



# Ingenería y Calidad de Software

## SCM - Herramientas de SCM

Implementar una estructura de repo propuesta para el Contenido de la Catedra

- Estructuras de directorios/carpetas: jerarquia de directorios
- Identificar los ICs
  - Nombre del Item
  - Regla de nombrado: unica para el item  
opcional: numero, fecha, cadena de texto
  - extension:
- Definicion de los posibles ICs q se pueden identificar/generar a lo largo de la cursada
- Definir, a mivel de equipo, un criterio para la creacion de la Linea Base:  
escribir en un parrafo diciendo que la linea base se crea en "tal momento" o cuando " se "cumplan ciertas condiciones"
- Cantidad de Items considerables q cumplan con las reglas de nombrados

## Linea base

nombre unívoco, no se establece con fecha y hora, se establece cuando se cumplen ciertas condiciones

"Se establecerá la Linea de Base del proyecto cuando se realice la devolución (entrega post-corrección) de los primeros 3 Trabajos Prácticos - ya sea evaluable o no evaluable- denido que estos tienen la aprobación (nota) del cliente (profesores)"

## User Stories

Técnica para reconocer requerimientos agiles de **usuario** (alineado con algún requerimiento de negocio → el req de usr es medio para cumplir un req de neg). Agil pq esta técnica cumple principios del Manifiesto Agil

Rol a → Producto Owner: persona no técnica a cargo de la US (quien sabe sobre, no quien la escribe). Debe hacer foco en las Necesidad de Usr que van a satisfacer las Necesidad de Neg, ya que se supone que conoce el negocio y será el eslabón entre el Usr/Neg y el Equipo de desarrollo

Técnica para reconocer requerimientos agiles. Busca traducir una necesidad del usuario (funcionalidad de sistema) de negocio. Interpretarla a nivel técnico pero describirla con el lenguaje del usuario.

"el enfoque de requerimientos agiles debe estar adherido al Man Ag - cumplir con sus valores y principios-

principio 1: Nuestra mayor prioridad es satisfacer al cliente.

principio 6: El método más eficiente de comunicar información es conversaciones cara a cara.

## Las 3 C de las user stories

- Conversación: (lo mas importante) reunión con el usuario para de allí extraer la verdadera necesidad que posteriormente sera una funcionalidad
- Card (o "Tarjeta")
  - Frase verbal: frase corta que identifica que permite saber de que se está hablando (

- Descripcion: Quien(rol a cardo), Que(valor de negocio), Para que: Como [rol] quiero [que] para [valor de negocio]  
nota: valor de negocio es uno por user stories (no puede ser compuesto (2), habra q separar)
- Criterios de aceptacion: se describe una parte de la conversación ya que ahí se mencionan las condiciones por las cuales el cliente aceptara la US
- Pruebas de usuario: luego de los casos de prueba, pequeña experiencia del usuario con el sw donde este acepta o no, esa parte de funcionalidad implementada descripta en la US
- Confirmación:
  - Pruebas de aceptacion: conjunto de escenarios posibles a los que se somete el sw implementado, si el P.O. acepta, posteriormente pasara a producción

### Caso de prueba

Examinar la calidad del sw, detallando paso por paso ,con ejemplos, de lo que haria el cliente en una situación

## Criterio de Ready

Conjunto de condiciones que se deben cumplir para que la US esté lista para Implementar, es decir, para que el equipo tecnico la desarrolle

INVEST: un criterio aplicable

- Independiente: en un momento de desarrollo dado, tengo independencia entreUserStories
- Negociable: que la US posee cierta flexibilidad a cambios propuestos por parte del equipo de desarrollo con el Product Owner, este es el que finalmente aprueba o no.
- Valuable: "o que da valor": que las US ademas de su funcionalidad, tiene que darle valor al negocio
- Estimable: la posibilidad de definir tamaño (de lo que pide la US) y duracion (que llevaria desarrollarla). Si no se puede, candidata a dividir en Us mas pequeñas

- Small: completar la totalidad de la US en una iteración de trabajo definida (2 o 4 semanas)
- Testable: que lo desarrollado en base a la US sea fácil de aplicar test y que permita la calidad del software

## **Empaquetado u organización de US**

- Temas
  - Epicas
    - User Story 1: dos US pueden ser parte de una epica
    - User Story 2:

## **Estimaciones**

### **Según el enfoque**

- Tradicional:
  - Tamaño: (líneas de código → módulos → paquetes):
  - Esfuerzo: Horas personas lineales horas efectivas de trabajo de
  - Calendario
  - Costo
- Ágil
  - Tamaño: Cantidad de características del Producto → US

### **Cómo se realiza**

El equipo de desarrollo acuerda una US canónica (patrón). Gralmente la US más pequeña / "fácil". Story Point: peso que se le asigna a la US por compararla con la US Canónica

Escalas:

- Escala decimal: del 1 al 10.

- Exponencial:  $2^n$
- Fibonacci: 0, 1, 1, 2, 3, 5, 8

Asignación de peso segun:

- Complejidad: inherent
- Esfuerzo: despendiendo de la capacidad individual y grupal del equipo del desarrollo
- Incertidumbre: que tan informado y cuanto experiencia con respecto al conocimiento del dominio,

## PRÁCTICO 1 - REQUERIMIENTOS ÁGILES – User Stories

### Conversación

Identificación de los

Identificación Técnica	Conversión a lenguaje del Usr
Consultar gastos	Ver planilla de gastos
Registrar gasto	Guardar gasto
Reg un tipo de gasto	Reg un tipo de gasto
Registrar usuario	Registrar usuario
Iniciar sesión	Iniciar sesión
Cerrar sesion	Cerrar sesion
Filtrar gastos	Filtrar gastos
Modificar gasto? abmc?	Modificar gasto
Eliminar gasto? abmc?	Eliminar gasto
Registrar responsable de gasto?	Agregar responsable de gasto

## Gestión de Productos

## Tipos de Productos de Sw

- **Software enlatado (o estándar, empaquetado):**
  - Se desarrolla para un mercado masivo.
  - Ejemplo: Microsoft Office, navegadores web, sistemas operativos.
- **Software a medida (o dedicado, personalizado):**
  - Se crea específicamente para un cliente o empresa.
  - Ejemplo: un sistema de gestión de stock hecho para una pyme.

## Explicando MVPs, MVFs, MMFs con el Dinosaurio Lean/Agile

- Uvp: Disparador/ Hipotesis de necesidad y funcionalid/es que satisfaceran esas necesidades en el mercado
- Mvp: producto minimo viable → contiene un conjunto de funcionalidades
  - Mvf: los mismo que mvp pero de una sola funcionalidad
  - Mvp[vesion]: versiones de mvp q pueden ser incorporadas al mvp inicial por variaciones o feedback de mercado
- MMP: salgo al mercado con el producto que posee ese conjunto de funcionalidades (Mvp comecializable)
- MMF: salgo a mercado con una sola funcionalidad (feature) (Mvf comercializable)
- MMR: realese: version nueva que se pone en produccionx

## PRÁCTICO 2 - REQUER. ÁGILES – User Stories y Estimaciones

### Conversación

Identificación de los

Nº	Indentificación Técnica	Conversión a lenguaje del Usr
1	Consultar mapa interactivo	Ver mapa interactivo

2	Registrar usuario	Registrar usuario
3	Iniciar sesion	Iniciar sesion
4	Cerrar sesion	Cerrar sesion
5	Consultar horarios de alimentacoin	Ver horarios de alimentacoin
6	Registrar inscripcion actividad	Inscribirse a actividad
7	Registrar compra de entrada	Comprar entrada con efectivo
8	Filtrar actividades	Ver actividades

Nota!: siempre verificar el criterio INVEST

- Comprar entrada con efectivo y Comprar entrada con tarjeta: son criterios de aceptacion
- Los ABMC son candidatas a analizar para ser tomadas como Canonicas
- Analizar "pocresos automaticos" que son desencadenados por acciones del usr

## Planificación de un proyecto de Software

Recordando las 4 P, pero haciendo enfasis en el proyecto:

P de Proyecto:

Caracteristicas

- Resultado unico
- Fecha de inicio y fin
- Elaboracion
- Tareas interrelacionadas

# ¿QUÉ IMPLICA LA PLANIFICACIÓN DE PROYECTOS DE SOFTWARE?

- Definición del Alcance del Proyecto
- Definición de Proceso y Ciclo de Vida
- Estimación
- Gestión de Riesgos
- Asignación de Recursos
- Programación de Proyectos
- Definición de Controles
- Definición de Métricas

## Objetivo del proyecto

- Claro: sin ambigüedades y que todos los involucrados en el proyecto entiendan lo mismo por "objetivo"
- Alcanzable: que con los recursos que se poseen es posible realizar lo que se plantea

## Alcance del proyecto

"es el trabajo, solo el trabajo que es necesario para cumplir con el objetivo"

"no son las cosas que hace el producto (reqs), es el listado de cosas que se deben realizar (estilo paso a paso) para cumplir uno de

## Definir el proceso y el proceso de vida del proyecto

Dependiendo del ciclo del vida de proceso (dependiendo del tipo que sea: Empírico o Definido) que se elija, el proyecto adapta ese proceso, definido teóricamente, estableciendo cuánto de qué actividad se realizará, y en el caso que sea necesario, la omisión de alguna.

Las actividades, que producto de la adaptación no se llevan a cabo, es bajo el concepto de "esta actividad No Aplica", es decir, bajo alguna justificación; no por libre albedrio

## Definir el equipo de trabajo del proyecto

El proceso define los roles y responsabilidades, en el proyecto se instancia. "el proceso se define en termino de roles"

ej: el product owner: es el responsable de...

En el proyecto se define qué **persona** tendrá qué rol. En el proyecto se le asigna a una persona el/los roles

Una persona puede tener más de un rol, por lo tanto, más de una responsabilidad

ej: el product owner será Juan Jose W

## Estimación (enfoque TRADICIONAL)

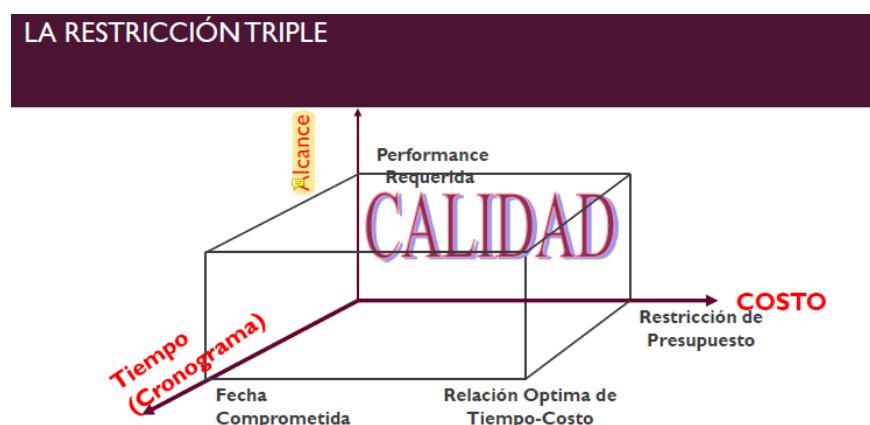
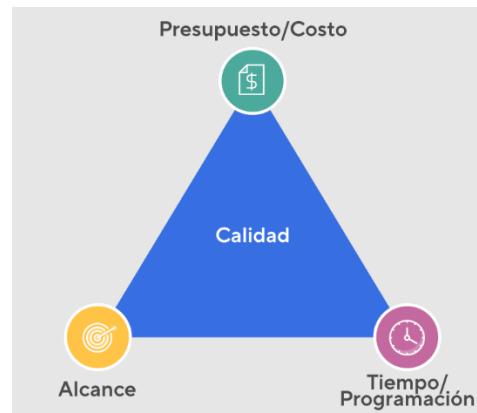
Predicción. Un supuesto que siempre tiene una probabilidad asociada

- Tamaño del Producto → asociado a Obj, Alcances
  - Unidad de medida de estimación: Casos de Uso x Complejidad
    - ej.: n cantidad de CU simples; m cantidad de CU de mediana complejidad, k cantidad...
- Esfuerzo del Proyecto
  - Unidad de medida de estimación: Horas personas lineales
    - Cantidad de horas netas/efectivas que se dedican a la construcción/desarrollo de los CU
    - ej.: n cantidad de horas para un Junior + m cantidad de horas para un Senior
- Tiempo y calendario del Proyecto
  - Análisis de actividades secuenciales y en paralelo
  - Contemplar las horas no productivas en la jornada laboral
- Costo
  - Ultimo ítem ya que todo lo anterior condiciona la monetización
  - costo base:
    - cantidad de horas definidas en el esfuerzo
    - precio hora del personal: puede ser horas ingeniero planas o por rol
  - ganancia: fuera del proyecto (objetivos comerciales)

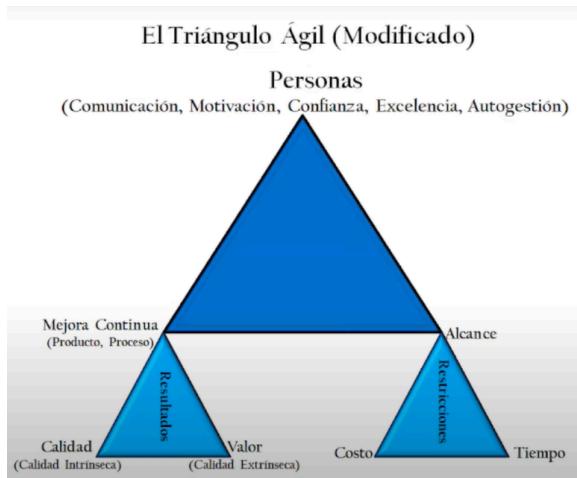
- Recusos Criticos

licencias de sw, hardware especifico, otros

### Triangulo de Hierro (enfoque Tradicional)



### Modificación hacia un Enfoque Agil



## Proceso Unificado de Desarrollo

con ciclo de vida iterativo

- En el enfoque Tradicional

De alcance fijo: se trabaja el tiempo que sea necesario hasta que se obtenga esa funcionalidad de sistema requerida

- En el enfoque Agil:

De duracion fija: se desarrolla todo lo relacionado a la funcionalidad en un tiempo fijo

## Calendarizacion

Diferida

anticipada

## Riesgos

## Seguimiento

# PRÁCTICO 3 - REQUER. ÁGILES – User Stories y Estimaciones

## Listado de Requerimientos

Nº	Conversión a lenguaje del Usr	Identificación Técnica
1	Comprar ropa	
2	Vender Ropa	
3	Comprar ropa	
4	Filtrar ropa	
5		
6		
7		
8		

## Scrum

Framework para gestión agil de proyectos para poder generar de manera daptatatipta solciones a complejas

No dice como contruir software

Adhiere a todas las practicas que dicta el Manifiesto Agil (4Valores; 12 Principios)

Como sigue el Manifiesto Agil es aplicado a los Procesos Empirico, por consecuencia, tambien es guiados sus 3 pilares:

- Transparencia (hacer que la información sea accesible y comprensible),
- Inspección (evaluar periódicamente el progreso y los artefactos) y
- Adaptación (ajustar el plan o el producto basándose en los hallazgos de la inspección).

Categorias de responsabilidad (anteriormente llamda roles)

remposables de crear los productos de trabajo en el proyecto

- Scrum Master
- Product Owner
- Desarrolladores t

Principio N° 4 del MA: Tecnico y no tecnicos trabajando en conjunto

1. P.O.: No tecnico → eslabon entre los usuarios/clientes y el equipo de desarrollo. Si no existe, se inventa uno, este es llamado Proxie (asume el rol del P.O.). Un P.O. por producto
2. S.M.: Tecnico: Lider de servicio: eslabon entre el P.O. y el equipo de desarrollo
  - a. que las reglas del **Scrum** se cumplan
  - b. organizar reuniones
  - c. gestionar y disponibilizar los recursos necesarios para
  - d. encargado del Cómo se

TimeBox/Eventos/Ceremonias

Planning

Daily

Sprint: es una iteracion de duracion fija. Al final, genera un incremento del producto. Contenedor de todas las actividad. Duracion maxima de **un mes**

TimeBox:

Product Backlog

La cantidad que caracteristicas minimas necesarias que permita realizar la primera iteracion completa, osea, el primer Sprint.

PBI (product backlog items): de forma generica ya que Scrum recomienda que estos items sean User Stories. Sin embargo, puede usarse otras cosas dif a las US

Elementos que puede contener el PB

Lista priorizada de caractiriticas que debe tener.

- US (User Stories): requerimiento de usuario que sigue un req de negocio
  - Epica: "una US grande"
  - Tema:

- Deuda Técnica: un proyecto con deuda técnica funciona. Ej.: hardcodeo, nivel malo de acoplamiento y cohesión
- Defectos

### Definition of Done

En el contexto ideal de Scrum con equipos Autogestionados, Motivados, etc. donde una persona podría realizar una US completa

To	Doing	Done
US1 (sp:3)		
US1 (sp:5)		

Si la US se desagrega en Tareas, al completarlas se logra que esa US cumpla con DoD

Estimación:

Complejidad con respecto a pagos y conexión

Nivel de incertidumbre:

A nivel negocio

A nivel técnico

Esfuerzo:

Complejidad:

Justificación:

## Canonica

Se puede estimar la canonica con Sp de 1, esta es:

- US Patrón: US parámetro para la estimación de las otras US en base a la comparación a la US a estimar con esta ya estimada de Sp1
- Al asignar una Sp a la US: se tiene q justificar el Por qué, para uno (capacidad técnica, experiencia previa), la US canonica tiene 1 Sp o la US estimada tiene la Sp q obtuvo

- Se puede asignar

## Producto

alcance del producto

que se le va a entregar al cliente, eso hay q definirlo

en el enfq tradicional → ESR (def objetivos, def alcances y **listaba requerimientos**)

en el enfq agils → tomaba un puñado de reques y los trabaja en un proyecto  
donde no hay necesario

## Listado de caracteristicas

## Proyecto

"el proyecto de desarollar el producto de autogestion"

cumple con un objetivo que depende...

esta relacionado con el objetivo del producto (por eso solo menciona)  
en la culmi

alcance

todo el trabajo y solo el trabajo ncesario para poder cumplir con el objetivo  
del proyecto

es decir, el conjunto de actividades/tareas

- gestionar de proy
- reques
- diseño
- prueba

el alcance del producto condiciona el alcance del proyecto

mvp: una idea de producto de la cual se plantea una hipotesis (si el mercado (gente/ususrios) pretede usarlo)

prototipo, video, ppt

mmp: producto minimo comercializable: q se puede lanzar al mercado (pago o gratis)

## Testing

### Tradicional

Recordando las etapas del Proceso de Desarrollo de Sw (PUD)

- Requerimientos
- Analisis
- Diseño
- Implementacion
- PRUEBA/TESTING
- Despliegue

Dependiendo del nivel de especificación o detalle (decisiones) que se haya hecho en una Etapa/Ambiente del Desarrollo de Sw, la siguiente tendra mas o menos responsabilidad de detallar (tomar esas decisiones o las faltantes) lo necesario para poder cumplir

### Proceso de Testing

Etapas	Artefacto	Se puede realizar a partir de	Producto de la etapa de Pud
Planificación	Plan de pruebas	Requerimientos (ers)	Requerimientos
Diseño	Casos de Prueba	Requerimientos (ers)	Requerimientos
Ejecución	Reporte de defectos	Release (versión de producto)	Implementación
Cierre			

A partir de que los requerimientos están mediamente definidos, se puede adelantar con el Plan de pruebas y Diseño de Casos de prueba

Luego, a partir de una o la primera versión de producto (salida de la Implementación del PUD), se puede llevar a cabo la Ejecución de los Casos de prueba para poder obtener un Reporte de defectos

## Tipos de testing

- AdHoc
 

no metodológico, no sistemático, no tiene proceso, no se basa en casos de prueba.

Testing al estilo Fuerza bruta o Explorativo
- Metodológico
 

todo lo contrario del adhoc: sigue

## Niveles de Pruebas

Vinculación de nivel de testing con nivel de proceso de desarrollo tradicional

## Caso de prueba

Etapa del Proc Desarr Trad (PUD)	Niveles de Testing/Prueba Teórico	Qué se prueba	Rol	Que se falla se encuentra
Requerimientos				
Análisis				
Diseño				

Implementación	Pruebas Unitarias	Un componente individual	Desarrollador	Error
Prueba	Pruebas de Integracion	Una funcionalidad: un conjuntos de componentes individuales que se relacionan para cumplir con esa funcionalidad	Rol de Tester en el equipo de desarrollo	Defecto
Prueba	Pruebas de Sistema/Version	Se prueba la aplicación funcionando como un todo	Rol de Tester en el equipo de desarrollo	Defecto
Despliegue	Pruebas de Aceptacion de Usuario (UAT)	Se prueba la aplicación funcionando como un todo	Usuario del sistema: Product Owner	Defecto

se prueban/llevan a cabon las pruebas de usuario (posibilidades/caminos posibles de la US)

- Identificacion: nombre de la funcion de prueba: generalmente un nombre largo muy preciso/especifico

Se debe conocer el producto, es decir, debo trabajar a con los Requerimientos para Diseñarlos, y para eso, los reqs deben estar

Descripcion de pasos/acciones de un ESCENARIO: que se deben hacer indicando que variables se seleccionaran y con cual valor deberan completar (condiciones de setteo)

Imp: sino se cumple con lo que dicta el CdP este (falla)

- qué acciones se ejecutaran
- que variable y con cual valor se completa o ingresa
- con lo anterior, que resultado se espera del caso de prueba

- precondiciones o contexto de sistema: que cosas deben estar ya disponibles en el sistemas para que caso de prueba pase

Ejemplo: Caso de Prueba correspondiente al Caso de Uso: Iniciar sesion

paso1: Seleccionar la opcion Iniciar sesion

paso2: ingrese el usuario genavillegas e ingrese la contraseña 1234

Cuando se deja de probar?

existen diferentes estrategias

- Defectos: cero defectos bloqueantes y graves y una cantidad n de los otros defectos.
- tiempo de testeo
- 

estas estrategias son plasmadas en la PLanificacion de Pruebas

el testing no asegura la calidad, pone en evidencia los defectos de sw.

la calidad de un sw es una cuestion dedicional: construyo sw con calidad: actividad transversal a toda y cada una de las actividades del desarrollo del sw, sea tradicional o agil

## Verificación y Validacion

### Error y Defecto

Error: cosas mal hechas (problemas de codigo) en el workflow de Implementacion en el PUD

Defecto: Errores sin corregir que pasaron del workflow de Implementación

	Error	Defecto
Dónde ocurren	En la etapa de Implementacion del PUD	Se descubren en las etapas Prueba y Despliegue del PUD
Cómo se detectan	Revisiones tecnicas (se escribe codigo y se lo revisa)	Ejecutando los Casos de Prueba

Cómo se solucionan	Re-escribir el código	Informar el defecto, delegar a Desarrollo donde se aplica Re-trabajo
Costo	Medio	Alto
Problemas		En pruebas de aceptación <b>fallidas</b> a veces se deben a mala definición de los requerimientos: Sistema construido correctamente pero Sist NO correcto

## Ciclo de prueba y Estrategia de pruebas

Estrategia de pruebas

- Sin retroceso
- Con retroceso

## Requerimientos - Pruebas y Tipos de pruebas

En que Ambiente del testing	Que se prueba	Tipos de Prueba asociada	En que consiste	Ejemplo
	Funcional	Reqs de operación normal	Se prueba el sistema en un escenario normal	Las pruebas de usuario que (pasa) de la US
	Funcional	Reqs negativa	Se prueba el sistema haciendo cosas que no se podrían hacer	Inscribirse a una materia que no puedo por correlativas
	Funcional	Proceso de negocio		
	Funcional	Documentación	Usuario y de Config de sistema	
	No Funcional	Performance	Se mide Tiempo de respuesta y	

			Concurrencia	
	No Funcional	Interfaces de usr	gui	
	No Funcional	Escala completa		
	No Funcional	Stress	Probar e limite soporta el sistema, que pasa cuando "se clava" y en cuanto tiempo se "levanta" de nuevo el sistema	
	No Funcional	Seguridad	Buscar vulnerabilidades en el sistema	
	No Funcional	Accesibilidad	Si estan cubiertas todas o la mayoria de las posibilidades de las formas de acceder a las funcionalidades del sist. Versatilidad del software para que pueda ser usado por usuarios con diferentes capacidades	

## Agilismo

cambia el paradigma de trabajo.

se coloca en detalle en el lugar/estapa/ambiente donde corresponda/sea indispensable:

- En la US no se detalla/plasman valores de las variables
- En los Casos de Prueba se plasma "esa decision"/ese nivel de detalle asociado a los variables, valores, contexto de sistema, etc.

incrementa la calidad del software ya que aumenta la probabilidad de que el sw sea el correcto

## TDD - Test Driven Development

tecnica de testing agil que resuelve as pruebas unitarias, lo que hace el desarrollador

"desarrollo guiado por testing"

- detección temprana y eficiente de errores
- scoup correcto: programar lo justo y necesario de lo que el cliente quiere funciones como caja negra

la salida de tdd: funcionalidades de sistema correctamente escritas

## PRÁCTICO 6 - User Stories y TDD

### Eco Harmony Park - Inscribirme a Actividades (Par)

- Uso de la tecnica de US: preguntar al cliente (profes) las dudas sobre la US
- Aplicacion de la tecnica de TDD
  - Desarrollo de la pruebas unitarias: las PU deben estan antes del codigo desarollado
  - Correcto uso del repo: primeros los commits de las las pruebas y luego el codigo
- Documento de estilo de codigo o buenas practicas: PEP8 (python)
- La funcionalidad debe andar/correr
- Interfaz de usuario

- Base de datos
- No se corrige la aplicación de patrones de código
- Simular la conexión con algún externo: Mock o Mockito

## Casos de prueba

Recordar la Severidad de los Defectos:

Baja

Media

Alta

Critica

Bloqueante : (no se suelen presentar en el testeting)

Niveles de pruebas

Pruebas de usuario

Casos de pruebas

Regresión: la ejecución de un conjunto de casos de prueba completo

Problema: introducción de errores nuevos

Se dan tamb en la pruebas de integración para verificar q no se haya roto  
nada durante esta etapa

CI/CD:

Github/Gitlab: tiene plataforma para automatizar los CI/CD. Otros: Jenkins,  
Docker

Utilizar un esquema de branching: permitiría hacer un merge al main  
automático

Es un evolución en la gestión de configuración

Consejos al pasar a producción y que falle:

- Enterarse antes que el cliente
- Analizar la posibilidad de solucionarlo, sino
- Hacer Rollback (volver a los cambios anteriores)

## Práctico N° 11 - Casos de Prueba - Detección de Bugs

Ejecutar los Casos de pruebas y Reportar bugs de los Tp de los otros compañeros??

## CI/CD

Continuos Integration / Continuos Delivery

Integración continua / Entrega continua

En esta etapa se encuentran Defectos pq ya paso la etapa de implementación (trad) de esa funcionalidad

Agil

Parte del Criterio de Hecho (DoD -Definition of Done) del equipo es unir la rama donde se esta implementando la funcionalidad al main, osea, hacer merge; o hacer directamente push sobre la rama master.

Ahi es cuando se dispara la ejecución automatizada de el Pipeline (diseñado por el equipod) de Integracion Continua: ejecucion de forma automatica de pruebas Unitarias y de Integración.

Con estas pruebas automatizadas se detectan fallas de construcción, que como ya pasaron del Ambiente de Desarrollo, y justamente son encontradas en el Ambiente de Prueba, en este caso automatizadas, estas fallas son Defectos

Es una buena practica: si se sabe cuales son las pruebas automatizadas, ejecutarlas en el ambiente de desarrollo, de forma local, antes de pushear o

mergear. En este caso, todavía se estaría en el Ambiente de Desarrollo por que las fallas en este caso serían Errores asociadas a una funcionalidad

## Filosofía Lean y Framework Kanban

el framework Kanban se sustenta sobre la filosofía Lean

### Filosofía Lean

enfoque/pensamiento más antiguo que la Filosofía Agile. Surge para mejorar la productividad. Luego, se busca la adaptación de ésta al Software

### Principios Lean (7)

- Eliminar desperdicio

Relacionado con el principio Agil → Software funcionando y el de simplicidad: Evitar invertir demasiado tiempo en hacer especificaciones, al principio, de lo que va a hacer el producto, sabiendo que posiblemente en un futuro cambie (por cambio de requerimientos)

Al momento de implementar los esos reqs/funcionalidades especificadas, el cliente ya no las quiere o necesita

Los desperdicios Lean en el Sw son:

- El trabajo hecho de forma incompleta.
- Las caract extras no solicitadas
- Pasos extras en un proceso que no generan valor o incremento.
- Movimiento: ?
- Retrabajo por Defectos
- Esperas de información o de porciones/componentes de software para poder seguir avanzando en el desarrollo
- Cambio de tareas: desarrollar sw es un trabajo humano intensivo y estar cambiando el enfoque entre una u otra tarea presenta perdidas de

tiempo productivo real (asociadas a tiempos de concentracion, foco, entendimiento y tiempo realmente productivoos)

- Amplificar el aprendizaje

Crear y mantener una cultura de **mejoramiento continuo** y solucion de problemas

El conocimiento o experiencia adquirido por los integrantes del equipo de desarrollo debe ser compartido entre los miembros. Hacer explicito un conocimiento implicito.

- Embeber la integridad conceptual

Integrar la calidad al Sw como producto:

- Integrar todas de partes del producto de Sw con coherencia y consistencia (Reqs NO funcionales).
- Integrar las personas que trabajan mediante la buena comunicacion.
- E integrarle (embeberle/incorporare) al Sw calidad, en todo momento, desde el principio, mediante buenas practicas o tecnicas

- Diferir compromisos

Decidir lo mas tarde posible pero responsablemente (pero a tiempo). Si decidimos antes no tenemos toda la informacion para hacerlo

Relacionado con el principio agil → La simplicidad es esencial: No hacer el trabajo que no va a ser utilizado. Mientras mas tarde decidimos mas conocimiento tenemos

- Dar poder al equipo

Equipos formados, auto gestionados, motivados que puedan todas decisiones, y respetarlas; que pueda estimar el trabajo de forma autónoma.

- Ver el todo

Relacionado con el Valor Agregado: Al entender desde lo bajo nivel (que estoy codificando) hasta la mas alto nivel (Reques Funcionales, NO funcionales, Valores del negocio del cliente) el producto resultando tendra un alto valor agregado

- Entregar rapido

Estabilizar ambientes de trabajo a su capacidad mas eficientes ya acortar los ciclos de desarrollo. (eso se ve en los Sprint ciclos de vida de tiempo

fijo)

Relacionado con el principio Agil → Satisfacer al cliente, Entregar sw funcional freq, Ajustar comportamiento: Entregas rapidas dan como resultados pequeños incrementos de valor, sumada la experiencia por la aprobacion o no, de las features, por parte del cliente

## Framework Kanban

para mejora continua de PROCESOS basado en la Filosofia Lean, por eso lo adaptamos para la mejora del Proceso de Desarrollo de Sw

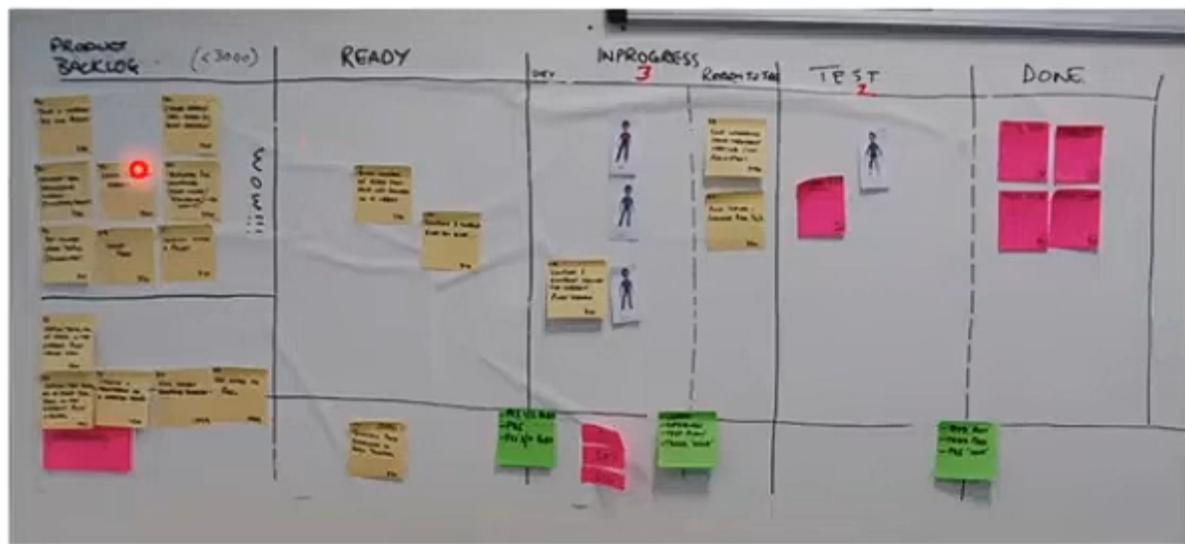
### Concepto

La Tarjeta es un concepto, simboliza un Contenedor de Información util para los involcados en la construcción de algo

Este contenedor es el medio que nos permite visualizar el flujo del proceso de trabajo que tiene el fin de construir ese algo.

### Tablero

El contenedor ultilizado en Kanban en un Tablero:



Donde se puede ver

- El proceso completo: cada columna es una etapa de este.

- Estado de actividad: en que columna del proceso se encuentra: Columna de acumulación “Done”. Columna de trabajo “Doing”
- Flujo de trabajo: como esas actividades avanzan de izq a der por haber completado su Estado. Estas con autoasignadas (pull)
- Product Backlog: Tareas (en nuestro caso: features) pendientes. No han sido comenzadas ha trabajar.
- Avatar: representa un persona del equipo a la que le pertenece el desarrollo de esa actividad
- WIP (Work In Progress): limite explicito de cuantos items (actividades o “piezas de trabajo”) que puede haber, en un determinado tiempo, en una columna en particular.

Hay que explicitar las politicas de trabajo (que le dan calidad al Sw). Permite el trabajo colaborativo y auto-gestionado corrigiendo cuellos de botella mediante la asignacion de recursos (personas)

## Cómo aplicar Kanban

Se elije un PROCESO EXISTENTE de la organización al que le quiere aplicar el Framework.

- Se analizan sus etapas y se determina si existe alguna que no genere valor
- Se lo mapea a un Tablero
- Se limita el WIP

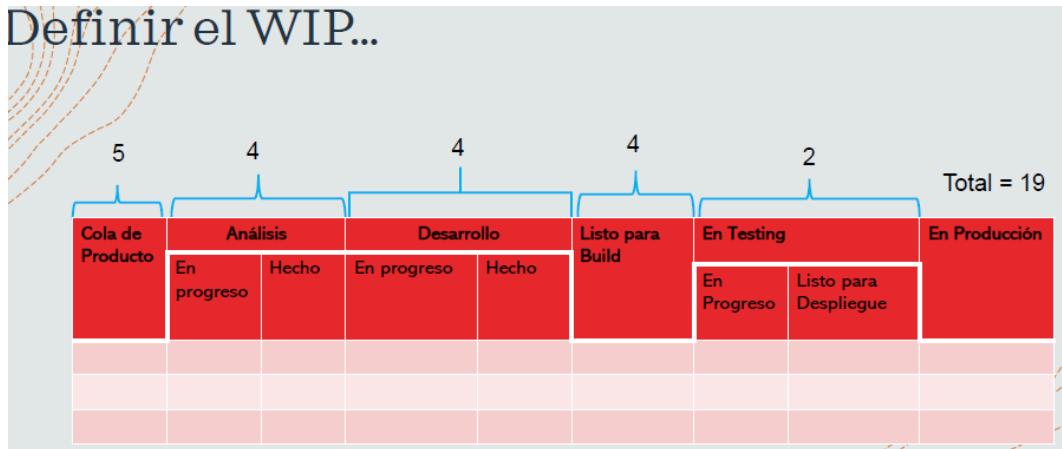
Luego de un cierto tiempo de trabajo se reflexionan posibles ajustes, como:

- Eliminación, adicion o agrupacion de pasos
- Cambios en el limite WIP
- Reasignacion de personas en las etapas debido a cuellos de botella

## Limitar el WIP

No es una tarea facil ya que consta de un estimación en base de la informacion sobre el proceso, sus actividades y las capacidades de los grupos de trabajo. Ayuda a que el trabajo fluya

## ~~Definir el WIP...~~



# **Clases de Servicio**

Diferentes trabajos que tiene diferentes políticas - definición de Done, para cada estado.



## Verde: Casos de Uso.

Cada color es un tipo

Clasificar/Difenciar los difetrentes trabajos que ingresan al backlog; por tipo y/o prioridad de desarrollo (definido en las politicas).

Otro ejemplo: colores para identificar clases de servicio y códigos de nombre para diferencias si es una US. Spike

## Ejemplo de tipos:

- Espresso (Blanco)
  - FechaFija (Rosa)
  - Estandar (Amarillo)
  - Intangible (Verde)

Cadencia

Asociada al trabajo. Con que "frecuencia" las actividades van pasando de estado en estado y finalmente terminan en el estado Done.

En Kanban el flujo es contante debido a la constante aparición de actividades en el Backlog, la autoasignación de estas y la finalización

## Métricas Kanban

- Lead Time: desde que se lista en el Backlog hasta el asume el estado Done o Complete  
"Desde que el cliente me lo pidió hasta que yo se lo entrego"
- Cycle Time: desde que se toma la actividad que estaba en el backlog y se lo empieza a trabajar, hasta el momento donde se completa
- Touch time: sumatoria de cada uno de los tiempos en donde la actividad estuvo "In Progress" en las diferentes etapas
- Eficiencia del ciclo de proceso: TouchTime/LeadTime

## Métricas de Software

Es un valor cuantitativo (número) que mide la presencia de algún aspecto o característica, en un algún contexto, de algún fenómeno o entidad.

Propósito: aumenta la visibilidad o foco sobre ese aspecto, aumentando el nivel de conocimiento.

Objetivo: Mejora la capacidad de tomar decisiones.

El beneficio de tener métricas debe ser mayor al costo de obtenerlas

Son relativas al rol y al tipo de SW q se construye

## Dominio de las métricas

- Métricas de Proceso
- Métricas de Proyecto
- Métricas de Producto

## Métrica de enfoque TRADICIONAL

## Métricas básicas

- Tamaño de (Producto)

(Qué) Cantidad de funcionalidades o características de Software: Alcances, Casos de uso, Opciones de menú (Se las puede llamar de diferentes formas pero el concepto es el mismo)

PUD: Casos de uso por complejidad (los ABMC son CU son muy simples pero las funcionalidades principales son CU complejas)

- Esfuerzo de (Proyecto)

(Cómo?) Horas personas lineales: cantidad horas bajo el supuesto q trabaja una sola persona haciendo una cosa por vez (desarrollo en serie de las actividades). No se tiene en cuenta el tiempo No productivo

- Calendario (Tiempo) (Proyecto)

(Cuándo?) Lapso de tiempo dado en semanas o meses

Cantidad de personas involucradas + Horas persona lineales + Todas las horas NO productivas (solapamiento, trabajo en paralelo, feriados, fin de semana, horas de almuerzo, etc)

- Defectos del (Producto)

Cantidad de Defectos por Severidad

## Ejemplos

De Proyecto → Vinculado con la 3ple rectriccion (

- Esfuerzo:
  - Esfuerzo por etapa
  - Esfuerzo estimado vs real
- Calendario
  - Duración estimada vs real
  - Tiempo por iteración (pq en tradicional es de alcance fijo y el tiempo puede variar)

De Producto

- Tamaño
  - Líneas de código, alcances paquetes

- Complejidad aciclonomatica (interna)
- CU (trad)
- US (agil)
- Defectos
  - Por nivel
  - Por Severidad
  - Densidad de defectos
  - Cobertura de testing (%)

#### De Proceso

surgen a partir de las metricas de proyecto. En Tradicional las metricas de proceso se pueden extrapolar a diferentes proyectos, por lo tanto, son publicas para la negocio.

Estas metricas se despersonalizan: no se las atribuye ningun proyecto ni producto

- Cantidad de defectos por severidad que tiene la organizacion
- Porcentaje de desviacion de lo estimado vs lo realizado

## Métricas en ambientes ÁGILES

User Story : la cant de puntos de historia unifica el valor