RESUMEN ISW 2DO PARCIAL TEÓRICO

Lean

- Es un modelo de gestión
- Es un proceso sistemático
- Su precursor fue Toyota, pero no en el área de Software
- Busca la mejora de procesos a través de la disciplina
- Propone maximizar el valor para los clientes, reduciendo los costes/desperdicios y aumentando la calidad del producto o del servicio
- Su objetivo es eliminar desperdicios, seleccionando aquellas características que realmente aportan valor, y da especial importancia a la velocidad y la eficiencia (disciplina)
- Propone una respuesta muy rápida y de manera muy disciplinada a la demanda del mercado, en lugar de tratar de predecir el futuro
- Se enfoca en eliminar las tareas que no aporten valor sobre aquello que estamos realizando
- Visualiza todo el proceso

- Un desperdicio podría ser:
  - En manufactura
    - Stock sobrante
    - Horas de trabajo perdidas
    - Retrasos
    - Sobrante de materia
    - Transporte
    - Esperas
    - Defectos
  - En desarrollo de Software
    - Características extras
    - Defectos
    - Procesos a medias

Sigue una serie de 7 principios

- Eliminar desperdicios
  - Hace referencia a todos los tiempos muertos, recursos no usados, o procesos/actividades que no aportan valor para el cliente (incluyendo el re-trabajo)
  - Desperdicio es aquello que consume tiempo/recursos y no aporta un valor que el cliente perciba
  - Relacionado con “Maximizar el trabajo no hecho” de Agile
  - En la manufactura, puede ser el inventario
  - En Software, puede ser el trabajo parcialmente hecho y las características extra

- Amplificar el aprendizaje
  - Se basa en mantener una cultura de constante mejora
  - El nuevo conocimiento debe volverse disponible para toda la organización
  - Debe buscarse constantemente soluciones a los problemas encontrados
  - Con cada conocimiento adquirido y solución ideada, se cuenta con una base mayor de conocimientos para los próximos problemas
  - Los procesos anticuados o definidos pueden dificultar la mejora de los mismos
  - No se debe definir procesos de antemano, si no ir detallándolos a medida que se obtiene conocimiento de ellos

- Embeber la integridad conceptual
  - Encastrar todas las partes del producto o servicio, que tenga coherencia y consistencia entre si
  - Esta relacionado con los Requerimientos No Funcionales
  - Se busca una integración continua de los diferentes componentes del producto (cada artefacto del proyecto)

- La integración entre las personas hace el producto más integro
- Hay dos tipos de integridad:
  - Integridad Percibida
    - El producto total tiene un balance entre función, uso, confiabilidad y economía que le gusta a la gente → Lo que ve el cliente, la fachada, lo que produce verlo
  - Integridad Conceptual
    - Todos los componentes del sistema trabajan en forma coherente en conjunto → Integridad real en la arquitectura del Software
- El objetivo es construir con calidad desde el principio, no probar después.
- Dos clases de inspecciones:
  - Inspecciones luego de que los defectos ocurren.
  - Inspecciones para prevenir defectos.
- Si se quiere calidad, no inspeccione después de los hechos
- Si no es posible, inspeccione luego de pasos pequeños

- Diferir Compromisos
  - Se basa en tomar las decisiones lo más tarde posible (sin comprometer el desarrollo del producto o su calidad)
  - De esta forma, se tomaran las decisiones sobre la mayor cantidad posible de información, y no con información incompleta
  - Relacionado con el principio ágil “decidir lo más tarde posible pero
  - responsablemente”
  - Busca dejar abiertas la mayor cantidad de opciones posibles
  - Se relaciona con el principio anterior (entre mas tarde decidimos, mas conocimiento tenemos)
  - Además, la decisión sera mas acertada cuanta mas información tengamos
  - En lo posible, estas decisiones deben poder revertirse

- Dar poder al equipo
  - Se basa en confiar en el equipo de desarrollo/producción, de manera que ellos puedan manejarse de la manera que les resulte mas cómoda/eficiente
  - El responsable de la producción no debe meterse en todos los temas relacionados a la misma, el equipo decidirá ciertas cosas en base a su experiencia y su forma de trabajo
  - Se debe entrenar líderes para que lideren los equipos
  - Se fomenta una buena ética laboral
  - Se delegan decisiones y responsabilidades del producto en desarrollo a los equipos/lideres de nivel más bajo posible
  - Relacionado con el principio Agile “Las mejores arquitecturas/diseños surgen de equipos auto-organizados”

- Ver el todo
  - Se debe trabajar teniendo en cuenta una visión entera del producto, relacionando cada proceso/parte/componente/etc
  - No se debe trabajar solo enfocándose en la actividad que le toca a cada parte, si no que se debe comprender y analizar la relación entre cada elemento del proyecto/producto (personas, procesos, funcionalidades, etc)
  - De esta forma el desarrollo se hará teniendo en cuenta posibles mejoras o defectos relacionados a otras partes del producto

- Entregar lo antes posible
  - Se debe entregar al cliente un producto lo antes posible, de manera que el cliente no pierda tiempo, y el equipo de desarrollo aprenda rápidamente la opinión del mismo, pudiendo modificar o mejorar el producto según estas opiniones
  - Ademas, mientras antes se corrija una parte del producto, menor será el esfuerzo dedicado a ello
  - Se debe entregar rápido (trabajar en un entorno que facilite una entrega rápida del producto, con un equipo eficiente y pocos tiempos de desarrollo) → Relacionado a la velocidad de desarrollo
  - Se debe entregar rápidamente (las entregas deben darse con mucha frecuencia, de manera que se

tengan muchas entregas que aporten un incremento de valor pequeño) → Relacionado a la frecuencia de entrega

Scrum

- Es un marco de trabajo/framework para la gestión de proyectos (de cualquier naturaleza), que permite crear productos de manera es simple y metódica, a través de un proceso de desarrollo empírico, que define un equipo, tareas y elementos en el mismo
- Es una forma de llevar a cabo un proyecto según un conjunto de buenas practicas para desarrollar el mismo
- Define claramente los componentes de este desarrollo, pero no así las etapas o los procedimientos que se deben dar para desarrollar el producto, ni el orden de los mismos
- Esta basado en metodologías ágiles, por lo que le da una importancia primordial al equipo
- Es fácil de entender, pero difícil de dominar
- Se basa en inspecciones y correcciones frecuentes dentro de las iteraciones en las que se desarrolla el producto
- Se basa en la entrega parcial y frecuente de un producto final
- Se suele aplicar donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales

Consta de tres roles fundamentales

Product Owner

- Representa a la organización o el cliente que recibirá el software
- Es una sola persona, no un comité
- Es el encargado de transmitir/comunicar la parte de negocio del producto, así como de asegurarse que se este desarrollando el producto que realmente se requiere (es el encargado de aceptar/rechazar cada resultado de las iteraciones)
- Traslada la visión de la organización al equipo
- También se encarga de priorizar y administrar el Product Backlog → Es el único que puede priorizar las actividades del equipo, y nadie más que el puede cambiarlas
- Se encarga de maximizar el valor del producto
- Debe entender perfectamente que es lo que se desea del producto en todo momento, debiendo poder explicar y trasmitir a los stakeholders cuál es el valor del producto en el que están invirtiendo, y al equipo de desarrollo que es específicamente lo que se desea
- Debe poder tomar cualquier decisión que afecte al desarrollo del producto en una iteración

Scrum Master

- Dos funciones principales: asegurarse de que se lleve a cabo correctamente la metodología Scrum y ayudar a eliminar impedimentos que puedan afectar a la entrega del producto (cuando cuente con los conocimientos/herramientas necesarias para ello)
- Además, se puede encargar del apoyo, aconsejamiento o mentoreo de los integrantes del equipo de desarrollo
- Debe facilitar la ejecución del proceso Scrum y sus mecánicas, controlando la ejecución en tiempo y forma de los rituales
- Puede cambiar durante el desarrollo del proyecto, y su cargo suele ser ocupado por una persona que conoce o tiene experiencia trabajando con esta metodología
- Se relaciona ampliamente con el PO, y es el nexo entre la gerencia y el equipo

Scrum DevTeam

- Está formado entre 5 y 9 integrantes que se encargan de manera conjunta de desarrollar el producto
- La gestión y organización de equipo es tarea del equipo mismo, de manera que no responden a un líder ni un jefe, si no que definen ellos mismos de que manera trabajar y como repartir las tareas
- No esta compuesto por roles, ya que cada integrante debe poder resolver cualquier necesidad relacionada al desarrollo del producto (cada integrante debería poder implementar cualquier PBI de

principio a fin)

- El equipo de desarrollo está formado por el Scrum DevTeam y el Scrum Master
- El equipo Scrum esta formado por el Product Owner, el Scrum DevTeam y el Scrum Master

- Scrum trabaja con User Stories, que son descripciones de una funcionalidad deseadas por un usuario
- Para poder implementar estas US, deben estar completamente detalladas, y deben ser lo suficientemente pequeña como para poder implementarlas en una sola iteración
- El equipo puede estimar el esfuerzo que conllevara implementar esa US a través de distintas técnicas

Product Backlog

- Es un contenedor de Product Backlog Items, que representan características/funcionalidades que se implementaran en el producto (principalmente son US, pero puede haber Epics, Themes y Spikes, que son un tipo especial de US)
- Incluye los requerimientos del SW
- En un comienzo, estará incompleto, y se ira completando en el desarrollo del proyecto
- Se crea en base a las necesidades de todas las áreas de la organización que vayan a interactuar con el producto final
- En él se guardan todas las US que aún no se han implementado en el proyecto
- Las US en el PB están priorizadas (por el PO), de manera que las mas prioritarias (y mas detalladas) estarán en la cima, y serán las próximas a implementarse en una iteración
- En el fondo estarán las US que no están lo suficientemente detalladas, o son muy grandes para implementarlas, o simplemente no son una prioridad para el equipo

En cada iteración se trabaja con un Sprint Backlog, que contiene el conjunto de US que el equipo intentará implementar en la iteración

Scrum trabaja a través de iteraciones, llamadas Sprints
Estos Sprints duran como máximo 30 días, y en ellos no debe cambiarse jamás el objetivo

Los Sprints están dados de la siguiente forma:

- Lo primero que se hace al comenzar un S es el Sprint Planning
- Se planea que US se desarrollaran en el Sprint, el objetivo del Sprint, se reparten los trabajos y se define que tareas se deberán llevar a cabo para cumplir con el objetivo
- También define que US se van a implementar (define el Sprint Backlog de ese Sprint)
- Hace uso del IP anterior, el PB y las métricas para poder estimar y decidir que se va a hacer
- Para un Sprint de 4 semanas, no puede durar más de 8 horas

- Durante todo el Sprint, se llevan a cabo diariamente las Daily Meetings
- En ellas, el equipo se reúne y de manera breve, cada integrante explica:
  - Que hice desde la ultima DM?
  - Que planeo hacer para la próxima?
  - Que impedimentos tuve en mis actividades?
- Son de asistencia obligatoria para el equipo de desarrollo
- El PO puede estar presente o no
- Su principal objetivo es logar una sincronización y visibilidad en el equipo, dejando en claro en que esta trabajando cada uno para lograr el objetivo del Sprint
- Busca lograr una visión y una responsabilidad compartida del producto
- Los resultados de las DM no son documentados, si no que se llevan a cabo de manera informal
- No deben durar más de 15 minutos

- Al terminar el Sprint, se lleva a cabo la Sprint Review (o Demo), en donde se le presenta al PO (y a los stakeholders invitados) el resultado de la iteración
- Debe estar presente todo el equipo Scrum

- Este resultado representa el incremento de valor que se desarrollo para el producto, llamado Incremento de Producto
- Este IP debe poder ponerse en producción, por eso también se llama Potencialmente Implementable
- Esta ceremonia se basa en la inspección del IP por parte del PO y la adaptación del mismo por parte del equipo
- El PO decidirá si acepta el IP (y lo comienza a producir/implementar) o si lo rechaza
- En esta ceremonia, el equipo obtiene feedback de su trabajo
- Para un Sprint de 4 semanas, no puede durar más de 8 horas

- La ultima ceremonia es la Sprint Restrospective, que también se basa en la inspección y adaptación, pero del producto entero
- En ella están presentes el equipo de desarrollo, y puede o no estar el PO
- Se basa en un feedback/retroalimentación interna, en donde el equipo de desarrollo analizara lo que paso en el Sprint, los resultados obtenidos, las posibles modificaciones, lo que se hizo bien/mal, etc
- Se obtiene una lista de elementos para implementar en el futuro del proyecto (en el/los próximo/s sprint), de manera informal
- Para un Sprint de 4 semanas, no puede durar más de 3 horas

- También existe una actividad opcional (es decir, que no forma parte del proceso Scrum oficial) llamada Product Backlog Refinement, o PB Grooming
- En ella, el equipo Scrum revisa el PB, de manera que todos entiendan o detallen mas las US
- Además, el PO puede redefinir las prioridades de los PBI
- Debe durar como máximo 2 horas por semana

En conclusión, Scrum cuenta con 3 artefactos (PB, SB, IP), 3 roles (SM, SDT, PO) y 4 ceremonias (SP, DM, Sreview, SRetrospective)

Scrum esta formado por 5 cimientos:

- Time Boxing
  - Todas las actividades/rituales en Scrum tienen una duración (plazo máximo) fija, y su tiempo no se negocia
- Auto-organización
  - Se trabaja con equipos auto-organizados de 5 a 9 personas (si tuviese más personas, las divido en dos equipos)
  - El equipo es el que estima, decide, valora, planea, por si mismo para si mismo
  - El equipo no tiene un líder (el SM solo sirve como una guiá y una ayuda, pero no define que actividades hará cada integrante del equipo ni como las deberá hacer)
  - Asume que cada integrante esta comprometido y motivado en el proyecto
  - El SM es elegido por el equipo de desarrollo, y puede ir rotando entre distintos integrantes
  - El SM puede o no hacer trabajo técnico (si sabe hacerlo, si tiene tiempo, etc)
- Colaboración
  - Los integrantes del equipo trabajan colaborando entre si para aprovechar al máximo la potencia del equipo
- Priorización
  - Como Scrum trabaja con iteraciones cortas, es importante definir en cada momento las prioridades de las actividades, para saber cuales hacer primero y cuales ultimo
  - Define que se entrega ahora y que se entregara en el futuro (orden de los trabajos/entregas)
- Empirismo
  - Al basarse en un proceso empírico, hace uso del empirismo, y se basa en la experiencia del equipo y de las decisiones tomadas para la toma de decisiones

Scrum cuenta con 5 valores, que son características o virtudes que los integrantes del equipo deben tener para desarrollar con éxito un proyecto

- Compromiso → En realizar las tareas de una forma adecuada, y compartir la responsabilidad del éxito del proyecto. Dedicarse a crear el mejor producto posible

- Respeto → Por los demás compañeros, para crear un clima de trabajo ideal. Profesionalidad y respeto por las opiniones ajenas
- Apertura → En las actividades de sincronización/visibilidad, se espera que los integrantes sean sinceros y abiertos, para transmitir fielmente su situación actual a los demás compañeros y a los interesados en el proyecto. Los integrantes deben estar dispuestos a compartir lo que saben/hacen/etc
- Foco → Se debe trabajar enfocado y sin perder tiempo en cosas que no aporten valor. Se debe tener claro el foco en el objetivo del sprint y no perderlo de vista. No divagar en las tareas, terminar lo que se empieza.
- Coraje → No se debe temer a la critica o la vergüenza, se debe admitir errores o cosas que no sabe hacer. Aceptar desafíos y dejar la zona de confort

En Scrum, se trabaja con tiempo y costo fijos → El mayor costo son las personas, y se las contrata a todas por toda la duración del proyecto, por lo que no varia ese costo
Tampoco varia el tiempo, se cumplen siempre los plazos definidos para las entregas
Lo que puede variar es el alcance (analizando el costo y el tiempo) en cada entrega, aunque no es deseado y debería evitarse lo mas posible. Siempre se va a entregar algo, pero puede variar con lo que prometimos

Forma de trabajo
- Se comienza a trabajar (se hace el primer sprint) cuando tengo suficientes US en estado Ready (listas para implementarlas) → Ready debe cumplir mínimamente con el modelo INVEST, pero cualquier equipo puede agregar condiciones
- Solo se agregan US al SB cuando están Ready
- La condición de DONE esta dada por cuando el PO aceptara la US (se revisa en el IP presentado en el SReview)
- Un Sprint nunca cambiará de objetivo. Se mantiene fijo el objetivo y el plazo, no varían (no se podrá agregar funcionalidades a implementar).
- En situaciones particulares en las que no tenga sentido seguir con el Sprint (ya sea porque cambiaron los requerimientos de negocios, porque el PO decide que no tiene valor seguir con el Sprint, porque cambiaron las condiciones técnicas, porque el equipo estimo mal y se da cuenta de que no podrá completar el objetivo), el mismo puede cancelarse, y todas las US del SB volverán al PB (según el PO lo defina)
- De igual forma, cancelar un Sprint conlleva un costo grande de recursos y tiempo (y de credibilidad por parte del PO), por lo que no se deben cancelar en lo posible
- Durante un Sprint puede variar el PB.
- Es clave para aplicar Scrum que sea posible un nivel de interacción muy alto con el cliente/organización, o un representante de la misma. Si no se puede cumplir ese requerimiento, no se debería aplicar Scrum
- De la misma forma, el representante del negocio debe ser alguien que lo conoce profundamente y puede representar en sus decisiones la voluntad de toda la organización

Métricas Ágiles
- En Agile se trata de medir solo lo necesario, no perder tiempo realizando mediciones que no aportan valor
- Estas métricas deben ser una salida de un proceso, no una actividad puntual que se realiza
- Las métricas ágiles se rigen por dos principios de Agile:
  - “Nuestra mayor prioridad es satisfacer al cliente por medio de entregas tempranas y continuas de software valioso." → Con las métricas veo si estoy cumpliendo este principio o no
  - “El Software trabajando es la principal medida de progreso.” → Se mide realmente la cantidad de SW que funciona/acepta el PO

Capacidad
- No es realmente una métrica, porque no surge de la medición de algo, si no que es una estimación
- Estima cuando trabajo cree el equipo que podrá realizar en un periodo de tiempo dado (normalmente un sprint)
- Definirá cuanto tomar del PB para desarrollarlo en el Sprint
- Se debe estimar de manera realista y sincera, ya que en base a esta estimación se planteara el compromiso del objetivo del sprint

- Se estima en SP o en horas ideales (conviene medirla en SP, porque los PBI están medidos en SP, por lo que es mas fácil y acertado estimar)
- Es mas costoso para equipos sin experiencia definir su propia capacidad
- Horas ideales → Son horas de trabajo efectivo, no solo horas en el lugar de trabajo (horas en las que la persona esta trabajando realmente) Aproximadamente 4/5hs por cada 8hs de trabajo diario (normalmente representan el 50-70% de las horas de trabajo)
- Al iniciar un sprint, se toma en cuenta la capacidad del equipo
- La capacidad puede variar de sprint a sprint, por distintos motivos relacionados al equipo (vacaciones, faltas, otros proyectos paralelos etc), por eso es importante conocer la disponibilidad de los integrantes de antemano para definirla
- Cada integrante debe declarar cuantas horas puede comprometerse con el proyecto, y se estima la capacidad del mismo
- Va a estar conformada por la cantidad de horas ideales por día, multiplicada por la cantidad de días en el sprint, multiplicada por cada miembro del equipo

Velocidad

- Es realmente una medición, se mide (a través de un calculo) la capacidad real del equipo en cada iteración
- Esta dada por la cantidad de US que el PO acepto (no tiene en cuenta las que se desarrollaron pero el PO no acepto) en un periodo de tiempo
- Sirve para retroalimentar (y hacer mas certera) la estimación de la capacidad del equipo → Se podría decir que la Capacidad es una estimación de la Velocidad
- Se mide en SP
- Se mide al final del Sprint, determinando que velocidad tuvo el equipo en esa iteración
- Se puede comparar entre iteraciones y entre proyectos (siempre y cuando se trate del mismo equipo, porque dos equipos diferentes estimaran según distintos valores de SP a cada US)
- Con el paso del tiempo se va estabilizando (se visualiza con un grafico)
- Se consulta en el Sprint Planning para estimar la capacidad de ese sprint

Running Testing Factory (RTF)

- Muestra el crecimiento del producto en relación a las funcionalidades que tiene con el paso del tiempo
- Mide el progreso del producto en cantidad de features
- No refleja mucho el progreso real, porque una feature puede ser mucho mas importante/compleja que otra, y pesarían lo mismo

Herramientas en Scrum

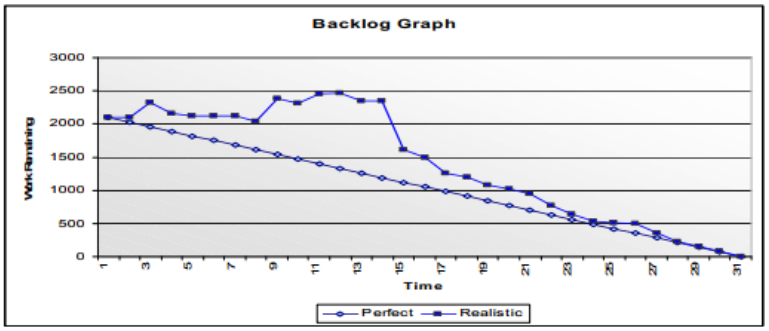
Tablero/Taskboard

- Es un tablero compuesto por tarjetas, que se van ubicando en distintas secciones del mismo
- Sirve para el equipo de desarrollo
- Las tarjetas representan las tareas dentro del sprint (las que están por hacerse las que se están haciendo y las que se hicieron)
- Cada tarjeta tiene:
  - El código del PBI al que hace referencia
  - Número del Sprint
  - Nombre de la Actividad
  - Iniciales de quien la realiza
  - Tiempo Estimado en Horas
- El tablero se divide en 5 secciones:
  - Story → Son US definidas, pero que aun no tienen lugar en ningún sprint
  - To Do → Tareas por hacer en el sprint
  - Work In Process → Tareas haciéndose en el sprint
  - To Verify → Tareas terminadas, que deben verificarse
  - Done → Tareas completadas y verificadas
- Se trata de que el SP tenga muchas US pequeñas, de manera que sea más granular, mas dividido, se nota más rápido el avance del proyecto

- De esta forma se van a tener muchas tarjetas en cada sección

Gráficos de Backlog

- Representan datos relacionados al progreso del proyecto (al sprint, al PB, al release, etc) que se pueden representar ante el equipo o los stakeholders
- Sirven para comparar lo que se esperaba con lo que realmente paso/se tiene
- Uno de esos gráficos es el Burndown Chart
- Sus ejes son:
  - Tiempo (en días) → Abscisas
  - Trabajo restante (en SP) → Coordenadas
- En él, se grafica una linea ideal y una linea real, que representa el avance del producto
- Idealmente, se tendrá una linea real muy parecida a la ideal
- De igual manera, dado que las US representan valores enteros distintos de uno, las dos lineas siempre serán distintas (la linea real no puede se runa linea con pendiente negativa constante)
- Se actualiza durante todo el sprint, no al final del mismo (por ejemplo, en cada Daily)
- La linea real suele estar constante (cuando no se terminó/quemo ninguna US ese día) o bajando (cuando se terminan US), pero también puede que suba (si se había considerado a una US como terminada, cuando en realidad no lo estaba)



Beneficios de Scrum

- Se pueden cambiar fácilmente los requerimientos o agregar nuevos
- El proceso se adapta a la prioridad que define la organización (producto personalizado y cliente satisfecho), y el equipo se auto-organiza para cumplir esas prioridades
- El desarrollo tiene en cuenta la visión de negocio/mercado
- Los clientes ven entregas con funcionalidades aplicadas con mucha frecuencia

El aseguramiento de la calidad del Software esta dado por tres disciplinas:

- Control de calidad del Proceso
- Control de la Calidad del Producto
- Pruebas de Software (Testing)

Testing de Software

- Es un proceso destructivo de desarrollo, porque su objetivo es encontrar defectos, lo que requiere una reestructuración del trabajo
- La calidad no es algo que se implementa en el final del proceso de desarrollo el mismo, si no que se va creando y evolucionando durante todo el desarrollo (mientras antes se detecten errores, mas fácil y barato es resolverlos)
- Ademas, la calidad compete tanto a la calidad del producto como a la del proyecto o proceso
- Se parte de la suposición de que siempre se tendrá defectos por encontrar → Es una característica inherente a los productos desarrollados por equipos de personas
- Los defectos son costosos de resolver, por lo que nos conviene detectarlos lo antes posible
- El Testing sirve para reducir riesgos, y obtener un producto más confiable

- Aun así, el Testing NO asegura que se tenga un producto de calidad, ni que el proceso por el que se desarrollo sea de calidad
- Verifica que el SW este bien implementado (cumple requerimientos), y sea correcto (sea el SW que el cliente quiere/necesita)
- El proceso de Testing pueda empezar antes de la codificación, cuando no tengo ni una linea de código (porque en ese momento ya cuento con los requerimientos, y ya puedo ir pensando en casos de prueba para probarlos)
- Su objetivo es encontrar defectos: no solucionarlos, no plantear posibles soluciones, no definir si se debe hacer así o de otra manera
- La persona que realiza el Testing no es un enemigo del desarrollador, si no que es mas bien un critico de su trabajo
- El Testing es exitoso si encontró defectos
- Representa aproximadamente el 30 – 50% del costo de desarrollo (para obtener un SW confiable)

Principios del Testing

- Excepto situaciones muy especificas, la persona que realiza el Testing no deberá ser la misma que desarrollo el componente
- Un componente no debería llevar a cabo pruebas para controlar su propio funcionamiento
- No solo debe cerciorarse de que el SW cumpla lo que debe hacer, si no que debe asegurarse de que no este haciendo lo que no debe hacer
- Al planificar el Testing, se debe asumir que se van a encontrar errores (no deberías planear pensando que no se van a encontrar)
- No existe el Testing exhaustivo (es decir, el Testing que prueba todas las posibilidades posibles en el funcionamiento del SW), debido a la cantidad de tiempo, esfuerzo y costo que implicaría
- Por eso se busca cubrir la mayor cantidad posible de las funcionalidades con cada caso de prueba
- No alcanza con saber que el sistema funciona correctamente
- El único parámetro para definir cuanto Testing es suficiente, es la confianza que se tiene de que el sistema funcione correctamente (viéndose afectada esta decisión por el riesgo permitido para el sistema, que depende de su naturaleza)
- A la hora de definir cuanto Testing es suficiente, se deberá tener en cuenta el nivel de riesgo que se puede permitir en el sistema y el costo asociado que lleva hacer mas Testing
- Los riesgos nos detallaran que debemos testear primero, que es mas importante, que priorizar a la hora de testear (y que cosas no son tan importantes)
- Paradoja del Pesticida → Al trabajar con Testing automatizado, las pruebas no podrán encontrar defectos que aparecieron luego de la creación de las pruebas
- El Testing se hace por el bien del producto, y no debe ofender a quienes lo desarrollaron

Criterio de Aceptación

- Para definir cuando dejar de testear una fase o el producto total, se puede usar un Criterio de Aceptación
- El CDA define lo que se debe cumplir para que se considere al software o la fase aceptadas
- Es definido junto con el cliente
- Puede medirse en:
  - Horas de testeo
  - Errores encontrados
  - Horas de testeo seguidas sin encontrar un error
  - Un nivel de cobertura (%) sobre el total del SW

Error

- Son las fallas que se encuentran mientras se esta desarrollando esa funcionalidad el producto
- Son propios de la etapa en la que se esta desarrollando (el producto de trabajo actual)

Defecto

- Son las fallas que se encuentran mientras se esta desarrollando una etapa siguiente del producto a la etapa en la que se desarrollo la funcionalidad que tiene la falla
- Son propios de la etapa anterior a en la que se esta desarrollando (el producto de trabajo anterior)

- Es más costoso corregirlo, ya que se deben corregir aspectos anteriores al SW, que pueden afectar al desarrollo posterior de la falla
- Los defectos pueden categorizarse en severidad (el impacto del defecto sobre todo el producto al ejercer pruebas) y en prioridad (que tan urgente es la resolución del defecto)
  - Severidad → Bloqueante, Crítico, Mayor, Menor, Cosmético
  - Prioridad → Urgencia, Alta, Media, Baja
- La S y la P de los defectos pueden o no tener una correspondencia marcada (aunque generalmente si la tengan)

Niveles de prueba/Testing

- Las pruebas que se realizan sobre el producto están divididas en distintos niveles, que se basan en la granularidad de esa prueba (que tan chicas e independientes son)

Pruebas Unitarias

- Son pruebas que se realizan sobre un componente, de manera independiente a los otros
- Se prueba la funcionalidad de un solo componente
- Suelen ser llevadas a cabo por el mismo desarrollador, una vez que se termino de construir el componente (por lo que se encuentran errores, no defectos)
- Se hacen teniendo acceso al código fuente, y se pueden usar herramientas para realizarlas (automatización, depuración)
- Se suelen reparar los errores apenas se encuentran, sin registrarlos formalmente
- Métodos, clases

Pruebas de Integración

- Hace referencia a las pruebas que se ejecutan para comprobar que los componentes del producto trabajen correctamente en conjunto
- Puede probar una cantidad chica o grande de componentes
- Se suele llevar a cabo una prueba de integración incremental, desde lo mas general (que abarca mas componentes) a lo mas especifico (top-down) o desde lo más especifico a lo mas general (bottom-up)
- Se deben probar lo antes posible aquellos conjuntos de componentes que sean críticos para el sistema
- Clases, paquetes, subsistemas

Pruebas de Sistema

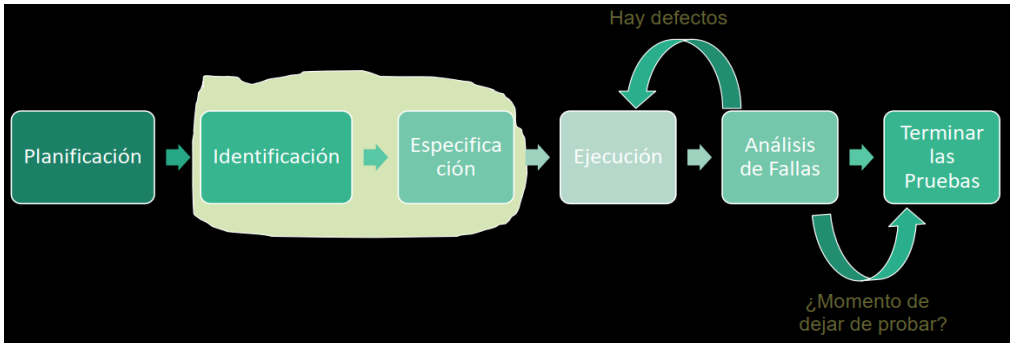
- Se prueba la funcionalidad de la aplicación completa y funcional
- Se trata de emular de la mejor manera posible un entorno de trabajo idéntico al entrono real en el que se usara el SW
- Se llevan a cabo pruebas del funcionamiento real y cotidiano del producto
- Busca definir si el sistema trabaja correctamente (seguridad, recupero de fallas, stress, etc)
- Abarca requerimientos funcionales y no funcionales

Pruebas de Aceptación

- Busca que el cliente/usuario se familiarice con el producto, generando comodidad y confianza sobre el mismo (su objetivo principal no es encontrar fallas)
- Es llevado a cabo por el cliente/usuario
- Determina si el sistema se ajusta a las necesidades del cliente
- En esta etapa no deberían encontrarse defectos (ya que se realizaron todas las pruebas anteriores con éxitos)
- Incluye las pruebas en laboratorio o en ambiente real de trabajo

Proceso de Testing/Prueba

- El Testing es un proceso definido (que se lleva a cabo durante todo el ciclo de vida del producto), por lo que esta compuesto de etapas detalladas



Planificación

- Al igual que cualquier proceso, se debe planificar como se llevara a cabo
- Define el plan de Testing/prueba
- Determina como se incluirá el Testing en el plan del proyecto, y como sera el Test Plan (que recursos se usará, que riesgos se tendrá, cual será el criterio de aceptación, que entornos se emularan, quien realizara cada Testing, cuando, etc)
- Se verifica que se entiendan las pruebas y sus objetivos
- Se trabaja respetando esta planificación durante todo el proyecto

Diseño

- Revisando las bases de la planificación del Testing, se identifican, diseñan y priorizan los CP que se llevaran a cabo (se define que entorno se usara, como se llevaran a cabo, por quien, cuando, etc)
- Se crean los CP
- Analiza si los requerimientos son testeables o no

Ejecución

- Se ejecutan los CP según como se definieron, controlando que se cumplan o no (se asignan valores de pruebas, se crea el entorno)
- Se trata de automatizar lo más que se pueda respecto de estas ejecuciones (ahorrando tiempo/costo)
- Cuando se ejecutan los CP, se registran y se comparan los resultados
- Se obtiene un reporte de defectos (define todos los defectos encontrados, con condiciones, entornos, etc)

Seguimiento

- Se hace un seguimiento de la corrección de los defectos encontrados
- Se hace hasta que se cierren todos los CP, es hayan solucionado todos

Caso de Prueba

- Son el conjunto de instrucciones que definen que se debe hacer, bajo un conjunto de condiciones y variables de entorno, para obtener un resultado esperado
- Deben estar expresados de manera que cualquier persona pueda llevarlos a cabo
- Mientras mejor estén definidos, mayor cantidad de defectos se abarcaran
- Permiten detectar si el producto funciona como se esperaba o no (su objetivo es encontrar errores)
- Son la unidad mas pequeña de prueba posible (la más granular)
- Se trata de que, con la menor cantidad posible de CP (es decir con la menor cantidad de tiempo/esfuerzo/costo), se abarque la mayor cantidad de funcionalidad posible en el código
- Existen distintas técnicas para declarar estos CP, cada una sera enfocada a un aspecto particular, teniendo en cuenta un entorno y tipo de producto especifico (conviene combinar varias técnicas)
- Esta compuesto por:
  - Un objetivo (lo que se desea controlar)
  - Condiciones de prueba (Datos de entrada y de entorno que deben estar presente para llevarse a cabo la prueba)
  - Un resultado esperado

Derivación de CP

- Se pueden ejecutar CP en casi todos los aspectos del producto (documentación, información, requerimiento, código, especificaciones sobre el código)

Ciclos de Prueba/Test

- Abarca la ejecución de todos los CP establecidos en una versión determinada del producto
- Se debe hacer al menos un ciclo de test por etapa (se pueden dejar de hacer pruebas en el segundo ciclo, o uno posterior)

Hay dos estrategias para aplicar las pruebas → Con y sin regresión

Regresión

- Los ciclos de test se pueden llevar a cabo con o sin regresión
- Al trabajar con regresión, por cada defecto corregido, se deben volver a hacer todos los CP (inclusive aquellos que te habían dado bien en el pasado, porque pueden haber cambiado) , para evitar que se propaguen errores sin que nos demos cuenta

Tipos de pruebas:

Smoke Test

- Se basa en probar, de manera rápida y arbitraria, muchas funcionalidades del sistema
- Sirve para asegurar, si se encuentra un error al realizar la prueba, que se tienen defectos por corregir
- Al probar todo rápido, podes asegurar: si encontré defectos poniendo valores así nomas, es porque hay defectos por corregir
- Si no aparecen defectos poniendo cualquier valor, menos defectos tendré aplicando los valores correctos
- Nos ahorra empezar a testear formalmente si encuentra un error, porque todavía hay algo que corregir

Testing Funcional

- Se fija que el SW se comporte de la misma manera en que lo especifica la documentación
- Controla funcionalidades y características definidas en la documentación
- Se basa en los requerimientos y el proceso de negocio
- Se basa en QUE hace el sistema

Testing No Funcional

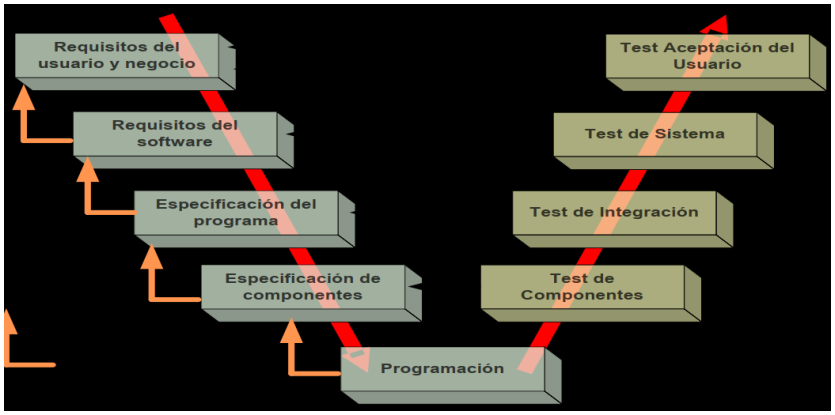
- Relacionado a la forma en la que el sistema resuelve sus funcionalidades
- Esta relacionado con los requerimientos no funcionales
- Incluye varios tipos de pruebas, como las de Carga, Stress, Mantenimiento, Portabilidad, etc
- Se basa en COMO trabaja el sistema

  De interfaz de Usuario
  - Se realizan pruebas relacionadas a la GUI, y las cosas que se puede hacer junto con los valores que se pueden ingresar

  Pruebas de Performance
  - Controlan la performance que tiene el sistema en varios escenarios teniendo en cuenta distintos aspectos (concurrencia, tiempo de respuesta, volúmenes de datos, etc)

  Pruebas de Configuración
  - Es un tipo de prueba más técnica (compatibilidad con otros componentes de SW, conectividad, periféricos del sistema, etc)

<u>Verificación</u> → Controla que el sistema se este construyendo correctamente, como se esta implementando el sistema
<u>Validación</u> → Controla que se este construyendo el sistema correcto, el que se ajusta a las necesidades del cliente, el que el cliente quiere y espera (Compara con los requerimientos)



Test Driven Development / Desarrollo basado en las pruebas

- TDD se basa en un proceso iterativo que permite desarrollar el programa partiendo de las pruebas del SW
- Se diseñan primero los pruebas del SW (principalmente pruebas unitarias) que se buscara validar, y se desarrolla el SW después en función a ellos
- Voy ejecutando las pruebas, y voy desarrollando hasta que las pase
- Hace uso de la refactorización → Al final, cuando ya se desarrollo de manera que el sistema pasa las pruebas, se ordena/refina el código y sus componentes (estéticamente, íntegramente)
- Una ventaja es que permite capturar los requerimientos y definir las pruebas al principio, al mismo tiempo (y se asegura de que se cumplan los requisitos, ya que se pasan la pruebas basadas en ellos)
- Se suele aplicar en proyectos ágiles, y solo en equipos con mucha experiencia

Manifiesto para el Testing en Agile

5 valores:

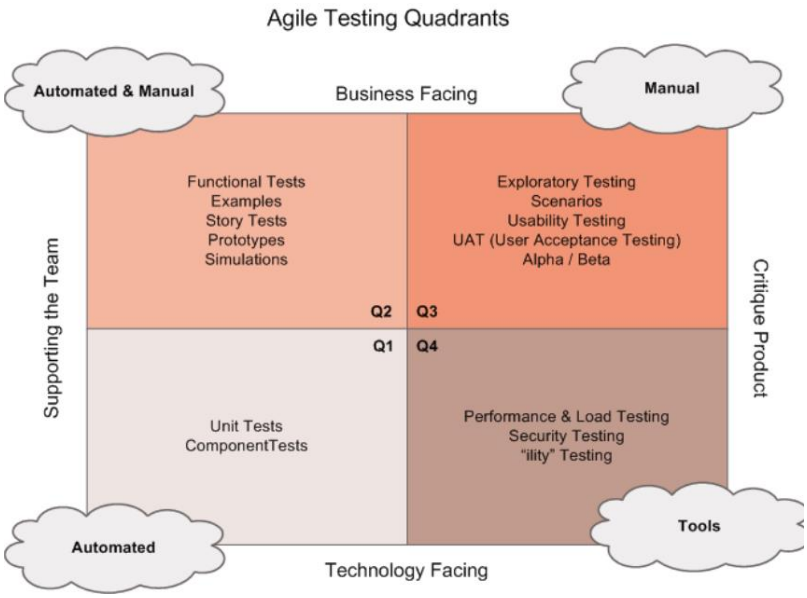
- El Testing se debe realizar a lo largo del desarrollo, y no al final
- Se debe priorizar prevenir fallas, en vez de encontrarlas
- Se debe entender el Testing, mas que solo comprobar que el sistema funcione
- Se debe trabajar para construir el mejor sistema posible, y no para destruir/romper el trabajo de otros
- El equipo debe compartir la responsabilidad de la calidad, no solo el tester

9 principios:

- Testing se mueve hacia adelante en el proyecto
- Testing no es una fase
- Todos hacen Testing
- Reducir la latencia del feedback
- Las pruebas representan expectativas
- Mantener el código limpio, corregir los defectos rápido
- Reducir la sobrecarga de documentación de las pruebas
- Las pruebas son parte del “done”
- De “Probar al final” a “Conducido por Pruebas”

6 prácticas que cumplen con el manifiesto:

- Automatizar las pruebas unitarios y de integración
- Automatizar las pruebas de regresión de sistema
- Hacer pruebas exploratorias (probar para ver que pasa, fuera de lo conocido)
- Aplicar TDD
- ATDD (Desarrollo conducido por Pruebas de aceptación)
- Controlar la versión de las pruebas con el código



Proceso de Verificación y Validación

- Es un proceso que esta presente en todo el ciclo de vida del producto
- Empieza cuando se revisan los requerimientos
- Analizan productos de trabajo → La salida de cualquier actividad dentro del ciclo de vida de desarrollo (puede ser código, una documentación, un manual, un registro, un informe, product Backlog, US, etc)
- Las fallas pueden darse por muchos motivos, uno de ellos siendo los problemas de comunicación en el equipo (por eso se busca trabajar con una cantidad reducida de integrantes)
- Otro motivo es la limitación de la memoria → Mas grande y complejo es el sistema, mas fallas tendrá
- La inspección se hace teniendo en cuenta los componentes que pueden generar fallas mas graves/caras primeros (por ejemplo, es mas importante inspeccionar la ERS que el manual de usuario)
- Al identificar errores busco evitar el re-trabajo evitable

Las fallas pueden ser:

- Mayores → Se tiene que corregir para seguir desarrollando, son muy graves y no permiten presentar el producto. Una vez corregidas se debe volver a inspeccionar
- Menores → Puede que el producto se acepto o no con estas fallas. Si se acumulan terminas teniendo un problema grave. Una vez corregidas no se vuelve a inspeccionar
- Notas cosméticas → Muy menores, no atrasan al desarrollo

La VyV tiene 6 principios

- La prevención es mejor que la cura
- Evitar es más efectivo que eliminar
- La retroalimentación enseña efectivamente
- Priorizar lo rentable
- Olvidarse de la perfección, no se puede conseguir
- Enseñar a pescar, en lugar de dar el pescado

Existen dos formas de aplicar la VyV → Aseguramiento de calidad de PRODUCTO

- Una es con Pruebas de Software/Testing
- La otra es con Revisiones Técnicas

Revisiones Técnicas

- Es un proceso estático (se hace sin ejecutar nada, mientras se va desarrollando el sistema, incluso antes de que se pueda ejecutar)que busca encontrar defectos y corregirlos lo antes posible
- Se puede inspeccionar cualquier producto de trabajo del SW (código o no)
- Se puede aplicar en cualquier momento del desarrollo

- No requieren una ejecución del trabajo
- Para cada producto de trabajo, se tendrá roles mas acertados/eficientes para realizar la inspección (el tester sirve para casi todos)
- Puede trabajar con métricas

Métricas Sugeridas	Fórmula
Densidad de defectos	Total de defectos encontrados / tamaño actual
Total de defectos encontrados	Defectos.Mayor + Defectos.Menor
Esfuerzo de la inspección	Esfuerzo.Planning + Esfuerzo.Preparación + Esfuerzo.reunion + Esfuerzo.Retrabajo
Esfuerzo por defecto	Esfuerzo.Inspeccion / Total de def encontrados
Porcentaje de reinspecciones	Cantidad Reinspecciones / Cantidad Inspecciones
Defectos Corregidos sobre Total de Defectos.	Esfuerzo.Inspeccion / tamaño actual

Ventajas

- Evita el re-trabajo
- Descubre errores
- Se puede inspeccionar aunque el producto de trabajo este incompleto
- Mejora la calidad

Desventajas

- Son complicadas de realizar (sobretodo al principio)
- Pueden parecer burocráticas, o una perdida de tiempo (se ven sus frutos en el desarrollo mas tardío)
- Ralentizan el desarrollo

Hay muchos tipos de RT, nosotros nos enfocaremos en dos:

Walkthrough

- Son un tipo de RT no formal (no tiene un proceso definido, con pasos detallados), parecida a una revisión de pares
- Su objetivo es capacitar a los desarrolladores y generar la menor sobrecarga (molestar lo menos posible)
- Se aplica en entornos rápidos
- No tiene mediciones ni registros, se hace de manera rápida
- No requiere capacitación o preparación para hacerse
- Es barata

Inspecciones

- Es un proceso de revisión formal, con pasos, roles, resultados y etapas detalladas
- Es una actividad de garantía de la calidad del SW
- Su objetivo es descubrir errores y verificar que el SW alcance los requisitos planteados, y ademas, generar un conocimiento compartido del producto de trabajo
- Garantiza ciertos estándares que cumple el SW
- Tiene un resultado binario (pasa o no pasa)
- Suele ser más caro y más lento
- Detecta y remueve todos los defectos de manera eficiente y efectiva
- Hace uso de mediciones y verificaciones
- Se lleva a cabo mediante reuniones, que deben estar bien planificadas si se espera tener éxito
- Forma más barata y efectiva de encontrar fallas

- Una forma de proveer métricas al proyecto
- Una buena forma de proveer conocimiento cruzado
- Una buena forma de promover el trabajo en grupo
- Un método probado para mejorar la calidad del producto
- No se usa para encontrar soluciones, se usa para encontrar errores
- Debe hacerse en un equipo pequeño (de 3 a 5 personas)

Roles en la inspección (una persona puede tener 1 o mas roles)

- Autor
  - Es el creador del producto, o quien se encarga de su mantenimiento
  - Es el que inicia el proceso de inspección, solicitando una inspección
  - Reporta datos al moderador
- Moderador
  - Planifica/lidera la inspección
  - Define los roles junto con el autor
  - Se encarga de entregar el producto a los inspectores 48hs antes de la reunión
  - Coordina la reunión
  - Realiza el seguimiento de los defectos encontrados
- Lector
  - Lee el producto que se esta inspeccionando a todos los integrantes de la reunión
- Anotador
  - Registra lo que se encontró en la inspección
- Inspector
  - Examina el producto antes de la reunión, buscando defectos

Proceso de inspección

- Planificación
  - El moderador, a pedido del autor, planea la inspección (lugar, tiempo, roles, etc)
  - La inspección debe durar como máximo 2 horas
- Visión General
  - El autor describe una visión general y antecedentes del producto de trabajo a inspeccionar a los inspectores
  - Es opcional
- Preparación
  - Es la preparación de cada rol para la reunión
  - Cada rol adquiere una copia del producto de trabajo, que deberán leer y analizar
- Reunión
  - En la reunión, el lector leerá cada producto de trabajo (se deben inspeccionar cosas de tamaño pequeño)
  - Cada inspector comparte los defectos que encontró
  - No solo se encuentran errores, si no que se aprende entre todos
  - Se terminara definiendo si se acepta el producto (con modificaciones o sin modificaciones) de manera permanente o provisoria (en caso de que los errores sean pequeños y se pueda aceptar de manera momentánea, aunque se deben corregir los errores)
  - También se realiza un informe detallando que se reviso, por quien, que se descubrió y que se concluyo
- Corrección
  - Es la corrección del producto de trabajo, la hace el autor
- Seguimiento
  - Se realiza un seguimiento de la evolución del SW, se revisan los cambios que se aplicaron