predispuesto a lo desconocido y a enfrentar problemas nuevos
- Lleva más tiempo empezar a trabajar concretamente en el desarrollo

- No se tiene una definición muy rigurosa/exigente de su plan de acción, en cambio sus objetivos y etapas tienen una naturaleza mas general (se tiene en claro que se quiere lograr, pero no como se llegará a lograrlo)
- Permite un desarrollo iterativo e incremental (se presentan versiones del producto de manera iterativa, cada vez añadiendo mas funcionalidad)
- Se suele aplicar en equipos de tamaño chico, ya que el grado de libertad de trabajo requiere una comunicación fluida y precisa
- Requiere un nivel mayor de creatividad, y el equipo debe trabajar cómodo entre sus integrantes, conociéndose y comunicándose de manera eficiente
- En el principio, a un equipo de desarrollo le costará aplicar este tipo de procesos, pero con el paso del tiempo se familiarizara con el trabajo de sus integrantes, volviéndose mas ameno y fácil de llevar a cabo
- Actualmente son una tendencia en el desarrollo de software, ya que presentan muy buenos resultados eficientes
- Se basa en tres ideales:
  - Transparencia: el proyecto es visible para todo el equipo en cualquier momento, ya que cualquier integrante del mismo puede conocer cualquier detalle de cualquier parte del proceso, de manera que se entienda y se naturalice el trabajo de todo el equipo en común
  - Inspección: el equipo debe realizar inspecciones/controles regularmente sobre cada componente de trabajo, identificando que se corresponda con los objetivos deseados y planificar los pasos a seguir de acuerdo a los mismos (la frecuencia de control debe ser suficiente para detectar problemas antes de que se vuelvan mas complejos pero que no sea una molestia o perdida de recursos sin valor en el equipo)
  - Adaptabilidad: hace referencia a la capacidad de cambio, adaptación y reorganización que tiene este tipo de proceso, adoptando cambios mayores o menores sin consecuencias graves, pudiendo reducir el riesgo que tiene el equipo de que una sorpresa lo afecte de manera negativa

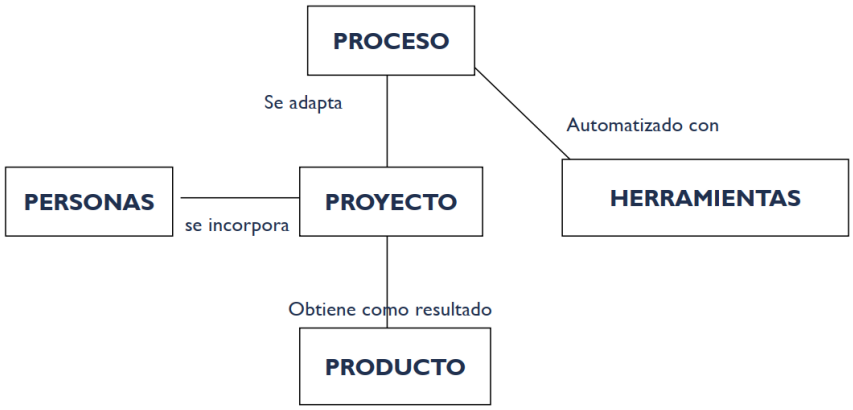
Ciclo de vida

- Es una representación, a través de una perspectiva/punto de vista particular, del proceso del desarrollo del producto
- Grafica una descripción del proceso desde una perspectiva particular.
- Especifica las etapas del proceso (PE: requerimientos, especificación, diseño) y el orden en el que se llevaran a cabo
- Establece los procedimientos por los que se pasa para desarrollar el producto
- Define como se organiza el proyecto, las fases o etapas del desarrollo del mismo y el orden de las mismas, desde el comienzo al final
- Fases/etapas:
  - Una fase es un conjunto de actividades relacionadas con un objetivo en el desarrollo del proyecto.
  - Se construye agrupando tareas que pueden compartir un tramo determinado del tiempo de vida de un proyecto.
  - En esencia, son un conjunto de tareas relacionadas dentro del proyecto que tienen un objetivo en común y se pueden considerar como un micro-proyecto dentro del proyecto.
- Es una secuencia bien estructurada y definida de los pasos del proceso de desarrollo de SW
- Definen, en cada etapa:
  - Que actividades se hacen
  - Quien debe hacerlas
  - Que objetivo se busca cumplir
  - Que procesos/herramientas/métodos se utilizan
  - Como controlar/aprobar cada fase
  - El orden de cada etapa
- Hay distintos tipos de ciclos de vidas, que son distintas formas de representar el desarrollo del proyecto:
  - Secuencial (pasos definidos, en orden lineal, una etapa necesita que se termine la anterior, errores se arrastran, menos riesgo) → Cascada

- Iterativo/Incremental (desarrollo a través de iteraciones con valor incremental, entregables en cada iteración) → Espiral
- Recursivo

Proyectos

- Son desarrollos temporales de productos de SW, que involucran un proceso de trabajo, un ciclo de vida que define sus etapas, y un conjunto de personas, herramientas, actividades y procedimientos que tienen como objetivo la construcción de un SW de calidad y funcional
- Cada proyecto es único, no se repite (puede repetirse el proyecto, pero se dará de manera distinta)
- En sus primeros pasos se debe definir el objetivo, ya que eso condicionara todos los aspectos del proyecto
- Otro de los pasos principales es definir con que tipo de proceso y ciclo de vida se trabajara (esta decisión debe ser acertada acorde al producto a desarrollar, ya que con el paso del tiempo, el cambio en estas decisiones se volverá mas costoso)
- Se basan en la implementación de un proceso (del tipo que sea) para cumplir el objetivo
- Organiza a los recursos con que cuenta
- Se orientan a los objetivos, que deben ser claros realistas, alcanzables y comprobables
- Deben estar definidos de manera correcta y clara, para poder llevar a cabo un control del avance o completitud del mismo
- Si no tiene fin, entonces no es un proyecto (porque estos se basan en los objetivos, en el fin)
- La interacción de personas, recursos y componentes en un proyecto generan mas complejidad, capacidad e información respecto al mismo
- Cada persona tiene un rol dentro del proyecto
- Tiene un líder que asegura que todo se este llevando a cabo correctamente para llegar al objetivo



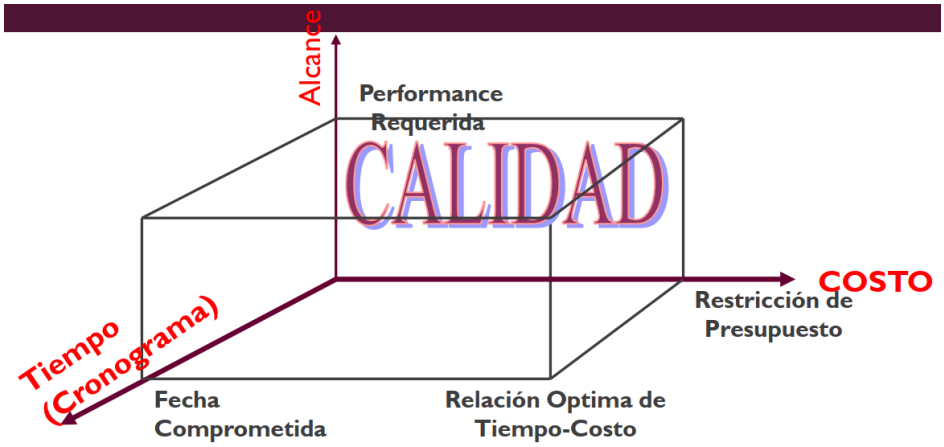
- Las personas son el recurso mas valioso del proyecto, y tiene roles con tareas definidas
- El proyecto se hace para desarrollar (y finalmente obtener) un producto de calidad y funcional
- El proyecto se adapta a un tipo de proceso elegido
- El proceso hace uso de herramientas especificas

El producto de software, que es el objetivo del proyecto de desarrollo, es un conjunto de código, documentación y componentes asociados (librerías, APIS, manuales, casos de pruebas, etc) que es completamente funcional y se entrega al cliente para su uso (aunque contiene partes o elementos que el cliente no conozca o vea directamente)

La administración del proyecto se basa en aplicar conocimientos, habilidades, herramientas y técnicas a las distintas actividades del proyecto, de manera que se satisfagan sus requerimientos
Incluye identificar los requerimientos, establecer (de manera correcta y clara) los objetivos y adaptar los planes/enfoques elegidos a los intereses del cliente

Triple constraint / triple restricción

- Es una restricción que se aplica a los proyectos de desarrollo de SW
- Se basa en tres factores que deben tener un balance entre si (siendo esto responsabilidad del líder de proyecto) para obtener un producto de calidad
- Sus tres factores son:
  - Objetivos/alcance: los objetivos definidos y prometidos al cliente, que se relacionan con la funcionalidad que ese SW brindará.
    - En las metodologías tradicionales se tiene definido desde el principio el alcance del producto, por lo que no era negociable.
    - De esa manera, se pasaban de tiempo y costo.
    - Por eso surgen las metodologías ágiles (que tienen como factor fijo el tiempo, negociando costo y alcance del SW)
  - Tiempo: el tiempo que se pacto que llevaría el desarrollo del producto
  - Costos: los costos (midiéndolos con cualquier tipo de recursos) de desarrollar el producto



- El balance de los tres permite un SW de calidad, pero si uno de los tres factores condiciona a los demás, se sacrificara un poco de uno para obtener un poco del otro, de manera que el SW se vea afectado en algún aspecto (el área resultante representa la calidad el producto)
- Balance de los tres factores: un SW de calidad, que cumple con los alcances determinados y se desarrolla en tiempo y costo establecido.

Equipo de desarrollo

- Esta compuesto por distintas personas que tiene un objetivo en común, y una motivación para alcanzarlo.
- Estas personas tienen un conjunto variado de habilidades, que usan de manera eficiente para juntos desarrollar un SW de calidad
- Todos comparte la responsabilidad del proyecto

Roles en el equipo de desarrollo

- Los roles en un equipo pueden variar (o estar o no definidos, dependiendo del proceso que se escoja), pero pueden incluir: tester, arquitecto, DBA, líder técnico, desarrollador, etc
- El líder NO es un rol del equipo, va a la par de los roles controlando las actividades de cada uno
- En un equipo se busca una cantidad controlable y eficiente de gente (de 4 a 8 personas), ya que en los grupos grandes se complica la comunicación, el control, los costos, los tiempos, la fluidez de trabajo
- Puede que una persona tenga mas de un rol, o que un rol pertenezca a mas de una persona

Plan de proyecto

- Es un documento, una guiá, un artefacto del proyecto
- Es un componente del proyecto
- Da soporte/es útil para el proyecto
- Define que, como, cuando y porque se hacen las actividades
- Tiene un objetivo determinado, que es el mismo que el del proyecto
- Define o estima las cosas/actividades que se van a dar en el desarrollo del proyecto
- Define el camino, el plan a seguir en el proyecto
- Sirve para todo el equipo y para el líder
- Ubica cada componente del proyecto en su lugar

- Estos componentes son:
  - Definición del Alcance del Proyecto
    - Es la definición del alcance que tendra el SW, de la funcionalidad que brindará, del producto final que se espera obtener
    - Se pueden definir dos tipos de alcance para el proyecto:
      - Alcance del producto: las características y funcionalidades que deberá tener el producto final (se compara con la ERS)
      - Alcance del proyecto: Define el trabajo/las tareas (y solo las tareas) que se tienen que realizar para construir ese producto, o sea que define cuando se considerara completo al producto (se compara con el plan de proyecto, sus recursos y fechas asignadas)
  - Definición de Proceso y Ciclo de Vida
    - Define el proceso y ciclo de vida que usará el proyecto
    - Teniendo en cuenta las características del desarrollo, se elegirá entre distintos tipos de procesos y ciclos de vida (algunos serán mas acordes al tipo de proyecto o al equipo de desarrollo)
  - Estimación
    - Realiza estimaciones sobre cosas que no se pueden medir de antemano y presentan incertidumbre, y que necesitan conocerse aunque sea aproximadamente, ayudando a planificar y tomar decisiones a medida que avanza el proyecto (y la incertidumbre va bajando, porque se cuenta con mas información y decisiones tomadas)
    - Se estima (en este orden):
      - Tamaño: a través de CU, lineas de código, cantidad de funciones, etc
      - Esfuerzo: Independiente del tamaño (para un equipo un gran tamaño puede requerir menos esfuerzo que para otro), a través de horas hombre, solapamiento de tareas, etc
      - Tiempo: de desarrollo, el tiempo real que lleva desarrollar el producto
      - Costo: impuestos, insumos, salarios, recursos, licencias, etc
      - Recursos: HW
  - Gestión de Riesgos
    - Se identifican los potenciales riesgos que puedan comprometer el cumplimiento del objetivo del proyecto y se define que se hará en su posible manifestación (se puede prevenir el riesgo, o preparar un plan de contingencia cuando suceda, o mitigarlo una vez que sucedió, o no actuar)
  - Asignación de Recursos
    - Se definirán los roles a usar, las personas que trabajaran en el proyecto, las herramientas que se van a usar, dividiéndolos por etapas, fechas, grupos, etc
  - Programación/Calendarización del proyectos
    - Cuando debe suceder cada cosa, las fechas a cumplir con el desarrollo, etc
  - Definición de Puntos de Control
    - Son los momentos en los que se detendrá a evaluar y controlar el desarrollo/avance actual del proceso, comparando con el desarrollo planeado
  - Definición de Métricas
    - Se definen distintos tipos de métricas para medir distintos aspectos del proyecto (performance, tamaño, esfuerzo, costo, etc)

        - Mayor sea la cantidad de métricas que tengamos (y mayor sea la cantidad de veces que las apliquemos), mas precisamente conoceremos el avance real del proyecto, pero usar métricas supone un costo (de energía, tiempo, esfuerzo, recursos), por lo que se debe definir una frecuencia para hacer la métricas que no comprometa al equipo
        - Por eso se trata de que las métricas sean lo más rápidas posibles de tomar, o que sean automatizadas

- El atraso se da día a día, de manera acumulativa, y compromete el tiempo de entrega y la calidad del producto
- Un proyecto puede atrasarse por distintos motivos:
    - Porque no se hace un monitoreo/control general, de forma que no se sabe exactamente cuanto se hizo y cuanto falta, acumulando el atraso (se pueden realizar controles y monitoreos del avance con métricas que nos definirán que tan desviados estamos del plan de proyecto)
    - Para hacer el monitoreo se pueden usar herramientas o métricas
    - Las métricas se definen en el plan de proyecto, y miden el avance de cierto aspecto del proyecto
    - Tienen un costo asociado (se pierde tiempo, energía, trabajo tomándolas) por lo que no se pueden tomar todo el tiempo, si no que se opta por automatizar la mayor cantidad posible de las mismas

Agile

- Al principio, el desarrollo de SW no tenia un proceso para nada definido, cada desarrollador/equipo lo hacia a su manera, de manera artesanal, sin un plan definido
- Esto trajo resultados muy pobres a la hora de desarrollar y adaptar productos/mantenimiento
- Los militares definieron que el desarrollo debía ser completamente definido, siguen un plan estricto y riguroso de etapas y actividades definidas
- El desarrollo se volvió muy lento, burocrático y poco eficiente (porque se priorizaba mas cumplir a tiempo y documentar el proceso que en construir SW de calidad)
- Los proyectos no cumplían expectativas de funcionalidad, costos tiempos, etc
- Surge un grupo de gente del ámbito informático con una propuesta para solucionarlo
- En 2001 declaran el manifiesto ágil
- **4 valores** (“por sobre” no define una obligación o un intercambio, si no una prioridad general):
    - Individuos e interacciones por sobre procesos y herramientas
        - Los procesos y las herramientas ayudan al trabajo y la eficiencia, son importantes, pero lo que les da su valor real son las personas que los usan, con conocimiento técnico y actitud adecuada. Sin personas, no tendrían valor
        - Una persona muy capaz y talentosa puede prescindir de las mejores herramientas y procesos, pero no viceversa.
        - Se toma a los procesos y las herramientas como un soporte para el desarrollo, y deben adaptarse a los cambios del mismo al proyecto/equipo.
        - Los procesos/herramientas se deben adaptar a las personas, y no al revés.
    - Software funcionando por sobre documentación detallada
        - El software funcional (aplicado sobre prototipos en entregas tempranas) es la mejor fuente de retroalimentación, y es muy útil para desarrollar software de calidad. Agile considera que esto es más importante y prioritario que documentar cada aspecto del desarrollo.
        - Aun así, los documentos son importantes para registrar información histórica, permitir una transparencia del SW y compartir el conocimiento asociados al mismo.
        - Los documentos tienen mucho menos valor para el desarrollo que la interacción y comunicación que se da a través de los prototipos de SW funcional con el cliente.
        - Si la comunicación se realiza a través de la documentación, se perderá la calidad y utilidad de la misma.
    - Colaboración por sobre negociación con el cliente
        - Se suele aplicar Agile en proyectos medianamente difíciles y complejos, en los que la comunicación con el cliente es clave para desarrollar un producto útil y adecuado para el mismo

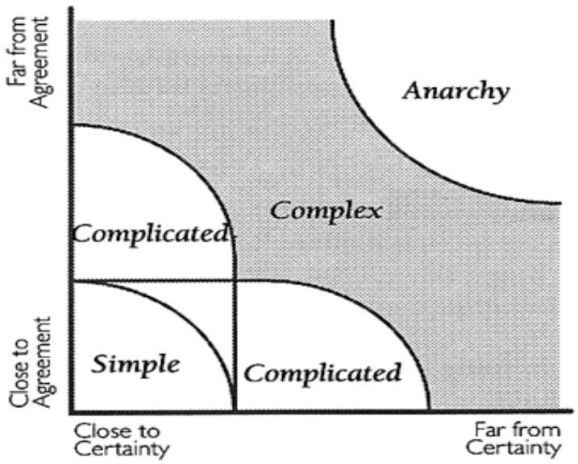
        - El contrato no aporta valor al desarrollo, pero sirve para definir las responsabilidades del producto y las condiciones del producto final
        - El cliente es un miembro más del equipo, y es mucho más importante interaccionar con él que definir un contrato y una forma de trabajar y los procesos asociados al trabajo
    - Responder a cambios por sobre seguir un plan
        - Agile se suele aplicar en entornos inestables, con muchos cambios, en los que el seguimiento estricto y riguroso de un plan no aporta mucho valor al desarrollo, ya que el proyecto esta cambiando constantemente respecto a aspectos que no se podrían haber tenido en cuenta al principio del proyecto, cuando se define el plan del mismo.

- **12 principios**:

1. La prioridad es satisfacer al cliente a través de releases tempranos y frecuentes (2 semanas a un mes) → Mas rápido es la entrega, más rápido es la devolución (rechazo, aceptación, modificación), ahorrando esfuerzo/trabajo/tiempo/recursos malgastados (ademas se trabaja en un ambiente competitivo, entregas frecuentes asegura que el cliente se quede con el equipo)
2. Recibir cambios de requerimientos, aún en etapas finales
3. Release frecuentes (2 semanas a un mes)
4. Técnicos y no técnicos trabajando juntos TODO el proyecto → La parte no técnica (gerencia, administración, etc) se familiariza con el proyecto, pudiendo afectar a las tomas de decisiones. Conociendo el producto tempranamente
5. Hacer proyectos con individuos motivados → Es una prioridad que la persona trabaje cómoda y motivada, da mejores resultados
6. El medio de comunicación por excelencia es cara a cara → Mayor comunicación, entendimiento
7. La mejor métrica de progreso es la cantidad de software funcionando → Al fin y al cabo es lo único que define el progreso real del proyecto, comparado a otras métricas
8. El ritmo de desarrollo es sostenible en el tiempo → Ritmo de avance constante, de manera que no se atrase. No esta mal adelantarse, pero se prefiere un ritmo constante en el tiempo, del que el equipo este acostumbrado
9. Atención continua a la excelencia técnica → Miembros con altos niveles de habilidades técnicas
10. Simplicidad - Maximización del trabajo no hecho → Obteniendo el mejor resultado posible, trabajar lo menos posible. No perder tiempo en cosas que no agregan valor real al producto. Trabajar lo menos posible/lo mas simple posible sin comprometer la calidad/funcionalidad
11. Las mejores arquitecturas, diseños y requerimientos emergen de equipos auto-organizados → De esta forma no se impone un plan ni un control, el avance es mas relajado, se organiza el equipo mismo, repartiendo/delegando en base a la opinión del equipo mismo
12. A intervalos regulares, el equipo evalúa su desempeño y ajusta la manera de trabajar → Restroinspeccion/feedback constante para mejorar

- Agile define un balance entre un proceso completamente definido y uno no definido
- Determina ciertas prioridades para los procesos a la hora de desarrollar un SW
- Principalmente, determina una menor cantidad de documentación, pero también define que el proceso debería ser adaptable en lugar de ser reaccionario, y que el mismo se puede orientar a las personas y no a los procedimientos.
- Agile se basa en un desarrollo incremental, añadiendo mejor/mayor funcionalidad al SW con cada iteración (y no en un desarrollo por partes) → No desarrollo una parte completa, luego otra, y así… Desarrollo el SW completo, de a poco añadiéndole funcionalidades
- Se basa en SW de entregas rápidas en un entorno cambiante
- Se prioriza la velocidad de entrega en un entorno competitivo
- Se involucra al cliente en el desarrollo, para obtener un retroalimentación rápida y constant

- **¿Cuando aplicar Ágil?**



- Si tengo poca certeza y poco entendimiento, no puedo trabajar
- Si tengo mucha certeza y mucho entendimiento, estoy en el ámbito de trabajo más simple
- Ágil suele funcionar mejor en el área de complex, los nuevos desarrollos no suelen estar completamente (ni incompletamente) certeros o en entendimiento.

User Stories

- Es una herramienta o método para descubrir/capturar los requerimientos del sistema, para tener en claro que se debe construir, en metodologías Agile
- Es una descripción corta de una funcionalidad que un rol identifica en el sistema (una US no es un requerimiento funcional, pero nace desde la necesidad de la implementación de un requerimiento funcional, y ayuda a identificarlo y describirlo)
- Es una petición que se hace al sistema en base a una funcionalidad que afecta a un rol.
- Storie = cuentan una historia, algo que se hace de manera narrada
- Representan una necesidad de un rol
- Ayudan a describir el producto (ya que no se trabaja con los requerimientos tradicionales)
- Son un ítem de planificación
- Son un token para la conversación y un mecanismo para diferir en una conversación (fomentan la conversación respecto a esa funcionalidad en el equipo)
- Mas importante que la US misma, es la conversación detrás de la misma entre el equipo de desarrollo y el rol
- Se dice que las US son de naturaleza vertical, porque afectan a todas las áreas del producto y de su desarrollo: su funcionalidad es parte de todo el producto y debe implementarse en todos sus aspectos (técnico, estructura, usuario, despliegue, etc)
- Debido a esta naturaleza vertical, no se dividen en funcionales y no funcionales, cada US tendrá una implicación no funcional que debe resolver la implicación funcional
- No comienzan con una certeza del 100%, se van acomodando en el desarrollo del producto → Se terminan de definir y detallar con el paso del tiempo
- Se pueden escribir con mas o menos detalle, dependiendo de que tan importantes/esenciales son
- Las US no definen como se debe implementar/crear la funcionalidad del sistema, simplemente definen esa funcionalidad (Por ejemplo: el usuario necesita recuperar información perdida → Eso define la US, la recuperación se hará a través de un backup de BD → eso define el equipo que la implementara)
- Tampoco son definiciones muy detalladas de cada requerimiento, si no una necesidad general de un cierto rol que le puede agregar valor al sistema
- Expresan intención (quiero que se haga algo)
- No necesitan mantenimiento
- Una vez que se implementan se pueden descartar

- No se deben pensar teniendo en cuenta el mayor nivel de detalle posible, al contrario, se deben pensar de manera muy general, y solo empezar a pensar en los detalles al final, cuando se este por implementar
- No son ni representan requerimientos, representan una herramienta disparadora para la conversación que detalles e identifique estos requerimientos
- En metodologías Agile, no tenemos una ERS, una descripción de los requerimientos, por lo que el conjunto de las US van a terminar siendo la descripción de mi producto
- Una US puede tener la misma implementación funcional que otra (por ejemplo, que el usuario pueda llamar al taxi mas cercano), pero dependiendo del rol que elijamos (chófer o pasajero) el valor sera distinto (para que el taxi llegue mas rápido / para gastar menos combustible)

Esta compuesta por tres partes: Conversación, Confirmación y Tarjeta

Tarjeta

- Es la representación física de la US
- Es una descripción de la US, sirve para expresarla y como recordatorio

- En el **frente** de la tarjeta se incluye la descripción/narración por parte de un rol de una necesidad del sistema (o un aspecto que puede agregarle valor para el)
- Se expresa de la siguiente forma:
- Como <nombre del rol>, yo puedo <actividad> de forma tal que <valor de negocio que recibo>
- El nombre del rol representa a quien esta haciendo la actividad, y quien recibe el valor de la misma
- La actividad es aquella acción que realizara el rol en el sistema
- El valor de negocio es la justificación de porque es necesaria esa funcionalidad, y nos permite definir cuando es el momento de implementarla
- Después de la narración puede o no ir una nota (que aclara un aspecto de la US, son los criterios de evaluación)

- En el **reverso** se escriben los criterios de aceptación de la US (las condiciones para que se considere completa la US). Le sirve al PO para ver si realmente esa es la implementación que quiere/necesita

Conversación/comunicación

- Es la parte mas importante de la US, es clave para el éxito de la US
- Define de manera dinámica los detalles de la US, su objetivo, como la ve el usuario, etc
- No queda documentada en el proyecto
- Representa las conversaciones cara a cara
- Son todas las discusiones/conversaciones acerca de la US (entre miembros del equipo y el Product Owner) que sirven para desarrollarla/implementarla.

Confirmación

- Son las pruebas de usuario que permiten determinar cuando la US esta completamente resuelta (determinan si se cumplen los criterios de aceptación)
- Son las cosas que hará el PO sentado al frente del sistema para ver que cumpla los criterios de aceptación.

Product Backlog

- Es el conjunto de cosas por hacer/implementar en el proyecto.
- Tiene una naturaleza de pila, teniendo arriba los ítems de mayor prioridad/mas próximos a realizar y mas detallados, y abajo los ítems con menor prioridad/los que se resolverán mas tarde y menos detallados (porque para implementar una US, esta debe estar completamente detallada, para saber exactamente que se quiere hacer y como hacerlo)
- En cada iteración, se realizan las US mas prioritarias del Backlog
- En él, se les asigna una prioridad a cada US (el rol que lo asigna es el Product Owner, que es el cliente, que conoce el negocio y define la prioridad de las US, siendo considerado parte del equipo)

Modelado de roles

- Rol ≠ persona
- Un rol puede atribuírsele a una persona, o a varias, o no tener una persona definida que lo cumpla
- En desarrollos ágiles, las personas no tienen asignado un rol, si no que el rol define las tareas que alguien debe hacer en una actividad
- Cuando se describe un rol, se basa en que espera ese rol del SW (tiempo de respuesta, performance, claridad, intuitividad, etc)
- Al describir un rol, tenemos que definir: que tan familiarizado esta con el uso del SW (o SW parecido), que prioridades tiene, que tan frecuentemente va a usar el SW, cuanto le importa el tiempo de respuesta, etc
- Se les tiene que asignar un nombre especifico
- Para definir un rol se pueden usar distintas técnicas:
  - Tarjeta de rol de usuario: es una tarjeta que tiene una descripción del usuario y define que características son importantes/relevantes para este rol, así como sus actividades habituales
  - Personas: se describe a la persona en ese rol (quien es, como es, que suele hacer, que sabe y no sabe hacer, etc)

Criterios de aceptación de la US

- Es la descripción de la US que se le presenta al PO para verificar que este completa
- El PO realizara pruebas de usuario (pruebas de ciertas acciones con ciertos valores) para validar el cumplimiento de la US

Definición de HECHO/DONE

- Representa cuando esta lista la US para mostrársela al PO (no para implementarla)
- Representa cuando la US esta completamente definida/detallada, antes de implementarla
- En cada equipo/proyecto puede variar este criterio
- Para un equipo, puede considerarse DONE cuando haya pasado una revisión de pares, o que cumpla los criterios de aceptación, o que cumpla con las pruebas, o que ya sea SW funcionando

Definición de LISTO/READY

- Representa cuando la US esta lista para incluirse en un sprint/iteración (es decir que esta lista para que el próximo sprint se desarrolle, se implemente en el sistema)
- Para que se pueda implementar debe tener un alto grado de detalle, el suficiente para saber exactamente como implementarla
- El equipo decidirá cuando la US esta lista
- El equipo puede tener en cuenta distintos criterios, uno de los mas famosos es el modelo INVEST (aunque no significa que si cumple el modelo INVEST ya esta lista)
  - [I]ndependiente: puede implementarse en cualquier orden o lugar, no depende de otras US, otros componentes, etc
  - [N]egociable: no debe detallar cómo implementarla, sino qué se debe implementar (el que, no el como). Se puede negociar el como, pero no el que
  - [V]aluable: Debe presentar un valor para el cliente, debe agregarle valor al sistema
  - [E]stimable: Debe poder medirse/estimarse (para ayudar al PO a armar un ranking basado en costos)
  - [S]mall/chica: Debe ser de tamaño pequeño, poder llevarse a cabo en una iteración (15 días)
  - [T]esteable: Se debe poder probar que se implemento correctamente
- También se puede considerar para que una US este lista que cumpla con el criterio de DONE

Las US se pueden clasificar según el nivel de abstracción

- User Storie
  - Es una descripción de funcionalidad deseada, contada desde la perspectiva del cliente
  - Cumple con el modelo INVEST
- Epics
  - Son una US de gran tamaño, de mucha funcionalidad, que tiene mucha incertidumbre

    - No cumple modelo INVEST (sobre todo Small)
    - No pueden implementarse en una sola iteración
    - Para implementarla en una iteración, debe dividirse en US mas pequeñas (cuando ya se fue disminuyendo la incertidumbre inicial y se tiene en claro los detalles de su implementación)
    - PE: implementar el modulo de compras
- Theme/tema
    - Son una colección de US relacionadas, por un criterio de negocio (o sea de manera horizontal), como por ejemplo de cierta área del negocio o cierta funcionalidad
    - PE: todas las US que administren el usuario (crea, modificar, clonar, dar de baja, etc)

Spikes

- Son un tipo de US
- Se usan en US en las que se tiene incertidumbre, en las que no se tiene total seguridad de algún aspecto de la misma, para reducir dicho riesgo/incertidumbre
- Definen como se debe trabajar, teniendo en cuenta un aspecto relacionado a la US (explica o aporta información)
- Puede ser de dos tipos:
    - Funcional
        - Algo que no quedo claro que quiere el cliente o como se relaciona la US con el negocio
        - Esta relacionado a la funcionalidad de la US
        - No se tiene en claro como interaccionara el US con el sistema
    - Técnicas
        - Algo que no estoy seguro como implementar a nivel técnico
        - Puede estar relacionado a la performance de la US
        - Ayuda a decidir si hacer algo o comprarlo hecho
        - Evalúa que tecnología utilizar, o como utilizarla si ya se la tiene definida
- Se debe poder estimar (al igual que una US) y se debe poder cumplir en una iteración (al igual que una US)
- Puede haber varias spikes para la US, tanto funcionales como técnicas
- Normalmente se aplican en iteraciones distintas a la de la US, a menos que sea un spike muy pequeño que se pueda resolver antes de la US en la misma iteración
- Pueden usarse para:
    - conocer mas sobre una US y así poder estimarla, o dividirla
    - adquirir conocimiento respecto a tecnologías, técnicas

Estimaciones Ágiles

- Los ítems que se encuentran en el Backlog (US, spikes, Epics, etc) se estiman a través de Storie Points (sp)
- Esta estimación no es absoluta, siguiendo una escala, si no que es relativa a otros elementos (porque comparar es mas rápido y mas acertado que dar una medida absoluta)
- Su asignación depende del equipo de desarrollo, ya que el equipo estima para si mismo, basándose en si mismo, en su nivel de habilidad, de entendimiento, etc
- Al principio, lo primero que se estima es el tamaño de la US (no el esfuerzo que lleva completarla, si no el tamaño de la misma), que representa la cantidad de trabajo, la complejidad de una US
- Se usara una escala (definida por el usuario), como puede ser una escala lineal, sucesión de Fibonacci, tamaño de prendas de ropa, etc (una vez que se elige la escala, no se cambia)
- Los sp van a dar una medida relativa del peso o complejidad de la US
- Los sp tienen tres componentes, que podrán tener distinto peso en cada US:
    - Complejidad
        - relacionado a cantidad de partes, o cuanto se relacionan sus partes, o que tan complejas son cada parte
        - aumenta exponencialmente al avanzar la US
    - Esfuerzo
        - Cuanto cuesta desarrollarla
        - No es lo mismo que tamaño (un equipo puede resolver una US con un gran tamaño con poco esfuerzo)

    - Duda
        - Representa la incertidumbre, los aspectos que todavía no se conocen para desarrollar la US
        - Puede ser incertidumbre de negocio (funcional) o técnica
        - Puede ser grande o nula

Velocity/Velocidad

- Es una métrica, una medición que se hace del progreso del equipo en el desarrollo de un proyecto
- Representa el avance del equipo en lo que va del proyecto
- Para calcularla, se suman los sp de todas las US que se implementaron (solo aquellas que se implementaron de manera completa) en una iteración
- Sirve para corregir errores de estimación
- Permite visualizar con que ritmo trabaje el equipo
- Tiene a estabilizarse a medida que avanza el proyecto
- Si sumo todos los sp de TODAS las US del proyecto,, y los divido por la velocidad, obtendré una estimación de tiempo para la planificación

Existen distintos métodos para estimar US

Poker planning

- Primero se visualizan todos los elementos a estimar
- Se elige una US canónica, que sera la base a la que se compararan todas las demás US
- Esta US canónica debe ser simple, deben entenderla todos los integrantes del equipo y debe tenerse total certeza sobre ella, nada de incertidumbre
- Representara la unidad (1) de estimación
- Una vez elegida no se cambia
- Voy pasando las demás US, una por una, y comparándola a la US canónica, y asignándole un puntaje según mi escala
- Cada integrante del equipo puntuara la US a través de una carta
- Los sp resultantes asignados a una US surgirán del promedio de sp de los integrantes

Gestión de configuración (SCM)

- El SW puede evolucionar con el paso del tiempo, paliarse, reducirse, volverse más complejas, etc
- Estos cambios en el SW pueden traer problemas si no se gestionan con cuidado.
- Los cambios se dan principalmente por cambios en el negocio (nuevos requerimientos), porque los productos asociados cambiaron y ya no puedo darles soporte, porque la empresa creció, porque cambio el presupuesto inicial, porque se encontraron defectos de funcionamiento, porque existen oportunidades de mejora, porque cambiaron los HW que lo soportan, etc
- El cambio es inevitable y beneficioso para el SW, porque significa que alguien lo esta usando, que esta siendo útil y se encuentran nuevas funcionalidades posibles
- Cuando se dan estos cambios, se debe asegurar la integridad del SW
- Esto significa que los cambios no deben afectar a la consistencia y las relaciones de los elementos que conforman el SW
- Para mantener la integridad del producto durante los (inevitables) cambios que sufrirá, se crea la SCM
- SCM es una disciplinas transversal de soporte al sistema, lo que quiere decir que da soporte a todo el proceso de desarrollo, desde las etapas mas superiores hasta las mas profundas, asegurándose de mantener en todas sus etapas la integridad de los componentes del producto
- Es una disciplina que aplica dirección y monitoreo administrativo y técnico a: identificar y definir las características de los ítems de configuración, controlar sus cambios, registrar y reportar los cambios y su estado de implementación y verificar que correspondan con los requerimientos
- Se aplica a las diferentes disciplinas dentro del desarrollo del producto, durante todo el ciclo de vida

- Involucra identificar la configuración en un momento dado, controlar los subsiguientes cambios y mantener la integridad durante estos cambios
- La integridad asegura que los términos o principios con los que fue creado el SW se mantenga inalterable a través de los cambios
- Un producto se considera integro cuando:
  - Satisface las expectativas del usuario (cumple los requerimientos funcionales)
  - Puede ser fácilmente rastreado durante su ciclo de vida, tiene trazabilidad (se puede saber, para cualquier componente del sistema, cuando se creo y que cambios sufrió, en que etapa del ciclo de vida del desarrollo)
  - Satisface criterios de performance (satisface requerimientos no funcionales)
  - Cumple las expectativas de costo
- Al manejar muchos componentes, se pueden encontrar problemas como: perder un componente en las distintas versiones, perdida de sincronía, registrar fallas, doble mantenimiento, cambios no validados, etc

Existen algunos **conceptos claves** relacionados a la SCM

Ítem de configuración

- Es cada artefacto que se ve afectado por la SCM (cualquier artefacto que pueda sufrir cambios y sea útil registrar sus cambios y conocer su estado actual y en el tiempo)
- Pueden ser sometidos al control de versión
- Puede que se requiera que se compartan entre los integrantes del equipo
- Se presentan mas que nada en procesos definidos, en Agile hay muchos menos (porque se documenta menos)
- Es todo aquello para lo que se registren los cambios en el tiempo
- Incluyen:
  - ❖ Plan de CM
  - ❖ Propuestas de Cambio
  - ❖ Visión
  - ❖ Riesgos
  - ❖ Plan de desarrollo
  - ❖ Prototipo de Interfaz
  - ❖ Guía de Estilo de IHM
  - ❖ Manual de Usuario
  - ❖ Requerimientos
  - ❖ Plan de Calidad
  - ❖ Arquitectura del Software
  - ❖ Plan de Integración
  - ❖ Planes de Iteración
  - ❖ Estándares de codificación
  - ❖ Casos de prueba
  - ❖ Código fuente
  - ❖ Gráficos, iconos
  - ❖ Instructivo de ensamble
  - ❖ Programa de instalación
  - ❖ Documento de despliegue
  - ❖ Lista de Control de entrega
  - ❖ Formulario de aceptación
  - ❖ Registro del proyecto
- Estos ítems pueden tener distintas **versiones**, que registran su estado y configuración actual en un momento de tiempo y contexto determinado
- También pueden tener **variantes**, que son versiones que evolucionan por separado (las versiones evolucionan de manera lineal, las variantes toman otra ruta de evolución aparte).
  - Por ejemplo, se tendrá una variante de un componente para cada SO que se aplique

Linea Base

- Son etiquetas, que sirven para marcar un conjunto de componentes que son/fueron consistentes entre si en un momento dado
- Relaciona componentes
- Son una configuración del producto que se reviso formalmente y sobre la que se llego a un acuerdo (funciona, es estable)
- Sirve como base para desarrollos posteriores
- Solo puede cambiarse a través de procesos formales
- Permiten volver en el desarrollo y reproducir un entorno anterior
- Son algo parecido a una versión, pero de todo el proyecto
- Las lineas bases se definen en un momento en el que los componentes crean una versión integra del producto, estable, que anda
- No se debe desarrollar/alterar la configuración de una linea base, a menos que se quiera mejorar o seguir desarrollando
- Puede relacionar cualquier tipo de componentes

Configuración del Software

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

Ramas/Branch

- Son como variantes, pero de todo el proyecto
- Se divide el desarrollo en base de un criterio, por ejemplo de una funcionalidad, sin afectar a las otras ramas
- Existe una rama principal (tronco/trunk/master)
- La razón de la creación de una rama puede variar
- Permiten la experimentación
- Una rama puede unirse nuevamente a la rama principal o descartarse (por ejemplo si se estaba haciendo una prueba)
- Tarde o temprano deja de existir (porque se integro a la rama principal o porque se descarto)
- Las ramas solo dividen la parte del código que se va a modificar
- La operación de integración de una rama a la principal se llama merge, y puede generar conflictos entre ambas

Repositorio

- Un repositorio es un contenedor de ítems e configuración
- Contiene ítems de configuración y herramientas que nos ayudan a documentar sus cambios y poder administrarlos
- Permiten el acceso organizado y controlado a los integrantes, y maneja las versiones de los ítems de configuración de manera que se lleva un registro de los mismos
- Almacena aquellos ítems de los que quiero llevar registro de sus cambios
- Cada usuario posee una copia del repositorio (o tiene acceso al mismo) y puede (o no) editarlo
- De cada ítem mantiene su relación con otros componentes y su historial de cambios/versiones
- Puede ser de dos tipos: centralizados (todo el SW se guarda en el mismo lugar en un servidor, no se trabaja con copias locales particulares, los administradores tienen mayor control sobre el repositorio, ante una eventual falla del servidor traerá muchos problemas) o descentralizado (el SW se reparte, se guarda de forma local, cada integrante tiene una copia idéntica del producto, posibilita el trabajo divergente)

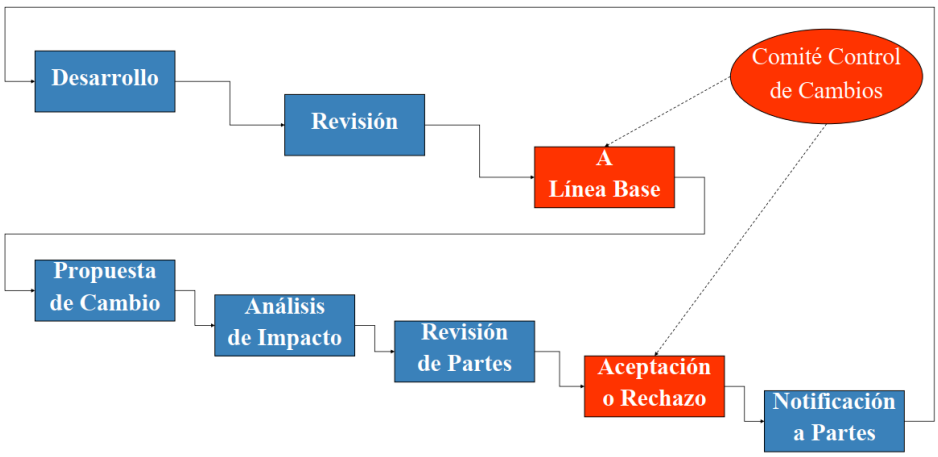
En SCM, existen **cuatro actividades fundamentales**

Identificación de ítems de configuración

- Se basa en identificar a los ítems de configuración, y una vez identificados darles una identificación unívoca (única, clara y que represente su naturaleza), de manera que no se repitan ni se confundan la identificación, a través del versionado y las distintas reglas de nombrado
- Luego se definirá la estructura que tendrá el repositorio, y se ubicara a cada ítem dentro de su ubicación correspondiente
- Para poder identificar los ítems de configuración, debo conocer la naturaleza del proyecto, ya que dependiendo de ella, algunos ítems pueden o no estar presentes
- Los ítems de configuración pueden ser de tres tipos:
  - De producto: aquellos que se usan o tienen utilidad aun cuando termino el proyecto y ya se esta usando el producto → ERS, código, manual de usuario
  - De proyecto: aquellos que no tienen mas utilidad una vez terminado el proyecto (aunque se documenten) → Plan de proyecto, cronograma
  - De iteración: son un caso particular de los de proyecto, que solo viven en una iteración del mismo → plan de iteración, reporte de defectos

Proceso de control de cambios

- La linea base define una marca para relacionar a todos los componentes que, juntos, forman una versión estable y consistente del producto
- Para cambiar la linea base (cambio que solo puede deberse a seguir con el desarrollo o mejorarla), se debe pasar por un proceso formal.
- Se comenzara un proceso de cambio cuando haya un requerimiento de cambio en uno o mas ítems de configuración que conforman la linea base
- Es un procedimiento formal y definido
- Partiendo de la linea base, se realiza una propuesta de cambio, se analiza su impacto, se revisan sus partes, se presenta al comité de control de cambios y si se acepta, se notifica a las partes y se desarrolla, para luego revisarla y definirla como una nueva linea base

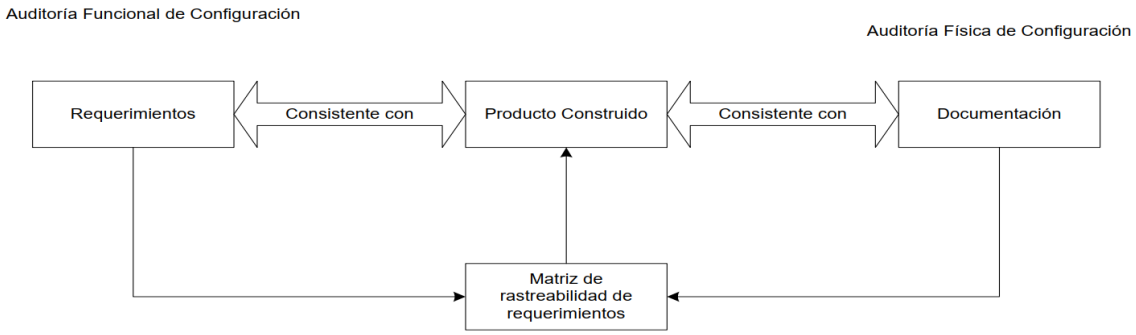


- Esa es una base o modelo del que parten todos los proceso de cambio, pero en cada organización sera distinto (esta definido en el plan de proyecto)
- El comité de control de cambios es un conjunto de personas que acepta o rechaza los cambios, basándose en su análisis y revisión.
- El comité debe representar a todo el equipo de desarrollo, de manera que se tengan en cuenta los puntos de vistas de todas las áreas, ya que el cambio las afectara a todas.
- En proyectos tradicionales, hay un rol que es el de gestor de cambios
- En principio, este proceso de control de cambios se usa principalmente en procesos tradicionales, no así en Agile

- Este proceso esta definido en el plan de proyecto (que define como sera el proceso de control de cambios)

Auditorías de Configuración de Software

- La auditoria suele llevarse a cabo en procesos definidos, en Agile no se hace auditoria (o al menos no por alguien externo)
- Existen dos tipos de auditorias:
- Auditoria de configuración física
  - Compara con la realidad
  - Asegura que se cumple lo que dice el plan de gestión de cambios de manera física (nombres de archivos, ubicaciones, etc)
  - Asegura que lo que indica en la linea base para cada ítem de configuración este realmente configurado de esa manera
- Auditoria de configuración funcional
  - Se fija si cada ítem de configuración se corresponde con los requerimientos funcionales del producto
  - Evalúa cada componente del producto, y se asegura de que su funcionalidad y performance estén en sintonía con la especificación de requerimientos
  - Compara con la ERS
  - Usa la matriz de trazabilidad
  - Establece si hay una trazabilidad en cada componente desde que se definió hasta que termino el proyecto
  - Asegura la consistencia de la funcionalidad
- La matriz se fija en un requerimiento funcional y viendo algún ítem de configuración define si cumple esa funcionalidad, si sigue siendo consistente desde cuando se definió, etc
- Sirven para dos procesos básicos:
- Validación:
  - Si el resultado del proyecto (el producto que se le entrega al cliente) es valido, es el producto correcto que necesita
  - Si se esta construyendo el producto que el usuario requiere
  - Relacionado con la auditoria funcional
- Verificación
  - Controla que el producto este siendo construido correctamente
  - Controla que el producto cumpla con los objetivos definidos, que todas sus funciones sean llevadas a cabo
  - Relacionado con la auditoria física



- Esta matriz de rastreablidad se usa en las dos auditorias, pero mucho mas en la funcional

Informe de estados

- Son informes y registros que se ocupan de mantener registrada la evolución del sistema
- Manejan información y salidas que suelen procesarse de manera automática
- Incluye reportes de rastreablidad de todos los cambios hechos a la linea base en el ciclo de vida
- Debe poder definir que estado tiene/tenia un ítem en un momento dado
- Debe poder definir si se acepto/rechazo tal propuesta de cambio
- Debe poder definir las diferencias entre una versión y otra de un ítem de configuración

- Permiten saber en que lugar estoy parado respecto de mi SCM

Plan de gestión de configuración

- Es un plan relacionado a la gestión de configuración
- Debe incluir:
  - ❖ Reglas de nombrado de los CI
  - ❖ Herramientas a utilizar para SCM
  - ❖ Roles e integrantes del Comité
  - ❖ Procedimiento formal de cambios
  - ❖ Plantillas de formularios
  - ❖ Procesos de Auditoría

SCM en Agile

- Se basa en aplicar SCM respetando los 4 valores y sus 12 principios
- Se automatiza la mayor cantidad posible de SCM (respetando el poco control que tiene Agile, y evitando que las personas pierdan tiempo en ello en vez de desarrollar el SW)
- No se basa en el control, ya que lo equipos se auto-organizan
- No hacen auditorias
- Aunque no se controle, se aplica la rastreablidad en los componentes, porque sirve para mantener la consistencia del producto
- SCM se aplica en todos los proyectos, pero en los Agile se hace de manera mas leve
- No aplica SCM en su totalidad, pone el foco en algunos aspectos, como la automatización, la trazabilidad, y no basarse en el control
- Trato de que las actividades de SCM sean las menos posibles, y las incluyo en las actividades que se hacen dentro de cada sprint
- No existe un comité de control de cambios, si no que el mismo equipo delibera que cambios hacer o no
- Solo se trabaja con los reportes necesarios