**Problemas del software**

* La versión final del producto no satisface las necesidades del cliente
* no es fácil extenderlo y/o adaptarlo. Agregar más funcionalidad en otra versión es casi una misión imposible.
* Mala documentación.
* Mala calidad.
* Más tiempos y costos que los presupuestados.

**Gráfico Procesos Vs Proyecto Vs Producto**

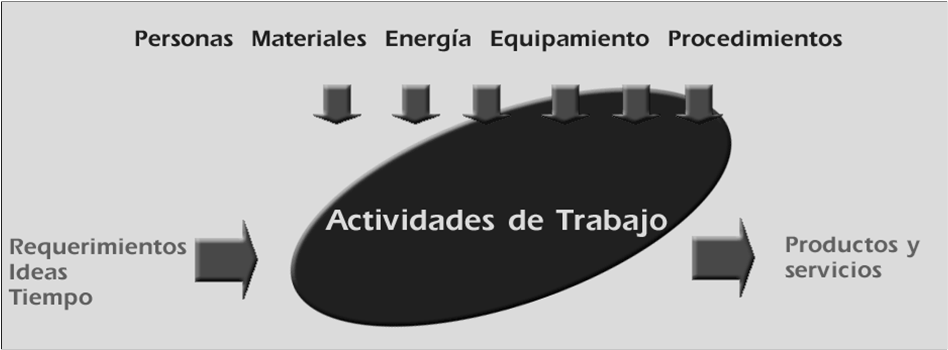


**Proceso**

*Definición de proceso*: la secuencia de pasos para un propósito dado (IEEE).

*Definición de proceso de Software*: un conjunto de actividades, métodos, prácticas y transformaciones que la gente usa para desarrollar o mantener software y sus productos asociados





* Conjunto estructurado de actividades para desarrollar un sistema de sw.
* Estas actividades varían dependiendo la organización y el tipo de sistema que deba desarrollarse.
* Debe ser explícitamente modelado si va a ser administrado.

*Proceso definido*

* Asume que podemos repetir el mismo proceso una y otra vez, indefinidamente, y obtener los mismos resultados.
* La administración y control provienen de la predictibilidad del proceso definido.

*Proceso Empírico*

* Asume procesos complicados con variables cambiantes. cuando se repite el proceso, se pueden obtener resultados diferentes.
* La administración y control es a través de inspecciones frecuentes y adaptaciones.
* son procesos que trabajan bien con procesos creativos y complejos.
* Patrón de conocimiento en procesos empíricos:



**Plan de desarrollo de software**

El plan de proyecto documenta que es lo que hacemos, cuando, como y quien lo hace.

1. Objetivo del proyecto
2. Alcance del proyecto
3. Definir el proceso/ciclo de vida
4. Estimaciones
5. Riesgos
6. Calendarización
7. Métricas
8. Seguimiento y Control

**A. Objetivo del proyecto**

Se define que se va a realizar (el producto) a lo largo del proyecto.

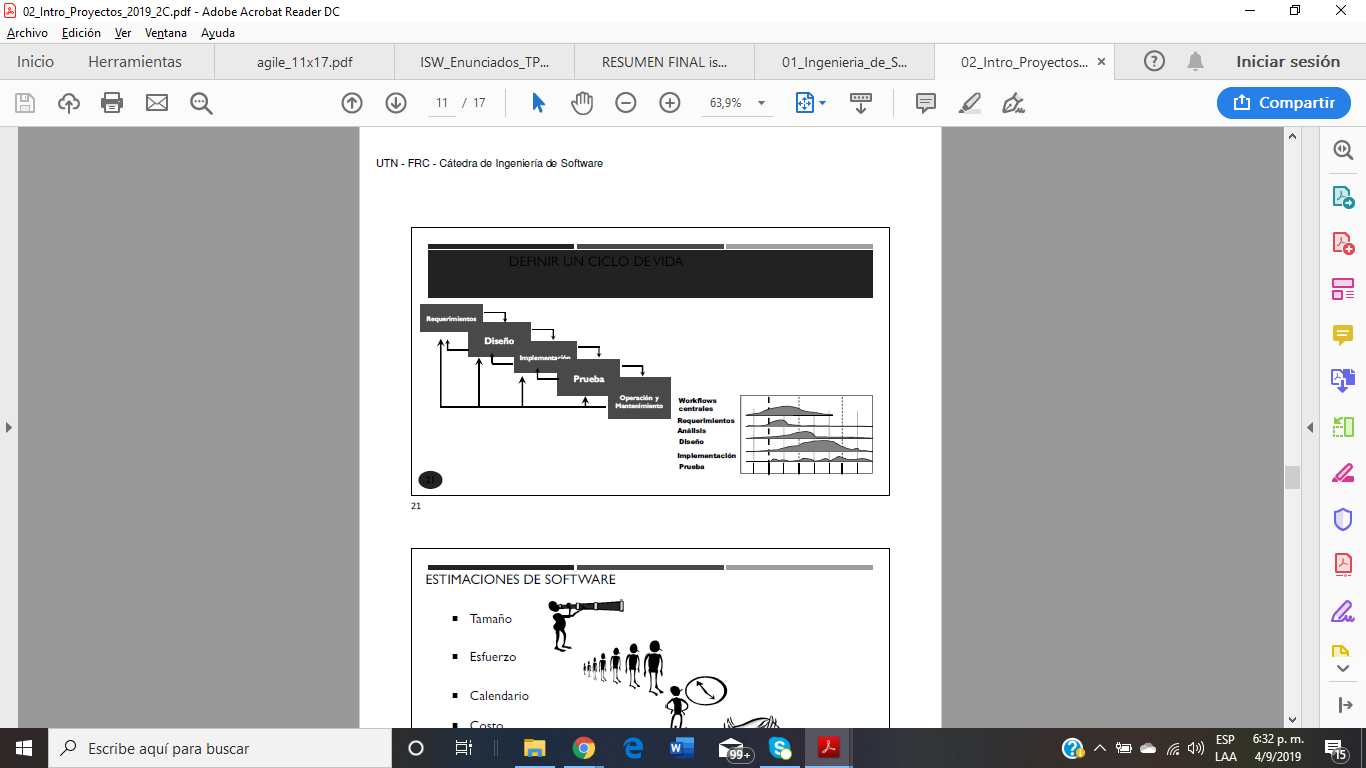
**B. Alcance del proyecto**

Es todo aquel trabajo para cumplir el objetivo del proyecto, son las tareas. Los **alcances del proyecto** se miden contra el plan (Capacitar personal, realizar seguimiento, hacer manual de usuario, etc.)

Los **alcances del producto** se refiere a lo que tiene que hacer el sistema (Gestionar cuentas, Gestionar Usuarios). Se miden contra la ERS.

**C. Definir el proceso/ciclo de vida**

El ciclo de vida del producto hace referencia a los estados por los que pasa desde que se concibe la idea hasta que se descarta. Incluye varios ciclos de vida de proyecto.



**D. Estimaciones**

**Tamaño:** tamaño del producto en términos del proyecto (Contar lineas de codigo, contar módulos, contar software funcionando y de valor, contar requerimientos.).

**Esfuerzo:** Horas personas lineales que se necesitan para terminar el proyecto. Acá tenemos que tener en cuenta que una persona no está trabajando al 100% todo el tiempo, que pasa tiempo viendo redes sociales, conversando, almorzando, etc.

**Tiempo:**

**Costo:** Estimación monetaria.

**E. Riesgo:** Probabilidad de que ocurra un evento que impacte negativamente en el proyecto. Debemos tener en cuenta la exposición (Probabilidad\*Impacto) y en base a esto saber si puedo mitigar la probabilidad, el impacto o ninguna, es decir establecer una contingencia. También hay que saber cuando mitigar (Evento disparador).

**F. Calendarización:** Establecer las fechas para el proyecto. MS project, tener en cuenta que un proyecto se atrasa de un día a la vez.

**G. Métrica:** Se planifican las métricas que se quieren medir.

* Métricas en proyecto
* Métricas en producto

**H. Seguimiento y control:**

* Reuniones
* Reportes
* Cantidad de reportes
* Cantidad de reuniones

**Principios Ágiles**

1. *Entregas tempranas y frecuentes de software con valor*

Cuando hablamos de valor nos referimos a que sirva al cliente, de forma tal que satisface sus necesidades. Reduce el trabajo y mejora el feedback.

1. *Recibir cambios de requerimientos en etapas finales*

Se analiza el impacto del cambio y se toma la decisión.

1. *Release frecuentes*

Hacer entregas de 2 semanas a un mes

1. *Técnicos y no técnicos trabajando juntos.*

El cliente trabajando junto al equipo.

1. *Hacer proyectos con individuos motivados.*

Foco en las personas y en los procesos empíricos.

1. *Comunicación cara a cara.*

La gran parte de la clave de la comunicación está en la no verbal.

1. *Mejor métrica de progreso, la cantidad de software funcionando.*

El conjunto de métricas se reduce a la cantidad de software funcionando.

1. *Ritmo de desarrollo sostenible en el tiempo.*

Se complementa con los release frecuentes. El tiempo se mantiene siempre a lo largo del proyecto.

1. *Atención continua a la excelencia técnica.*

Se relaciona con la calidad del producto.

1. *Maximizar el trabajo no hecho.*

Lo que no entrega valor al cliente no se hace

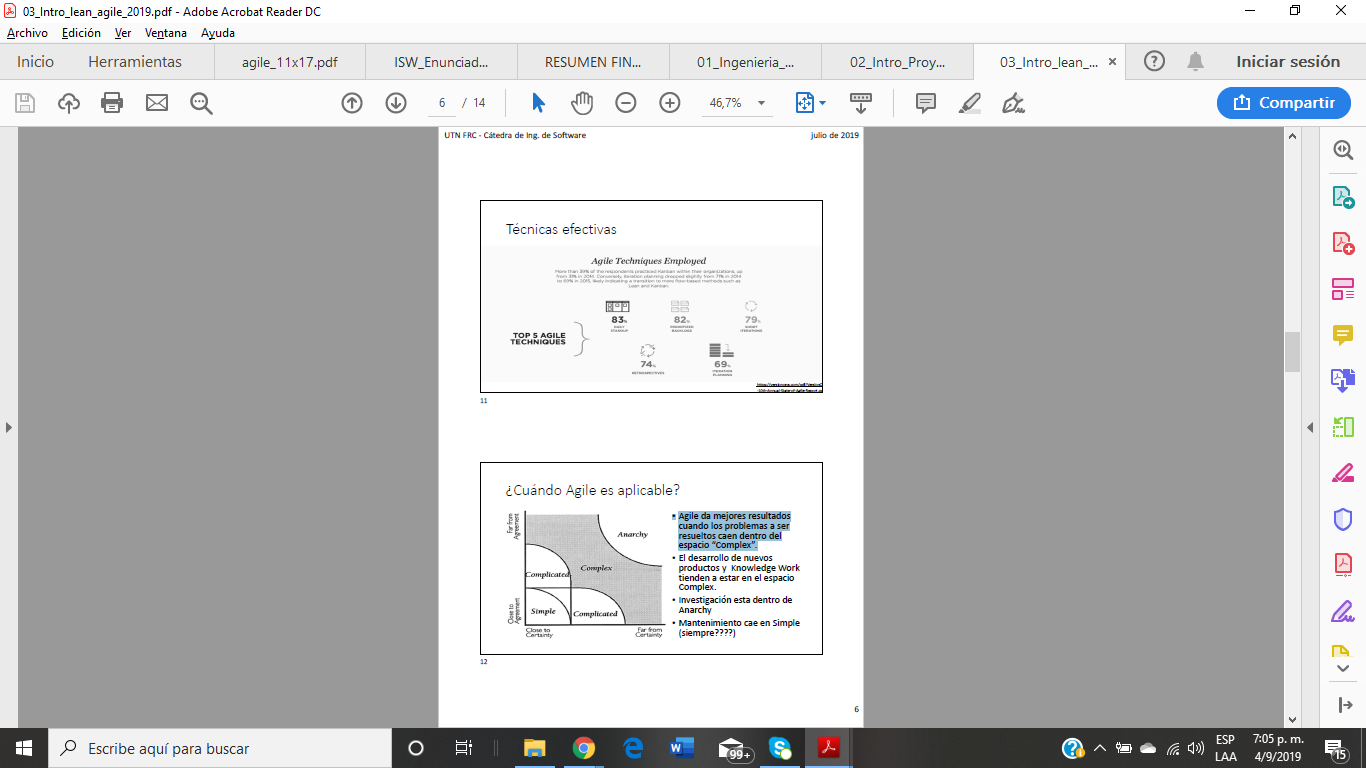
1. *Equipos auto-organizados*

No se construye el software de calidad sin un equipo con personas que tienen la capacidad de poder desarrollar un software de punta a punta.

1. *A intervalos regulares, analizo cómo viene trabajando el equipo y ajustó*

**Cuando Agile es aplicable**

Agile da mejores resultados cuando los problemas a ser resueltos caen dentro del espacio complejo.



**Principios Lean**

1. *Eliminar desperdicio*

20% del software satisface el 80% de las necesidades.

Evitar estimaciones, trabajo parcialmente hecho, tiempos muertos

1. *Amplificar el aprendizaje*

Cultura mejoramiento continuo y solución de problemas

1. *Embeber la integridad conceptual*

Todas las unidades se puedan integrar en un mismo todo. Mantener en el producto un lineamiento, estándares.

1. *Diferir compromisos*

Postergar las decisiones hasta el último momento responsable, de forma que no afecte la calidad del producto.

1. *Entregar rápido y entregar rápidamente*

Entregar en el momento que estén listas (Entregar rápido)

Entregar en el momento en donde pueda tomar decisiones (Entregar rápidamente)

1. *Dar poder al equipo*

Líderes, delegar responsabilidades, ética laboral.}

1. *Ver el todo*

Tener en cuenta y analizar el contexto en el que se inserta el producto.

**3 Dimensiones para controlar el proyecto**

Enfoque tradicional



* En el enfoque tradicional se fijan los alcances y en base a esto se derivan el tiempo y los recursos.
* El problema es que los requerimientos cambian.

Enfoque ágil



* Just in time en los requerimientos: se identifican los requerimientos que se van a realizar en esa iteración y luego hay retroalimentación.

**User Stories**

Principios relacionados de los Requerimientos Ágiles

*Principios Ágiles*

1- La prioridad es satisfacer al cliente a través de releases tempranos y frecuentes (2 semanas a un mes)

2 -Recibir cambios de requerimientos, aun en etapas finales

4 - Técnicos y no técnicos trabajando juntos TODO el proyecto

6 - El medio de comunicación por excelencia es cara a cara

11 - Las mejores arquitecturas, diseños y requerimientos emergen de equipos autoorganizados

*Principios Lean*

1- Eliminar el desperdicio

4 - Diferir Compromisos

6 - Ver el todo

Partes de una User Stories

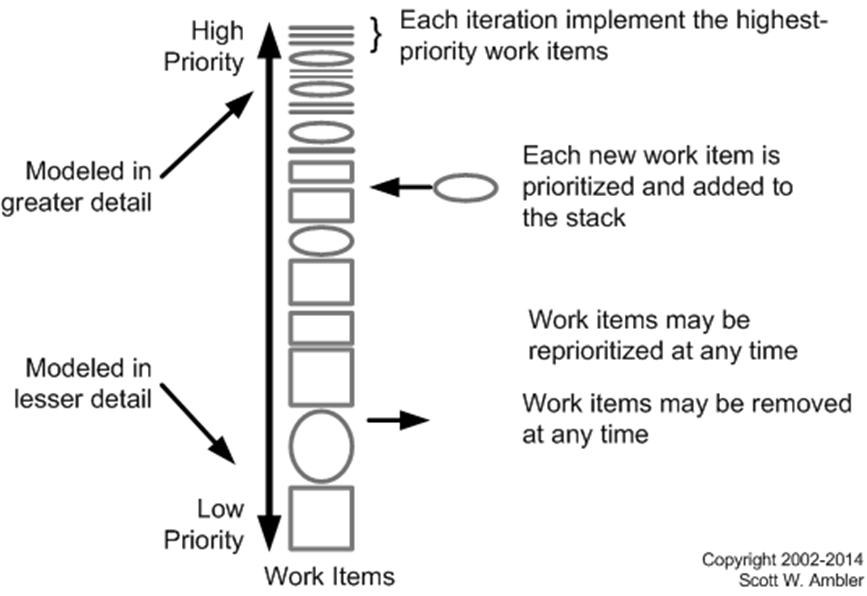
* *Conversación* (más importante): en la teoría no queda registrado.
* *Tarjeta*: parte visible.
* *Confirmación*: con el product owner a cerca de las pruebas de confirmación.

Las User Stories son Multipropósito

Las historias son:

* Una necesidad del usuario.
* una descripción del producto
* un ítem de planificación
* Token para una conversación
* Mecanismo para diferir una conversación

El product owner prioriza las historias en el product Backlog



* En cada iteración se implementan los ítems de mayor prioridad.
* Cada nuevo requerimiento es priorizado y agregado al product Backlog (lista)
* Importa la granularidad (los de fina tienen más prioridad)
* Los ítems pueden ser priorizados en cualquier momento.

Modelado de Roles

* Tarjeta de rol de usuario
* Personas
* Personajes extremos

Criterios de aceptación

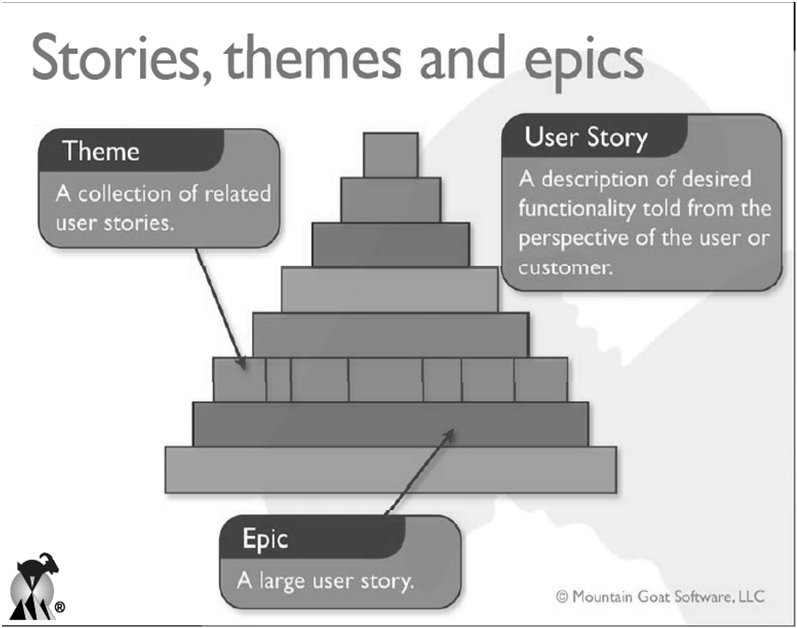
* Acuerdo de cómo se debe comportar el software para que el product owner acepte lo implementado.
* Se pone como una Nota en la User Story
* Del otro lado de la tarjeta se ponen las pruebas.
  + Son declaraciones de lo que tengo que probar
  + Se prueban escenarios “felices” (pasa) y “negativos” (falla)

ALGO MÁS SOBRE LAS USER STORIES…

* No son especificaciones detalladas de requerimientos (como los casos de uso)
* Son expresiones de intención, “es necesario que haga algo como esto…”
* No están detallados al principio del proyecto, elaborados evitando especificaciones anticipadas, demoras en el desarrollo, inventario de requerimientos y una definición limitada de la solución.
* Necesita poco o nulo mantenimiento y puede descartarse después de la implementación.
* Junto con el código, sirven de entrada a la documentación que se desarrolla incrementalmente después.

Diferentes niveles de abstracción

Se mide la granularidad en relación si entra en una iteración.



Definición de Hecho (Definition of Done)

* La user story está lista para presentársela al cliente.
* Se suele hacer un check list a cerca de las actividades que deben realizarse.

Definición de listo (Definition of Ready)

* Si cumple el criterio de aceptación entra en iteración de desarrollo.

*INVEST MODEL:* Si la característica no cumple con el Invest no puede entrar en la iteración.

* *Independiente*: Calendarizables e implementables en cualquier orden
* *Negociable*: El “que” no el “cómo”
* *Valuable*: Debe tener valor para el cliente
* *Estimable*: Ayuda al cliente a armar un ranking de costos
* *Small*: Debe ser consumida en un iteración
* *Testeable*: Demostrar que fueron implementadas.

*Spikes*

Es un tipo especial de user story, utilizado para quitar riesgo e incertidumbre de una User story. Se pueden utilizar para:

* Analizar comportamiento de una historia compleja y poder dividirla en piezas manejables.
* Inversión básica para familiarizar al equipo con una nueva tecnología o dominio.
* Ganar confianza frente a riesgos tecnológicos
* Frente a riesgos funcionales, donde no está claro como el sistema debe resolver la interacción con el usuario para alcanzar el beneficio esperado.

Se pueden clasificar en Técnicas o funcionales.

* ***Técnicas:***Utilizadas para investigar enfoques técnicos en el dominio de la solución. (Decisión hacer o comprar, evaluar performance potencial).

También se utilizan en cualquier situación en la que el equipo necesite una compresión más fiable antes de comprometerse a una nueva funcionalidad

* ***Funcionales:***Utilizadas cuando hay cierta incertidumbre respecto de cómo el usuario interactúa con el sistema. Usualmente son mejor evaluadas con prototipos para obtener retroalimentación de los usuarios o involucrados.

Las spikes deben ser estimables, demostrables y aceptables. Implementar la spike en una iteración separada de las historias resultantes.

**Gestión de configuración (SCM - Software Configuration Management -)**

*Ítem de configuración*

Cualquier elemento que es de interés para mantener la trazabilidad. Va evolucionando conforme a las distintas versiones.

*Configuración*

Conjunto de ítems de configuraciones con su versión en un momento dado.

*Repositorio*

Versión→ Conjunto de cambios definen la versión

El repositorio permite que diferentes versiones coexistan en un ítem

*MVP*

* Producto mínimo visible.
* Las mínimas funcionalidades para sacar al mercado y que obtener un feedback.