(1)

RESUMEN ISW - PDFs

SW: SET DE PROGRAMAS Y LA DOCUMENTACIÓN QUE LA ACOMPAÑA

TIPOS
- SYSTEM SW
- UTILITARIOS
- SW DE APLICACIÓN

SOFTWARE ≠ MANUFACTURA
- SW ES MENOS PREDECIBLE
- NO HAY PRODUCCIÓN EN MASA, NINGÚN SW ES IGUAL A OTRO
- NO TODAS LAS FALLAS SON ERRORES
- SW NO SE GASTA
- SW NO ESTÁ GOBERNADO POR LAS LEYES DE LA FÍSICA

PROBLEMAS
- VERSIÓN FINAL NO SATISFACE AL CLIENTE
- DIFÍCIL DE EXTENDER O ADAPTAR
- MALA DOCUMENTACIÓN
- MALA CALIDAD
- MÁS TIEMPO Y COSTO QUE LOS PRESUPUESTOS.

PARA QUE NOS VAYA BIEN
- INVOLUCRAR AL CLIENTE
- APOYO DE GERENCIA
- ENUNCIADO CLARO DE REQ.
- PLANEAMIENTO ADECUADO
- EXPECTATIVAS REALISTAS
- PERSONAS INVOLUCRADAS COMPE[TENTES]

"SABER PROGRAMAR NO ES SUFICIENTE"

ING DE SW - PARNAS: "MULTI-PERSON CONSTRUCTION OF MULTI-VERSION SW"

→ CUERPO DE CONOCIMIENTO DE ING DE SW (SWEBOK) [IEEE V3.0 2014]
15 ÁREAS (SW = REQ, DESIGN, CONSTRUCTION, TESTING, MAINTENANCE, CONFIGURATION MANAGEMENT, ENGINEERING (MANAGEMENT, PROCESS, MODELS AND METODS, PROFESSIONAL PRACTICE, ECONOMICS), COMPUTING FOUNDATIONS, QUALI[TY], MATEMATICAL FOUNDATIONS, ENGINEERING FOUNDATION)

→ DISCIPLINAS

TÉCNICAS
- REQ
- ANÁLISIS
- DISEÑO
- CONSTRUCCIÓN
- PRUEBA
- DESPLIEGUE

GESTIÓN
- PLANIFICACIÓN DE PROYECTO
- MONITOREO Y CONTROL DE PROCESOS

SOPORTE
- GESTIÓN DE CONFIG. DE SW
- ASEGURAMIENTO DE CALIDAD
- MÉTRICAS

|| COMO DISCIPLINA DE GESTIÓN ES UNA ACTIVIDAD TRANSVERSAL A TODO EL PROYECTO CON APLICACIÓN EN LAS DIF DISCIPLINAS ||

SOFTWARE CONFIGURATION MANAGEMENT (SCM) = UNA DISCIPLINA QUE APLICA DIRECCIÓN Y MONITOREO ADMINISTRATIVO Y TÉCNICO A: IDENTIFICAR Y DOCUMENTAR LOS 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 REQ. (ANSI/IEEE 828, 1990) PROPÓSITO = ESTABLECER Y MANTENER LA INTEGRIDAD DE LOS PRODUCTOS DEL PROYECTO DEL SW A LO LARGO DE SU CICLO DE VIDA.

CAMBIOS EN EL SW
- CAMBIO DE NEGOCIO Y NUEVOS REQ
- SOPORTE DE CAMBIO DE PRODUCTOS ASOCIADOS
- REORGANIZACIÓN DE PRIORIDADES DE LA EMPRESA POR CRECIMIENTO
- CAMBIO EN EL PRESUPUESTO

* DEL MOMENTO CERO
* CONTROLAR SET. LOS CAMBIOS

Problemas en el Manejo de un Componente: Pérdida de un Comp, Pérdida de Cambios, Sincronía Fuente – Objeto – Ejecutable, Regresión de Fallas, Doble Mantenimiento, Superposición de Cambios, Cambios No Validados

Integridad del Producto → Satisface Necesidades de Usuario, Puede Ser Fácil y Completamente Rastreable en cualquier momento del ciclo de vida, Satisface Criterios de Performance, Cumple Expectativas de Costo

SCM (Elementos)

- Identificación de Items de Configuración = Establecer estándares de documentación y un esquema de identificación de documentos.
- Control de Cambios = Evaluación y registro de todos los cambios que se hagan de la SCM
- Auditorías de Config = Junto con las revisiones técnicas formales para garantizar que el cambio se ha implementado
- Generación de Informes: Registro de la Evolución del Sistema, se suelen implementar dentro de procesos automáticos, rastreabilidad de todos los cambios realizados en la Línea Base.

Items de Config = Artefactos que forman parte del Producto o participan de la gestión de Proyecto que pueden sufrir cambios o necesitan ser compartidos entre los miembros del equipo y necesitamos conocer su estado y evolución. Plan CM, Propuesta de Cambio, Manual de Usuario; Casos de Prueba, Código Fuente, Registro de Proyecto, Arq de SW, Plan de Calidad, Plan de Integración.

Control de Cambios – Origina en 1 o Varios Cambios Items que se encuentran en Línea Base. Procedimiento formal que involucra diferentes actores y una evaluación de impacto del cambio

- Comité = Forma de Representantes en Áreas involucradas = Análisis, Diseño, Impl, Test, Integración

Repositorio = Depósito de info conteniendo los Items de Config. Mantiene historia de los CI con sus atributos y relaciones (Metadatos). Usado para hacer evaluaciones de impacto de los cambios propuestos. 1 o Varias BD. Poseer Herra de Automatización, Integración y generación de informes.

- Centralizados = Un servidor tiene todos los archivos con sus versiones, los admin tienen mayor control sobre el repo.
- Descentralizados = Cada Cliente tiene una copia igual del Repo Completo, si un servidor muere es copiar y pegar, posibilitan otros WF no disponibles en Modelo Centralizado

Auditorías de Config (Tiempo y Dinero, Suministra Visibilidad y Rastreabilidad de Ciclo por Varios)

- Auditoría Física de Config (PCA) = Asegurar lo que está indicado para cada SCI en la Línea Base o en la actualización se ha alcanzado realmente
- Auditoría Funcional de Config (FCA) = Evaluación independiente de los productos de SW verificando que la funcionalidad y performance reales de cada item de Config

Configuración = Conjunto de Ítems de Configuración con su correspondiente versión en un momento dado.

Línea Base = 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 A TRAVÉS DE PROCEDIMIENTOS FORMALES DE CONTROL DE CAMBIOS. PERMITE IR ATRÁS EN EL TIEMPO Y REPRODUCIR EL ENTORNO DE DESARROLLO EN UN MOMENTO DADO DEL PROYECTO. SE UTILIZAN ETIQUETAS PARA MARCAR LA BASELINE. LAS LÍNEAS BASE PUEDEN SER DE ESPECIFICACIÓN O DE PRODUCTOS QUE HAN PASADO POR UN CONTROL DE CALIDAD DEFINIDO

Ramas = RAMA PRINCIPAL – MASTER, SIRVEN PARA BIFURCAR EL DESARROLLO, PERMITE EXPERIMENTACIÓN, PUEDEN SER DESCARTADAS O INTEGRADAS

└→ Integración = "Merge", LLEVA LOS CAMBIOS A LA RAMA PRINCIPAL. SI SURGEN CONFLICTOS SE RESUELVE CON DIFF

Plan de Gestión de Configuración = INCLUYE REGLAS DE NOMBRADO DE CI. HERRAMIENTA A UTILIZAR PARA SCM, ROLES E INTEGRANTES DE COMITÉ, PROCEDIMIENTO FORMAL DE CAMBIOS, PLANTILLA DE FORMATOS, PROCESOS DE AUDITORÍA

SCM EN ENTORNOS ÁGILES
- INDIVIDUOS E INTERACCIÓN SOBRE PROCESOS Y HERRAMIENTAS
- SW FUNCIONANDO SOBRE DOCUMENTACIÓN EXTENSIVA
- COLABORACIÓN CON EL CLIENTE SOBRE NEGOCIACIÓN CONTRACTUAL
- RESPUESTA ANTE EL CAMBIO SOBRE SEGUIR UN PLAN

- COMITÉ: EN DUDA?
- CI = FOTOS DE BURNDOWN CHART, CORREOS DE APROBACIÓN DE SPRINTS, ROLES, USER STORIES, BACKLOGS, CONTENIDO DE RETROSPECTIVA

ANTIPATRONES = Merge-Paranoia, Merge-Mania, Big Bang Merge, Developer Freeze, Never-Ending Merge, Wrong-Way Merge, Branch Mania, Cascading Branches, Berlin Wall.

Testing de Software

SCM { CONTROL DE CALIDAD DE PROCESO, CONTROL DE CALIDAD DE PRODUCTO } → **ASEGURAMIENTO DE CALIDAD DE SW**; PRUEBA DE SW

ASEGURAR LA CALIDAD VS CONTROLAR LA CALIDAD

UNA VEZ DEFINIDOS LOS CRITERIOS DE CALIDAD HAY QUE TENER EN CUENTA ES = LA CALIDAD NO PUEDE INYECTARSE AL FINAL // LA CALIDAD DEL PRODUCTO DEPENDE DE TAREAS REALIZADAS DURANTE TODO EL PROCESO // DETECTAR ERRORES EN FORMA TEMPRANA AHORRA ESFUERZOS, TIEMPO, RECURSOS // LA CALIDAD NO SOLAMENTE ABARCA ASPECTO DEL PRODUCTO SINO TAMBIÉN DEL PROCESO Y COMO ESTOS SE PUEDEN MEJORAR Y QUE SE EVITEN DEFECTOS RECURRENTES // EL TESTING NO PUEDE ASEGURAR NI CALIDAD EN EL SW NI SW DE CALIDAD

DEFINICIÓN DE PRUEBA DE SW = PROCESO DESTRUCTIVO DE TRATAR DE ENCONTRAR DEFECTOS EN EL CÓDIGO. SE DEBE IR CON UNA ACTITUD NEGATIVA PARA DEMOSTRAR QUE ALGO ES INCORRECTO. TESTING EXITOSO → SE ENCUENTRAN DEFECTOS 30% A 50% DEL COSTO DE UN SW CONFIABLE

PRINCIPIOS DE TESTING

★ EL TESTING MUESTRA PRESENCIA DE DEFECTOS ★ EL TESTING EXHAUSTIVO ES IMPOSIBLE
★ TESTING TEMPRANO ★ AGRUPAMIENTO DE DEFECTOS ★ PARADOJA PESTICIDA
★ EL TESTING ES DEPENDIENTE DEL CONTEXTO ★ FALACIA DE LA AUSENCIA DE ERRORES
★ UN PROGRAMADOR DEBERÍA EVITAR PROBAR SU PROPIO CÓDIGO ★ EXAMINAR EL SW PARA PROBAR QUE NO HACE LO QUE SE SUPONE QUE NO DEBERÍA HACER. TAMBIÉN PROBAR LO QUE NO SE SUPONE QUE DEBERÍA HACER ★ UNA UNIDAD DE PROGRAMACIÓN NO DEBERÍA PROBAR SUS PROPIOS DEFECTOS ★ NO PLANIFICAR EL ESFUERZO DE TESTING SOBRE LA SUPOSICIÓN DE QUE NO SE VAN A ENCONTRAR DEFECTOS

¿Cuánto Testing es suficiente? TESTING EXHAUSTIVO ES IMPOSIBLE!!

- → EVALUACIÓN DE NIVEL DE RIESGO → PARA DETECTAR QUE TESTEAR PRIMERO, A QUE DEDICAR MÁS ESFUERZO, QUE NO TESTEAR
- → COSTOS ASOCIADOS AL PROYECTO

El criterio de Aceptación es lo que se usa para resolver el problema de determinar cuando una determinada fase de Testing ha sido completada. Puede ser definido en términos de = Costos, % de Test corridos sin fallas, no hay defectos que detemen la seguridad del software.

ERROR = SE ENCONTRÓ DENTRO DE LA ETAPA EN LA QUE SE ESTÁ TRABAJANDO

DEFECTO = SE ENCONTRÓ EN UNA FASE CONTRARIA (PRODUCCIÓN)

Cuando hago un Release? → DEPENDE DE LOS CRITERIOS QUE ESTÁN DEFINIDOS DE SEVERIDAD Y PRIORIDAD EJ: 0004S DE SEV

1 Bloqueante – Urgente
2 Crítico – Alta
3 Mayor – Media
4 Menor – Baja
5 Cosmético

Niveles de Pruebas:

★ **Testing Unitario** = SE PRUEBA TRAS SU CONSTRUCCIÓN, CADA COMPONENTE (INDIVIDUAL) E INDEPENDIENTE. SE PRODUCE CON ACCESO AL CÓDIGO BAJO PRUEBAS Y CON EL APOYO DE DESARROLLO, TALES COMO UN FRAMEWORK O HERRAMIENTAS DE DEPURACIÓN. LOS ERRORES SE REPARAN AL MOMENTO QUE SE ENCUENTRAN SIN DOCUMENTAR

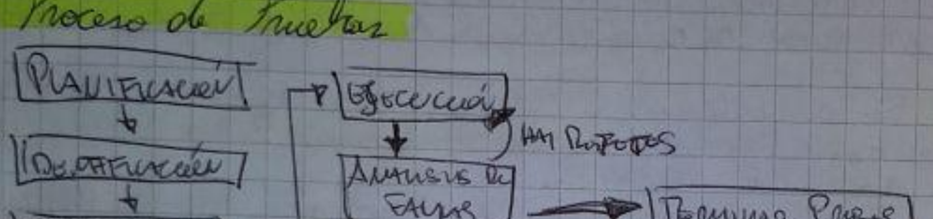
★ **Testing de Integración** = Test ORIENTADO A VERIFICAR QUE LAS PARTES DE UN SISTEMA QUE FUNCIONAN BIEN AISLADAMENTE, TAMBIÉN LO HACEN EN CONJUNTO. Cualquier ESTRATEGIA DE PRUEBA DE VERSIÓN O INTEGRACIÓN DEBE SER INCREMENTAL (TOP-DOWN // BOTTOM-UP) → IDEAL CONTINUACIÓN. HAY QUE TENER EN CUENTA MÓDULOS CRÍTICOS A PROBAR A TIEMPO

PUNTOS CLAVES =
- ↗ CONECTAR PARTES MÁS COMPLEJAS
- ↘ MINIMIZAR LA NECESIDAD DE PROGRAMAS AUX

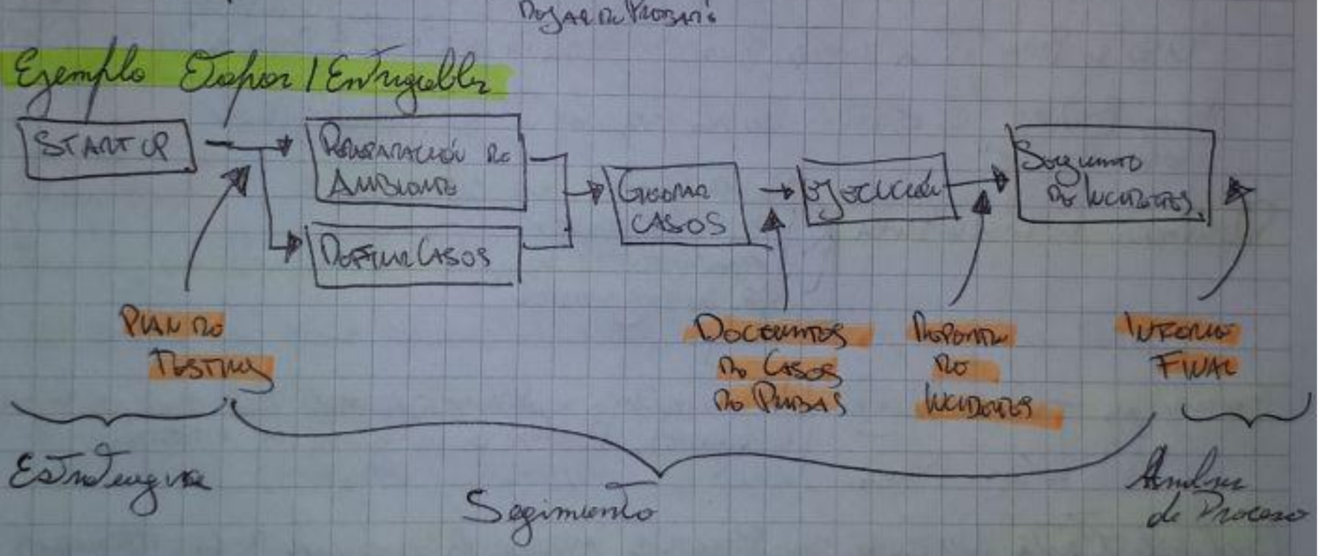
★ Testing de Sistemas = Prueba cuando la aplicación funciona como un todo (Prueba de construcción final). Determina si el sistema en su totalidad opera satisfactoriamente (recuperación de fallas, seguridad y protección, stress, performance etc.)
En el entorno de pruebas debe corresponder al entorno de producción. Se deben probar req. funcionales y no funcionales.

★ Testing de Aceptación = Prueba realizada por el usuario para determinar si la aplicación se ajusta a sus necesidades.
Meta = establecer confianza / encontrar defectos no es foco.
Pruebas Alfa (ambiente de lab) Pruebas Beta (ambiente de trabajo real)

Proceso de Pruebas



Ejemplo Etapas / Entregables



Proceso de Testing

Planificación y Control = La planificación de pruebas es la act. de verificar que se entiendan las metas y obj. del cliente, las partes interesadas, el proyecto y los riesgos de las pruebas que se pretenden abordar.

Test Plan = Riesgos y objetivos de testing + estrategia de testing + recursos + criterios de aceptación.

Controlar = Revisar los resultados de testing + test coverage y criterio de aceptación + toma de decisiones.

Identificación y Especificación

Revisión de la base de pruebas + Verificación de las especificaciones + Evaluar la testeabilidad de los req y los sistemas + Identificar datos necesarios + Diseño y priorización de casos de prueba + Diseño de entorno de prueba.

Ejecución

Desarrollar y dar prioridad a TC + Crear datos de pruebas + Automatizar lo necesario + Creación de conjunto de pruebas + Implementar y verificar ambiente + Ejecutar casos de pruebas + Registrar resultado de ejecución de los TC + Comparar resultados reales con los esperados.

Evaluación y Reporte

Evaluar criterios de aceptación + Reporte de resultados para los involucrados (stakeholders) + Verificación de entregables y defectos que hayan sido corregidos + Evaluación de resultados de actividad de testing y lecciones

Caso de Prueba (CP) Son de condiciones o variables bajo las cuales un tester determinará si el sw está funcionando correctamente o no. Buena definición de CP nos ayudan a reproducir defectos.

Condición de prueba = Reacción esperada de un sw frente a un estímulo particular. Una condición debe probar al menos un CP.

"Caso de Prueba es la unidad de la actividad de la Prueba" { Objetivo / Dato de entrada y ambiente / Comportamiento esperado

"Los BUGs se esconden en las esquinas y se congregan en los límites"

Derivación de Caso de Prueba { Doc de cambios / Info de relevamiento / Requerimiento / Esp. de programación / Código

Estrategias [Caja Negra / Caja Blanca] Ninguna es completa, atacan distintos problemas, es mejor combinarlas. Depende de lo que se va a testear

Ciclo de Teste: Un ciclo de pruebas abarca la ejecución de la totalidad de los CP establecidos, aplicados a una misma versión del sistema a probar

Regresión = Al concluir un ciclo de pruebas y reemplazarse la versión del sist sometido al mismo, debe realizarse una verificación total de la nueva versión, a fin de prevenir la introducción de nuevos defectos al intentar solucionar los detectados.

Smoke Teste: Primer corrida de los test que prueba cierto aseguramiento de que el sw que está siendo probado no provoca falla catastrófica

Tipos de Pruebas

- Testing Funcional = Las pruebas se basan en Funciones y Características basados en Req y Procesos de Negocio
- Testing No Funcional = Prueba "cómo" Funciona el Sist.
  (Performance Testing, Carga, Stress, Usability, Mantenimiento, Fiabilidad, Portabilidad)

Verificación = ¿Estoy construyendo el sistema correctamente? → Se construyó contra los requerimientos

Validación = ¿Estoy construyendo el sistema correcto? → Se construye con el cliente.

Testing de Manera Temprana → Anticipación de corrección a probar un producto
↳ Disminuir costes de corrección de defectos

TDD: Test Driven Development (Kent Beck)

El código, el señor Test es uno de los mecanismos conocidos más efectivos para generar valor ... El proceso manual que debe desarrollarse para tener tests útiles puede descubrir y eliminar problemas en todas las etapas del desarrollo ..."

B. Beizer

* Escribir las Pruebas Primero (Se utilizan Pruebas Unitarias)

Testing en Proyectos Agiles (Principios)

1. Testing se Mide Adecuadamente en el Proyecto
2. Testing no es una Fase
3. Todos Hacen Testing
4. Producir la Latencia de Feedback
5. Las Pruebas Representan Expectativas
6. Mantener el Código Limpio, corregir los defectos rápidos
7. Reducir la sobrecarga de Documentación de las Pruebas
8. Las Pruebas son parte del "Done"
9. Se Prueba al Fin a "Colocado por Pruebas"

6 Prácticas Concretas

- Pruebas de Unidad e Integración Automatizadas
- TDD
- Pruebas de Regresión a Nivel de Sist Automatizadas
- Pruebas Exploratorias
- ATDD (Conducidas por Pruebas de Aceptación)
- Control de Versiones de Pruebas con el Código

Estrategias de Caja Negra = Basado en Especificaciones (Partición de Clases, Análisis Valor Límite)
↳ " en la Experiencia (Adivinanza de Defectos, Exploratorio)

Estrategias de Caja Blanca = Basado en Análisis de Estructura Interna del SW
↳ Garantiza testing Cobertura (Técnicas Básicas, Sentencias, Decisiones, Condición, Sentencia/Decisión, Múltiples)

Componentes de un Proyecto de Software

Proceso de Software = Procedimientos y Métodos

- Personas con habilidades, entrenamiento y motivación ← Herramientas y Equipos

- **Proceso Definido** = Inspirados en línea de producción

  [Masa Entrada] → [Proceso Definido] → [Masa Salida]

- **Proceso Empírico** = Adaptar → Asumir → Construir

  Revisar ← Retroalimentar

- **Ágil**
  - Balance entre ningún proceso y demasiado proceso
  - Adaptables en lugar de predictivos
  - Orientado a la gente no a procesos

Proyecto
- Orientados a objetivos (obtener resultados, guían al proyecto, alcanzable)
- Tienen duración de tiempo con principio y fin
- Únicos ("No son tan parecidos para ser diferentes")
- Implican tareas interrelacionadas basadas en esfuerzos y recursos (complejidad sistémica de los proyectos)

- **Administración de Proyecto** = Aplicación de conocimientos, habilidades, herramientas y técnicas a las actividades del proyecto para satisfacer los req. del proyecto
  - ① Identificar req. ② Establecer objetivos claros y alcanzables ③ Adaptar las especificaciones, planes y el enfoque a los diferentes intereses de los involucrados

- **La Triple Restricción**
  - → **Objetivos** del proyecto: ¿Qué se está tratando de alcanzar?
  - → **Tiempo** = ¿Cuánto debería llevar completarlo?
  - → **Costo** = Cuánto debería costar

  Afectan la calidad

  El líder de proyecto balancea estos 3 objetivos

- **Equipo de Proyecto** = Grupo de personas comprometidas en alcanzar un conjunto de objetivos de los cuales son mutuamente responsables
  - **Características**
    - Diversos conocimientos y habilidades
    - Desarrollar sinergia
    - Grupo pequeño 7 ± 2
    - Responsabilidad como unidad

- **Plan de Proyecto**
  - Documenta ¿Qué hacemos? ¿Cuándo lo hacemos? ¿Cómo lo hacemos? ¿Quién lo va a hacer?

Planificación de Proyectos de SW
- Definir alcance del proyecto
- Definición de proceso y ciclo de vida
- Estimación
- Gestión de riesgos
- Asignación de recursos
- Programación de proyectos
- Definición de controles

Alcance
- De Producto = Todas las características que debe tener el Producto o Software según lo descrito en las especificaciones requeridas
- De Proyecto = Todo el trabajo y solo el trabajo que debe hacerse para entregar el Producto o Software con todas las características y funciones especificadas. Según el Plan de Proyecto

Ciclo de Vida = Descripción del Producto que se va a construir

Estimación de SW = Tamaño + Esfuerzo + Calendario + Costos + Recurso

Riesgo = Problemas potenciales para software, evento que podría comprometer el éxito del proyecto
Gestión = Identificar y controlar Riesgos en todas las fases del Proyecto

Factores para el Éxito de un proyecto = { Monitoreo y Feedback, Misión/Objetivo Claro, Compromiso }

Fracaso de Proyecto
① Fallas de Definición Requisitos ② Planificación basados en datos insuficientes
③ No hay seguimiento del Plan de Proyecto ④ Plan de Proyecto poco detallado
⑤ Planificación de Recursos Inadecuada ⑥ Estimaciones basadas en "Supuestos"
⑦ NADIE ESTABA A CARGO

User Stories = Lo que escuchó en la tarea no es importante lo que usted hizo (Jeff Patton)

Desarrollo Ágil de SW = Un compromiso útil entre Modelo de Proceso y Nada Proceso (Fowler)

"La parte más difícil de construir SW es decidir precisamente qué construir. Ninguna otra parte del trabajo conceptual es tan difícil como establecer los requisitos técnicos detallados... Ninguna otra parte afecta tanto al sistema resultante si se hace incorrectamente. Ninguna otra parte es tan difícil rectificar más adelante"

No hay Balas de Plata - Escoger y Aceptar los SW

→ Marcadores conversacionales más detallados y análisis que deberán ocurrir

User Story

- Necesidad de Usuario
- Descripción de Producto
- Item de Planificación
- Token de Conversación
- Mecanismo para diferir una conversación

[Yo como <Rol usuario>
Yo puedo <Actividad>
De forma tal que <Valor de negocio recibido>]

[El product Owner prioriza las historias en el Product Backlog]

Modelado de Roles = Actividades de Usuario

→ Personas = Características y Forma de Pensar

Criterio de Aceptación de User Stories → Pruebas de Aceptación

- No son especificaciones detalladas
- Son expresiones de intención
- No están definidos a priori del proyecto
- Necesitan poco o ningún mantenimiento y pueden desecharse después de la implementación
- Junto con el código sirven de entrada a la documentación que se desarrolla incrementalmente después.

Spikes = Tipo especial de historia utilizado para quitar riesgo o incertidumbre de una User Story u otra faceta del Proyecto. Se clasifican en técnicos y funcionales. Pueden utilizarse para = Familiarizar equipo con nueva técnica o dominio, Analizar un comportamiento de una historia compleja, Ganar confianza frente a riesgos tecnológicos investigando o prototipando frente a riesgos funcionales donde no está claro cómo el sistema debe resolverlo

- **Técnicos** =
  - Evaluar performance potencial, Decisión hacer o comprar, Evaluar implementación de cierta tecnología. (Investigar)
  - Comprensión antes de comprometerse con una funcionalidad en tiempo fijo
- **Funcionales** =
  - Interacción cuando el usuario interactúa con el sistema
  - Son mejor evaluados con prototipos para obtener retroalimentación de los usuarios o involucrados

[Utilizar Spikes como última opción]

Estimación Ágiles = En los equipos ágiles las features/stories son estimadas usando una medida de tamaño relativo conocida como Story Points (SP)
Las medidas son relativas no absolutas
Story Points no es una medida basada en tiempo

El tamaño indica: Cuán compleja es una feature/story, Cuánto trabajo es requerido
Cuán grande es una feature/story
→ Define una escala, si se cambia se cambia el patrón

Story Point = Unidad de medida específica (del equipo) de complejidad, riesgo y esfuerzo

Velocity = Medida métrica del progreso de un equipo. Se calcula sumando el nº de story point (asignado a cada user story) que el equipo completa durante la iteración

User Stories Priorizadas → Estimación de Tamaño en Story Points → Derivar Duración → Release Plan

FALLA - Error en un producto de trabajo (salida de cualquier actividad correspondiente al ciclo de vida de desarrollo)

→ Porque existen fallas:
- Ruido de comunicación
- Limitaciones de memoria (7 ± 2)

"Las fallas más persistentes están relacionadas con la complejidad inherente al producto que se desarrolla"

→ **Clasificación**
- **Defecto (mayor)** = Originado en el producto de trabajo predecesor
- **Error** = Originado en el producto de trabajo que se está inspeccionando
  - **Mayor** = Condición que puede causar falla operacional o producir un resultado inesperado durante la ejecución de la operación especificada
  - **Menor** = Condición no deseable en el producto no puede ocasionar una falla operacional
  - **Cosmético** = Error de tipeo, ortografía, errores menores que no se cuentan como fallas.

Principios = ① La prevención es mejor que la cura ② Evitar es más efectivo que eliminar

③ La retroalimentación enseña efectivamente ④ Priorizar lo rentable

⑤ Olvidarse de la perfección, no se puede conseguir ⑥ Enseñar a revisar

Revisiones Técnicas - Proceso de verificación y validación estático, principal objetivo de detectar defectos y corregirlos en las etapas tempranas de desarrollo.

Se puede inspeccionar cualquier representación de su lógica. Se aplican en varios momentos del desarrollo. Motiva a realizar un mejor trabajo. No requiere que el programa se ejecute

→ **Ventajas** = Descubrir muchos errores, inspeccionar versiones incompletas, considerarse otro atributo de calidad

→ **Desventajas** = Difícil de introducir inspecciones formales, sobrecargan al inicio de los costos y conducen a un ahorro cuando los equipos adquieren experiencia de usarlo., Requieren tiempo en reorganizarse y reorientar el proceso de desarrollo.

Métodos
- **Walkthroughs** = Mínima sobrecarga capacitación de desarrolladores rápido retorno (Proceso informal, no hay métricas)
- **Inspecciones** = Detectar y remover todos los defectos eficiente y efectivamente (Proceso formal, métricas, fases de planificación)

Métricas

Densidad de defectos = Total de defectos encontrados / Tamaño actual

Total de defectos encontrados = Defectos mayor + Defecto menor

Esfuerzo de la inspección = + Esfuerzo planning + Esfuerzo preparación + Esfuerzo reunión + Esfuerzo retrabajo

Defectos corregidos sobre el total de los defectos = Esfuerzo inspección / Tamaño actual

Esfuerzo por defecto = Esfuerzo inspección / Total de defectos encontrados

Porcentaje de reinspecciones = Cant. de reinspecciones / Cantidad inspecciones

Inspección

- Roles = Autor, Moderador, Anotador, Lector, Inspector
- Procesos = Planificación → Visión general → Preparación → Reunión insp. → Corrección → Seguimiento
- Mejores resultados, procesos comunes, métricas útiles

Recorridos/Walkthroughs = Buenos resultados, pocas métricas, no hay control de proceso

Paper = No hay Balas de Plata; Esencia y Accidentes de Ing. de SW

Ningún invento podrá hacer para la productividad, confiabilidad y simplicidad del SW lo que la electrónica, transistores e integraciones a gran escala hicieron para el HW

La anomalía no es el progreso de SW lento, sino que el progreso de SW es rápido

La esencia de una entidad de SW es una construcción de conceptos interrelacionados

Lo difícil de hacer SW es la especificación, diseño y prueba

Propiedades

- Complejidad = Dos SW es una propiedad esencial, no accidental. Viene de la comunicación de los miembros del equipo que lleva a deficiencias del producto, excesos de costo y atrasos
- Adaptabilidad = El SW debe adaptarse porque es el último en llegar. Adaptación a otras interfaces
- Modificabilidad = Todos los elementos cambian continuamente y sus cambios usualmente fuerzan modificaciones al producto de SW
- Invisibilidad = A pesar del progreso en las estructuras y simplificaciones de las estructuras de SW, estas son inherentemente no visualizables y por ende impide usar una de las herramientas conceptuales más poderosas de la mente. Dificultando la comunicación entre las mentes

Ataques prometedores a la esencia conceptual del SW

Mayor ataque tiene la solución ninguna, pero todos están limitados por la ecuación de la productividad: Tiempo de ejecución = $\sum$ (Frecuencia) × (Tiempo)

- Comprar vs Construir = No construir en su totalidad ya que es mejor comprar herramientas de programación y añadirlas que construirlos
- Refinamiento de Req y Prototipos Rápidos = El cliente nunca sabe bien lo que quiere por lo que este enfoque de desarrollo a través de la validación de requerimientos refinados de manera iterativa con prototipos hace que obtengamos un SW más consistente y usable para el cliente

7

El secreto es que el Software se cultiva, no se construye.

L Diseñadores Brillantes = La cuestión central en cómo manejar el arte de SW se resuelve en la gente. Para cultivar buenos diseñadores es importante asignar a un seguidor por diseñador para que cree y mantenga un plan de desarrollo de carrera y que el diseñador interactúe con otros diseñadores.

Paper: A view of 20th and 21st Century Software Engineering

Visión Hegeliana: Tesis — Antítesis / Síntesis

1950 Tesis: Ing de SW es como Ing de HW
1960 Antítesis: Elaboración de SW = Mejor infraestructura, aplicaciones pequeñas manejables, comienzan empresas de desarrollo de SW

1970 Síntesis y Antítesis: Formalidad y cascada = Éxito de programación estructurada, bastantes estructurados
1980 Síntesis: La productividad y la escalabilidad = enfoques en procesos de lenguajes de codificación. Reutilización de SW

1990 Antítesis: Concurrente vs Secuencial = Métodos orientados a objetos como patrones de diseño. Desarrollo de código abierto. Usabilidad en interacción Humano-Computadora

2000 Antítesis / Síntesis Parcial: Agilidad y Valor = Tendencia a desarrollo rápido de aplicaciones. Métodos ágiles. Valor para el usuario. COTS, de código abierto y su legado. Desarrollo MDD

2010 → Más Allá:

2010 Antítesis y Síntesis Parciales: Globalización y Sistemas de Sistemas = Integración de la Ing de SW con los sistemas de marketing, finanzas son críticos.

(1) Al mismo tiempo de diseñar y validar la consistencia y viabilidad de la OP, Req, Arq. Prototipos
(2) Iteraciones cortas de desarrollo con personas eficientes y efectivas
(3) Verificación y Validación Continua (V&V)
(4) Espíritu Ágil — Adaptarse rápidamente al cambio

2020 → Más allá: Autonomía y Bio-informática "Porque el futuro no es lo que solía ser"

Conclusiones: No olvidarse de las ciencias. Mirar antes de saltar. Evitar uso de proceso secuencial riguroso. Pensar fuera de la caja. Diferencias de SW. Evitar programación vaquero. Eliminar errores de manera temprana. Determinar propósito del sistema. Evitar desarrollo arriba hacia abajo. Mejor productividad. Bueno para el producto es bueno para el proceso. Esotéricos a balas de plata. Tiempo es dinero. SW útil para las personas. Ser rápido pero sin prisas. Cambio rápido → adaptabilidad. Cuidar y satisfacer todas las propuestas de valor de las partes interesadas. Alcance dentro del alcance. Tener estrategia de salida. No creer todo lo que lee