# Ingeniería de Software (incluye papers Orphans Preferred - No Silver Bullet - Software’s Ten Essentials)

#### ¿Qué es software?

Programas de cómputo y documentación asociada. Los productos de software se desarrollan para un cliente en particular o para un mercado en general.

**¿Cuáles son los atributos del buen software?**

El buen software debe entregar al usuario la funcionalidad y el desempeño requeridos, y debe ser sustentable, confiable y utilizable.

**¿Qué es ingeniería de software?**

La ingeniería de software es una disciplina de la ingeniería que se interesa por todos los aspectos de la producción de software, desde las primeras etapas de la especificación del sistema hasta el mantenimiento del sistema después de que se pone en operación.

**¿Cuáles son las actividades fundamentales de la ingeniería de software?**

Especificación, desarrollo, validación y evolución del software.

**¿Cuál es la diferencia entre ingeniería de software y ciencias de la computación?**

Las ciencias de la computación se enfocan en teoría y fundamentos; mientras la ingeniería de software se enfoca en el sentido práctico del desarrollo y en la distribución de software.

**¿Cuál es la diferencia entre ingeniería de software e ingeniería de sistemas?**

La ingeniería de sistemas se interesa por todos los aspectos del desarrollo de sistemas basados en computadoras, incluidos hardware, software e ingeniería de procesos. La ingeniería de software es parte de este proceso más general.

**¿Cuáles son los principales retos que enfrenta la ingeniería de software?**

Se enfrentan con una diversidad creciente, demandas por tiempos de distribución limitados y desarrollo de software confiable.

**¿Cuáles son los costos de la ingeniería de software?**

Aproximadamente 60% de los costos del software son de desarrollo, y 40% de prueba. Para el software elaborado específicamente, los costos de evolución superan con frecuencia los costos de desarrollo.

**Cuáles son los mejores métodos y técnicas de la ingeniería de software?**

Aun cuando todos los proyectos de software deben gestionarse y desarrollarse de manera profesional, existen diferentes técnicas que son adecuadas para distintos tipos de sistema.

Por ejemplo, los juegos siempre deben diseñarse usando una serie de prototipos, mientras que los sistemas críticos de control de seguridad requieren de una especificación completa y analizable para su desarrollo. Por lo tanto, no puede decirse

que un método sea mejor que otro

**¿Qué diferencias ha marcado la Web a la ingeniería de software?**

La Web ha llevado a la disponibilidad de servicios de software y a la posibilidad de desarrollar sistemas basados en servicios distribuidos ampliamente. El desarrollo de sistemas basados en Web ha conducido a importantes avances en lenguajes de programación y reutilización de software.

### Características del producto (características del software)

#### Mantenimiento

El software debe escribirse de tal forma que pueda evolucionar para satisfacer las necesidades cambiantes de los clientes. Éste es un atributo crítico porque el cambio del software es un requerimiento inevitable de un entorno empresarial variable.

#### Confiabilidad y seguridad

La confiabilidad del software incluye un rango de características que abarcan fiabilidad, seguridad y protección. El software confiable no tiene que causar daño físico ni económico, en caso de falla del sistema. Los usuarios malintencionados no deben tener posibilidad de acceder al sistema o dañarlo

#### Eficiencia

El software no tiene que desperdiciar los recursos del sistema, como la memoria y los ciclos del procesador. Por lo tanto, la eficiencia incluye capacidad de respuesta, tiempo de procesamiento, utilización de memoria, etcétera.

#### Aceptabilidad

El software debe ser aceptable al tipo de usuarios para quienes se diseña. Esto significa que necesita ser comprensible, utilizable y compatible con otros sistemas que ellos usan

### Proceso de software

Un proceso de software es una secuencia de actividades que conducen a la elaboración de un producto de software.

El enfoque sistemático que se usa en la ingeniería de software

**Actividades fundamentales que son comunes a todos los procesos de software (actividades de alto nivel de especificación)**

**Especificación del software**, donde clientes e ingenieros definen el software que se producirá y las restricciones en su operación.

**Desarrollo del software**, donde se diseña y programa el software.

**Validación del software,** donde se verifica el software para asegurar que sea lo que el cliente requiere.

**Evolución del software,** donde se modifica el software para reflejar los requerimientos cambiantes del cliente y del mercado.

### Puntos clave

* La ingeniería de software es una disciplina de ingeniería que se interesa por todos los aspectos de la producción de software.
* El software no es sólo un programa o programas, sino que también incluye documentación. Los **atributos esenciales** de los productos de software son **mantenimiento, confiabilidad, seguridad**, **eficiencia y aceptabilidad.**
* El proceso de software incluye todas las actividades que intervienen en el desarrollo de software**. Las actividades de alto nivel de especificación, desarrollo, validación y evolución** son parte de todos los procesos de software.
* Las **nociones fundamentales** de la ingeniería de software son universalmente aplicables a todos los tipos de desarrollo de sistema. Dichos fundamentos incluyen **procesos, confiabilidad, seguridad, requerimientos y reutilización de software.**
* Existen muchos tipos diferentes de sistemas y cada uno requiere para su desarrollo de herramientas y técnicas adecuadas de ingeniería de software. Existen pocas, si es que hay alguna, técnicas específicas de diseño e implementación que son aplicables a todos los tipos de sistemas.
* Las ideas fundamentales de la ingeniería de software son aplicables a todos los tipos de sistemas de software. Dichos fundamentos incluyen **procesos de administración de software, confiabilidad y seguridad del software, ingeniería de requerimientos y reutilización de software.**
* Los ingenieros de software tienen responsabilidades con la profesión de ingeniería y la sociedad. No deben preocuparse únicamente por temas técnicos.
* Las sociedades profesionales publican códigos de conducta que establecen los estándares de comportamiento esperados de sus miembros.

### Software’s Ten Essentials

Su finalidad es evitar problemas y en caso de que ocurran, minimizar los daños.

#### Una especificación de producto

La brújula del software. Sin una guía que marque el camino, por más que el trabajo sea el mejor, no se va a poder construir un software que funcione de la mejor forma.

#### Un prototipo de UI detallado

Con los sistemas actuales, resulta casi obligatorio realizar prototipo de UI detallado para capturar la esencia del sistema a construir. Esto sería como un mapa que detalla las dificultades del terreno.

#### Un calendario realista

Esto sirve para saber que es todo lo que necesito para poder cumplir con este calendario. Es decir, sirve para planificar las personas que se involucraron, los tiempos de QA y que sean adecuadas y el nivel de formalidad en los procesos. Todo esto ayuda a evitar que se retrasan las entregas del proyecto y estar preparados para posibles situaciones adversas.

#### Prioridades explícitas

Cuando ocurre un problema, se sabe que es esencial y qué cosas deben ser priorizadas. De esta forma saber que partes del sistema deberían tener la mejor calidad posible y a cuales dedicarle menos tiempo.

#### Manejo activo de riesgos

Es bueno espera lo mejor pero se debe estar preparado para lo peor. El manejo activo de los riesgos es esencial para un software exitoso. Si no se atacan los riesgos, estos te atacan a vos.

#### Plan de QA (Aseguración de la calidad)

El kit de primeros auxilios del sistema. La primer prioridad es intentar evitar el uso de este kit de primeros auxilios. Pero en caso de necesitarlos, se deben usar lo antes posible (en lo posible apenas se insertaron). Si no se toma acción rápidamente, llega un punto donde no se puede ignorar más un defecto y para corregirlo implicaría un gran costo.

#### Lista de actividades detallada

Como un itinerario. En estas listas se detallan tareas lo más atómicas posibles que pueden ser realizadas en unos pocos días y que puedan ser consideradas como hechas o no hechas (No x% hecha). Si comparamos la lista de actividades completas y la lista de las planificadas, tenemos un indicador de si el proyecto está en tiempo.

#### Manejo de configuración del Software

No te asegura estar a salvo pero ayuda a no caer/sucumbir a algunos de los mayores riesgos de un proyecto de software. Por ejemplo, integración continua, git, etc.

#### 

#### Arquitectura de Software

Si no se hace foco en este punto, un software parecería a una mochila que se parcho tanto con cinta que no se diferencia que es parte de la mochila original. Promueve el diseño consistente y la aproximación de implementaciones, lo que se refleja en mayor facilidad de futura correcciones y extensiones. Es decir, ayuda a la consistencia, la cual es muy difícil de conseguir si no se tiene una buena arquitectura.

#### Plan de integración

Uno de los mayores problemas es la integración componentes de software que no fueron tomados en cuenta en el diseño original.

### 

# Software

Software, en general, es un set de programas y la documentación que acompaña.

**Tipos de software:** ???

* System software
* Utilitarios
* Software de aplicación

**Disciplinas Técnicas**

* Requerimientos
* Análisis y diseño
* Construcción
* Prueba
* Despliegue

**Disciplinas de Gestión**

* Planificación de proyecto
* Monitoreo y control de proyectos

**Disciplinas de Soporte**

* Gestión de Configuración de Software
* Aseguramiento de la calidad
* Métricas

## El proceso de Software

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

**Proceso:** La secuencia de pasos ejecutados para un propósito dado (IEEE)

**Proceso de Software:** Un conjunto de actividades, métodos, prácticas, y transformaciones que la gente usa para desarrollar o mantener software y sus productos asociados (Sw-CMM)

# 

# 

# 

# 

# Procesos Definidos vs. Procesos Empíricos

**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 llegar a obtener resultados diferentes.
* La administración y control es a través de inspecciones frecuentes y adaptaciones
* Son procesos que trabajan

**Patrón de conocimiento en procesos empíricos**

* Asumir
* Construir
* Retroalimentar
* Revisar
* Adaptar
* Asumir

**Ciclos de vida**

La serie de pasos a través de los cuales el producto o proyecto progresa

**Hay tres tipos básicos de Ciclos de Vida (ciclo de vida)**

* Secuencial (en cascada???)
* Iterativo/Incremental (espiral???)
* Recursivo

# Componentes del software

* Programas
* Procedimientos
* Reglas
* Documentación
* Datos

### Proyecto

* Están orientados a objetivos
* Tienen una duración limitada en el tiempo, tienen principio y fin.
* Implican tareas interrelacionadas basadas en esfuerzos y recursos.
* Son únicos
* Los proyectos están dirigidos a obtener resultados y ello se refleja a través de objetivos.
* Los objetivos guían al proyecto
* Los objetivos no deben ser ambiguos
* Un objetivo claro no alcanza…debe ser también alcanzable.
* Los proyectos son temporarios,

**Son únicos**

* Todos los proyectos por similares que sean tienen características que los hacen únicos

### ADMINISTRACIÓN DE PROYECTOS

Es la aplicación de conocimientos, habilidades, herramientas y técnicas a las actividades del proyecto para satisfacer los requerimientos del proyecto.

**Administrar un proyecto incluye:**

* Identificar los requerimientos
* Establecer objetivos claros y alcanzables
* Adaptar las especificaciones, planes y el enfoque a los diferentes intereses de los involucrados (stakeholders).

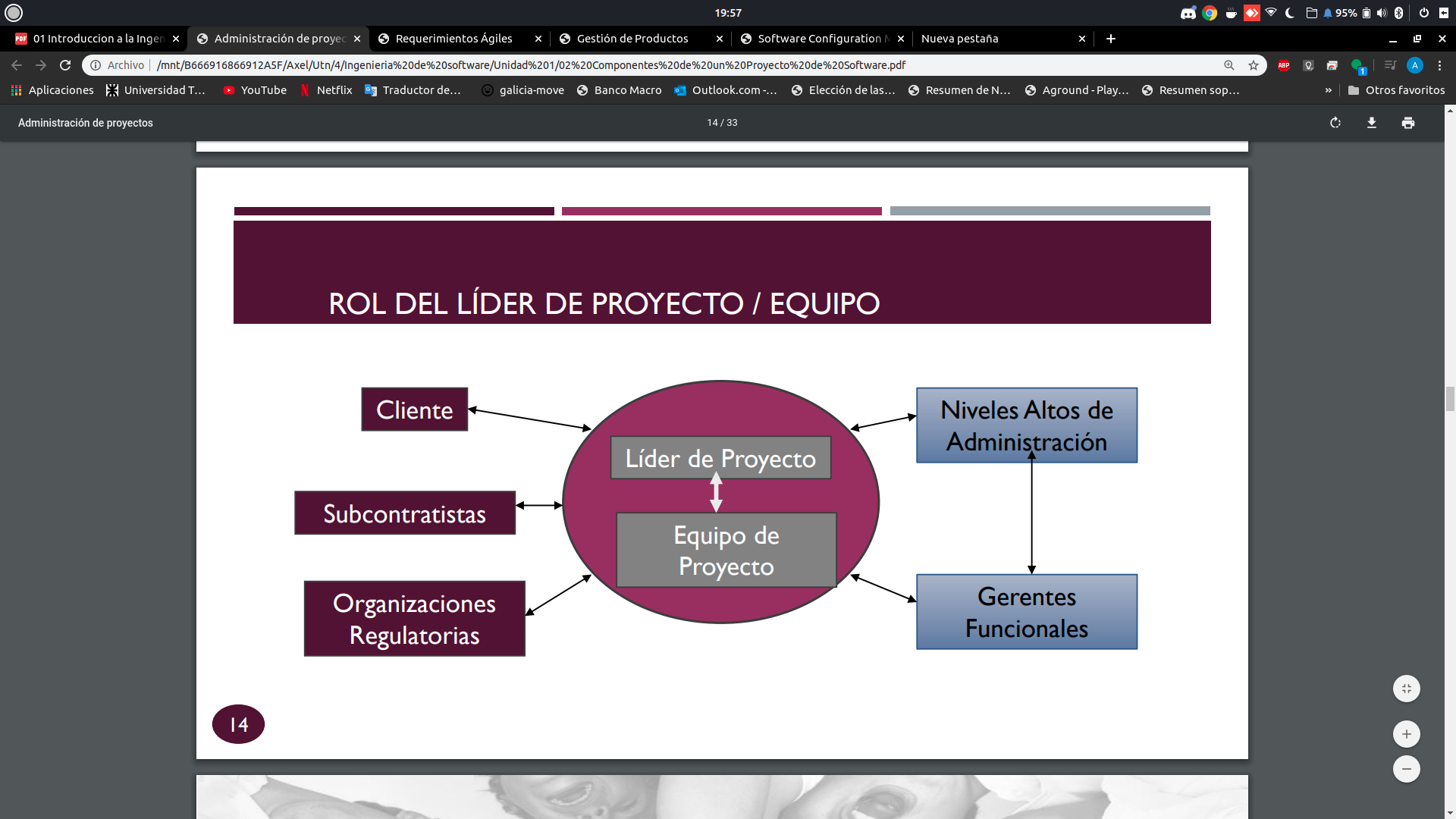
**LA RESTRICCIÓN TRIPLE” (THE TRIPLE CONSTRAIN)**

* Objetivos de proyecto: que está el proyecto tratando de alcanzar?
* Tiempo: cuánto tiempo debería llevar completarlo?
* Costos: cuánto debería costar?

Es responsabilidad del Líder de proyecto balancear estos tres objetivos

**Producto Software:** Cada nueva versión es desarrollada incrementalmente en una serie de pasos

**ROL DEL LÍDER DE PROYECTO / EQUIPO**



### Plan de proyecto

Incluye:

¿Qué es lo que hacemos?

¿Cuándo lo hacemos?

¿Cómo lo hacemos?

¿Quién lo va a hacer?

### ¿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

## DEFINICIÓN DEL ALCANCE

**Alcance del Producto:** Son todas las características que pueden incluirse en un producto o servicio.

**Alcance del Proyecto:**  Es todo el trabajo y solo el trabajo que debe hacerse para entregar el producto o servicio con todas las características y funciones especificadas.

### Cómo se mide?

**El cumplimiento del Alcance del Proyecto:** Se mide contra el Plan de Proyecto (o Plan de Desarrollo de Software).

**El cumplimiento del Alcance del Producto:** Se mide contra la Especificación de Requerimientos.

ESTIMACIONES DE SOFTWARE

* Tamaño
* Esfuerzo
* Calendario
* Costo
* Recursos Críticos

## RIESGO

* Problema esperando para suceder
* Evento que podría comprometer el éxito del proyecto

### GESTIÓN DE RIESGOS

## FACTORES TOP PARA EL ÉXITO DE UN PROYECTO

* Monitoreo & Feedback
* Tener una misión/objetivo claro
* Comunicación

## CAUSAS DE FRACASOS EN PROYECTOS

* Fallas al definir el problema
* Planificar basado en datos insuficientes
* La planificación la hizo el grupo de planificaciones
* No hay seguimiento del plan de proyecto
* Plan de proyecto pobre en detalles
* Planificación de recursos inadecuada
* Las estimaciones se basaron en “supuestos” sin consultar datos históricos Nadie estaba a cargo

# Manifiesto Ágil

# 

# Son mejores formas de desarrollar software, ayudando a otros mientras lo hacemos.

# 

### Valores del manifiesto Ágil

# **INTERACCIÓN Y PERSONAS** sobre procesos y herramientas

# **PRODUCTO ENTREGADO Y FUNCIONAL** por sobre documentación comprensiva

# **PERSONAS INTERRELACIONADAS (COLABORACIÓN CON CLIENTE)** sobre contrato de negociación

# **RESPONDER AL CAMBIO** sobre seguir al plan.

# 

### Principios del manifiesto ágil, Principios ágiles

# 

# Satisfacer al cliente con entregas frecuentes y tempranas

# Cambios de requerimientos son bienvenidos

# **RELEASES FRECUENTES (DE 2 A 4 SEMANAS)**

# Técnicos y no técnicos juntos

# **INDIVIDUOS MOTIVADOS**

# Medio comunicación: Cara a Cara

# **MÉTRICA DE PROGRESO: SOFTWARE FUNCIONANDO**

# **RITMO DE DESARROLLO SOSTENIBLE**

# **ATENCIÓN CONTINUA A LA EXCELENCIA TÉCNICA**

# **SIMPLICIDAD: MAXIMIZACIÓN DEL TRABAJO NO HECHO**

# Arquitecturas diseños y requerimientos emergentes

# **A INTERVALOS REGULARES EL EQUIPO EVALÚA SU DESEMPEÑO**

# 

# Requerimientos Ágiles

**Usar el "valor" para construir el producto correcto** --> Usar historias y modelos para mostrar que construir --> Determinar que es "Solo lo suficiente"

El **desperdicio está en todos lados**, sea grande o pequeño el producto, sea exitoso o fracase.

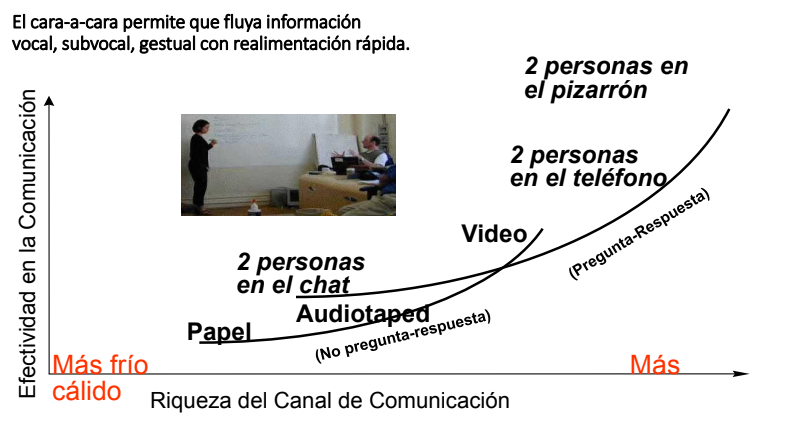
### Gestión Ágil de Requerimientos de Software

**Los requisitos cambiantes son una ventaja competitiva si puede actuar sobre ellos y los mismos tienen una prioridad alta o baja**.

Se puede pensar a la **prioridad** como si fuese una pila, donde cada nuevo requerimiento es priorizado y agregado a la pila.

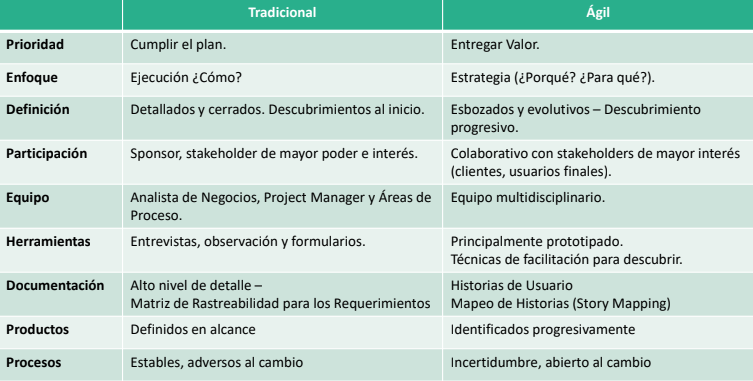
Recordemos que la priorización puede hacerse en cualquier momento y por igual los requerimientos pueden irse en cualquier momento también.

**La técnica cara-a-cara** permite que fluya información vocal, subvocal, gestual con realimentación rápida.



Recordemos que **Valor es la obtención de beneficio tangible o intangible, y al valor lo asociamos con utilidad, beneficio satisfacción de los usuarios.**

### Diferencias con Metodologías Ágiles vs Tradicional:



**La metodología tradicional** está dirigido por un plan, y el mismo tiene fijo solo los requerimientos y se estiman recursos y tiempo, mientras que la **metodología agile** es dirigida por valor y tiene fijo los recursos y tiempo y va estimando el alcance del proyecto a medida que requerimientos cambian por ejemplo.

### Tipo de requerimientos ágiles:

**Requerimiento de negocio**: Disminuir un X% de tiempo invertido en procesos manuales relacionados con atención al cliente.

**Requerimiento de usuario**: Realizar consultas en línea del estado de cuenta de los clientes

**Requerimiento funcional**: Generar reporte de saldos de cuenta. Recibir notificaciones por email

**Requerimiento no funcional**: Formato del reporte PDF. Cumplir niveles de seguridad para credenciales de usuarios según la ley bancaria 9999xx

**Requerimiento de implementacion**: Servidores en la nube

(Todo eso fueron ejemplos no definiciones estaría bueno buscar en alguna bibliografía y completar ??? )

En resumen

Entendiendo la necesidad y el negocio --> Descubriendo la solución de forma colaborativa --> Junto a un equipo motivado y competente --> Entregamos frecuentemente valor a los stakeholders

**Los cambios son la única constante**

Stakeholders: No son todos los que están.

Siempre se cumple eso de que "El usuario dice lo que quiere cuando recibe lo que pidió"

No hay técnicas ni herramientas que sirvan para todos los casos

Lo importante no es entregar una salida, un requerimiento, lo importante es entregar un resultado una solución de valor

### Los principios ágiles relacionados a los requerimientos agile son:

* La prioridad es satisfacer al cliente a través de releases tempranos y frecuentes
* Recibir cambios de requerimientos, aun en etapas finales
* Técnicos y no técnicos trabajando juntos TODO el proyecto
* El medio de comunicación por excelencia es cara a cara
* Las mejores arquitecturas, diseños y requerimientos emergen de equipos autoorganizados

# Estimaciones

**Las estimaciones surgen para darle valor de tamaño relativo a las user stories, utilizando algo que se llama Story Points (SP)**

No son medidas basadas en tiempo, tiene que quedar en claro

Estas estimaciones son **relativas** porque las personas no pueden estimar en valores absolutos y precisos, mucho menos si no se ofrece una comparación a la hora de estimar. Este proceso es mejor hecho en grupo

En el caso de las estimaciones el tamaño va a indicar una medida de cantidad de trabajo necesaria para hacer algo, es decir que **el tamaño indica cuán compleja es una US y cuánto trabajo se va a necesitar para crearla**

Hay que recordar que **TAMAÑO NO ES ESFUERZO**.

Ver bien la diferencia

Por lo general se recomienda no estimar basado en tiempo, porque hay muchas variables en juego como habilidades, conocimiento, asunciones, etc de las cuales no podemos estar 100% seguros

Para estimar este tamaño que las US poseen se va a usar una **escala QUE NO DEBE SER CAMBIADA UNA VEZ ELEGIDA** y por lo general se usa Fibonacci

### Story Points

Entonces, un **Story Point es una unidad de medida específica elegida por EL EQUIPO que simboliza complejidad, riesgo y esfuerzo.**

Da idea de la complejidad de una US, y suele **incrementar exponencialmente**

Hay 3 factores que debemos tener en cuenta: COMPLEJIDAD, ESFUERZO Y DUDA

Por lo general si la complejidad sube la duda también y viceversa pero no siempre es el caso.

La velocidad es la métrica utilizada para medir el progreso del equipo (En teoría es métrica o por ende no va?)

**La duración de un proyecto no se estima**, sólo se deriva a partir de los SP que estimamos para cada US dividiéndolos por la velocidad del equipo

Los sp que se cuentan son los de las user stories resueltas hasta el momento (no casi resuelto)

### Poker Estimation

Usamos **Poker Estimation, la cual es popular en Agile y combina opiniones de experto, analogías y desagregación**

Es muy importante saber que **los participantes son los desarrolladores** porque son los más competentes del grupo que están dispuestos a resolver las tareas que son estimadas (Los que trabajan saben lo que hacen y por ende estiman)

**Usamos Fibonacci dentro de Poker Planning** (Poker Planning es un metodo de asignacion de valores que se hace con cartas mientras que los valores derivan de la sucesión de Fibonacci y pueden ser repetidos)

* **0:** No tengo idea del producto o su funcionalidad
* **1/2, 1:** Funcionalidad pequeña y por lo general cosmética
* **2-3**: Pequeño a mediano. Esto es lo que queremos (Canónica (?))
* **5**: Media. Sigue siendo lo que queremos
* **8**: Grande, se puede hacer pero se busca dividir en porciones más pequeñas
* **13**: Alguien me explica porque es tan grande?
* **20**: Con qué razón de negocio es tan grande e indivisible?
* **40**: No podemos hacer esto en un sprint
* **100**: Algo está muy mal, no arrancar.

**RECORDAR QUE AGILE ES UN PROCESO EMPÍRICO, DONDE ADAPTARSE EN MANDATORIO!**

(Falta bibliografía pero en este caso mucho no hace falta)

# 

# 

# Gestión de Productos de Software

Los productos son creados para satisfacer a los clientes, tener usuarios logueados, obtener dinero, realizar una gran visión, obtener cambios en el mundo, etc.

Por lo general no se utiliza todas las características de nuestro producto por los usuarios normales, el **80% de la gente solo usa el 20% de nuestro SW.**

En una **pirámide de abajo a arriba, un software debe evolucionar de la siguiente forma**

### Evolucion Software

Debe ser **funcional**, **confiable, usable, conveniente, placentero y significativo** (Donde la grieta de conveniente es la más difícil de cruzar, debemos enganchar al usuario y hacer que le convenga nuestro SW antes que las demás opciones del mercado)

Ver en libro si amplía algo más ???

Un producto primero surge de una idea, a la cual se le aplica un filtro estratégico para ver si conviene o no adentrarse en la misma. Luego si se decide por un si se busca la visión del proyecto final, se crea un plan de desarrollo teniendo en cuenta tiempo, presupuesto y foco del producto y finalmente se lo libera al público.

Ver planificación de productos en ambientes lean-agile ??? (Entrara? Porque es para lean)

Un aspecto importante a remarcar es el **valor vs el desperdicio**, es decir cuáles de nuestros esfuerzos crean valor y cuales son desperdicio?

Según el Lean Thinking, **valor es todo aquello que provee beneficio a los clientes.**

Por lo tanto la productividad no se mide en términos de lo que se hace cada día, sino lo que se hace correctamente cada dia.

Hay que tener **3 conceptos** en mente a la hora de hablar de gestión de productos (Creo que es lo más importante de esto)

### MVP (3 conceptos)

El **MVP (Minimum Viable Product) es lo minimo que tengo que tener para que el producto sea validado y aceptado por el cliente**. Es decir lo mínimo para conseguir aceptación por el cliente. (Permite recopilar la cantidad máxima de aprendizaje validado sobre clientes con el menor esfuerzo) El cliente prefiere ver antes de hablar sobre las capacidades del producto.

El **MRF (Minimal release feature) son las características mínimas que debe tener el producto para el primer release.**

El **MMF (Minimum Marketable feature) es lo de antes pero para ser comercializado.**

“el conjunto más pequeño posible de funcionalidad que, por si misma, tiene valor en el mercado”

En cuanto al MVP:

Valor suficiente para convencer al cliente de usarlo o comprarlo, demuestra beneficio futuro y proporciona un ciclo de retroalimentación para guiar el desarrollo futuro.

**No se debe confundir con MMF** a pesar de que ambos se enfocan en "ganar", hay riesgo de que la idea no este bien plasmada y que los clientes pierdan interés. **Viable antes que Mínimo** siempre.

Recordar que el valor está en que el cliente lo use no en solo entregarlo. (Ver Build Experiment Learn Feedback Loop si hace falta ??? )

Hay un camino para crear el MVP que va desde:

**Smoke tests:** Pruebas de humo muy simples como anuncio o algo mas

Hasta:

**Prototipos**: Producto con características faltantes o problemas

**Tips para el MVP**: En caso de duda, simplificar. No tiene que haber promesa o trabajo excesivo

**UN MVP SIRVE PARA PROBAR LAS HIPÓTESIS COMERCIALES FUNDAMENTALES, A LA HORA DE CREAR UN MVP TENEMOS QUE VALIDAR UNA HIPÓTESIS CON EL MISMO.**

Es decir se usa el MVP para experimentar y validar nuestras hipótesis en el mercado.

**Saltos de fe:**

Existen 2 tipos (?)

**Hipótesis del valor:** Prueba si el producto está entregando valor a los clientes DESPUÉS de que comienzan a usarlo, utilizando la métrica de tasa de retención (MÉTRICA NO VA)

**Hipótesis de Crecimiento**: Prueba como nuevos clientes descubrirán el producto, usando tasa de referencia o NPS (Net promoter score)

EN RESUMEN:

Para armar un MVP:

**Encontrar un nicho** --> **Crear un roadmap realist**a --> **Investigar la competencia** --> **Prevender el MVP** --> **Testear las suposiciones** --> **Asegurarse que el MVP resuelve el problema correcto** --> **Focalizar en las funcionalidades principales**

(Si SCRUM NI LEAN van entonces referirse a bibliografía Agile Product Management, MVP de Paul VII? )

# Gestión de Configuración

Recordar que el **Software** en contexto es:

### Que es el software

**Personas que se incorporan a un proyecto el cual bien obtiene un producto y se adapta a procesos, los cuales son automatizados con herramientas**.

Software, es un **conjunto de programas, procedimientos, reglas documentación y datos.**

Posee **información estructurada con propiedades lógicas y funcionales**, la misma es creada y mantenida en varias formas y representaciones. Es confeccionada para ser procesada por computadora en su estado más desarrollo.

El **software va evolucionando con el paso del tiempo**, tal vez no del todo pero sí por partes.

Evoluciona por los cambios que pueden haber en el mismo, como cambios en el negocio y nuevos requerimientos, soporte de cambios de productos asociados, reorganización de las prioridades, cambios en presupuesto, defectos, oportunidades de mejora, etc.

Recordemos que **SCM es una disciplina de soporte, transversal al proyecto con aplicación en las diferentes disciplinas (Está en todos lados)**

### Disciplinas SCM

**Control de calidad de proceso, control de calidad del producto y prueba de software.**

**El software necesita de un proceso definido o empírico**. Y el mismo proceso es el que nos lleva a la construcción del producto.

**Gestion de configuracion es una actividad de aseguramiento de calidad.**

### Definición SCM

**En definición SCM es:** Una disciplina que aplica dirección y monitoreo administrativo y técnico a: identificar y documentar las características funcionales y técnicas de los ítems de configuración, controlar los cambios de esas características, registrar y reportar los cambios y su estado de implementación y verificar correspondencia con los requerimientos.

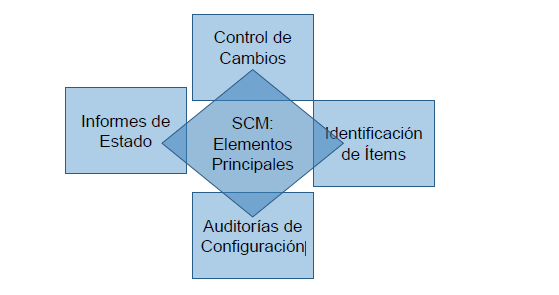
Se debe gestionar la configuración para establecer y mantener la integridad de los productos de software a lo largo de su ciclo de vida.

Involucra para la config: Identificarla en un momento dado, controlar sistemáticamente sus cambios, mantener su integridad y origen

**Integridad del producto satisface las necesidades del usuario**, puede ser fácil y completamente rastreado durante su ciclo de vida, satisface criterios de performance, cumple con sus expectativas de costo. En general saber que el sw es un blanco móvil

**Problemas en el manejo de componentes** como la pérdida de uno, perdida de cambios (no tengo el último componente), falla la sincronía fuente objeto ejecutable, cambios no validados, etc.

### Actividades Fundamentales SCM



### Ítem de Configuración de Software (SCI)

**Es todos y cada uno de los artefactos que forman parte del producto o del proyecto, que pueden sufrir cambios o necesitan ser compartidos entre los miembros del equipo y sobre los cuales necesitamos conocer su estado y evolución**

Son documentos de diseño, código fuente, código ejecutable, etc

#### Versión

**Una versión** se define, desde el punto de vista de la evolución, como la forma particular de un artefacto en un instante o contexto dado.

#### Control de Versiones

**El control de versiones** se refiere a la evolución de un único ítem de configuración (IC), o de cada IC por separado.

#### Variante

**Una variante** es una versión de un ítem de configuración (o de la configuración) que evoluciona por separado. Estas variantes representan configuraciones alternativas. Recordar que un producto de sw puede adoptar distintas configuraciones dependiendo el lugar donde se instale.

#### Línea Base

**La Línea base o baseline** no es la versión del producto. Indica una configuración estable y a la que puedo volver en un determinado momento que conserva las características que poseen la integridad de mi producto

Sirve de base para desarrollos posteriores y puede cambiarse sólo a través de un procedimiento formal de control de cambios, permite volver en el tiempo a un momento específico.

#### Configuración del Software

**La configuración del Software es un conjunto de ítems de configuración con su correspondiente versión en un momento determinado.**

#### Ramas

**Las ramas** dentro de un proyecto bifurcan el desarrollo, permitiendo razones de creación con semántica, experimentación, y permiten ver que se va a agregar y que no al proyecto.

**Merge junta ramas**, lleva cambios a rama principal

#### Repositorio

**Un repositorio de información** contiene los ítems de configuración (ICs). Puede ser una o varias bases de datos, permite mantener la historia de cada IC con sus atributos y relaciones.

Existen los **repositorios centralizados** (un servidor contiene los archivos con sus versiones y si falla falla todo) y los **repositorios descentralizados** (cada cliente tiene una copia exacta del repositorio completo y si falla no falla todo, se copia y pega de un repo al otro, es más flexible)

En resumen **SCM Está compuesto por control de cambios, informes de estado, identificación de ítems y auditorias de configuración.**

En cuanto a **los ítems de configuración, tienen que ser identificados unívocamente**, debe existir una regla de nombrado o convención a la hora de mencionar un ítem. Se debe definir una estructura general en el repositorio para mantener el orden en el pipeline de trabajo.

Ejemplo de convención/regla:

**BC\_CP\_[NombreCP]\_[NroCasoPrueba]\_[Nro\_CU]\_[Escenario]**

Los mismos pueden identificarse en el producto, proyecto y siguientes iteraciones del mismo.

Ver proceso de control de cambios ???

#### Control de cambios

**Control de cambios** tiene su origen en un Requerimiento de cambio a uno o varios ítems de configuración que se encuentran en una línea base. Es un procedimiento formal, que involucra diferentes actores y una evaluación del impacto del cambio.

**El comité de control de cambios está formado por**: Area analisis y diseño, Area implementacion, Área Testing y otros posibles interesados

Existen dos diferentes **auditorias de configuración**:

### Auditorías de Configuración

**Auditoria Fisica(PCA)**: Asegura que lo que está indicado para cada ICS en la línea base o en la actualización se ha alcanzado realmente.

**Auditoria Funcional(FCA)**: Es una evaluación independiente de los productos de software, controlando que la funcionalidad y performance reales de cada ítem de configuración sean consistentes con la especificación de requerimientos.

Ver bien ambas auditorías en bibliografía ???

Las auditorias de gestion de configuracion sirven a la Validacion y Verificacion

### Validacion y Verificacion

**La validación** es chequear que el problema sea resuelto de manera apropiada y que el usuario obtenga el producto correcto. Lo que se hace, se hace bien y se quiere lograr que se cumpla el objetivo (Es lo que estamos haciendo lo adecuado para el negocio/proyecto?)

**La verificación** asegura que un producto cumple con los objetivos preestablecidos, definidos en la documentación de líneas base. Todas las funciones son llevadas a cabo con éxito y los test cases tienen el status "OK" (VER TEST CASES) o bien consten como "problemas reportados" en al nota de release. Comprobamos que lo que decimos lo hacemos (Estamos haciendo lo que decimos que hacemos?) Suele ser un cierre de monitoreo por parte de un supervisor o gerente, de la misma auditoría interna.

Ver gráficos necesarios de las definiciones de recién ???

#### Informe de Estado

**Informe de estado:** Se ocupa de mantener los registros de la evolución del sistema, maneja mucha información y salidas por lo que se suele implementar dentro de procesos automáticos. Incluye reportes de rastreabilidad de todos los cambios realizados a las líneas base durante el ciclo de vida

Estos informes o reportes, permiten responder cuál es el estado de un ítem, si un requerimiento de cambio fue rechazado o no por el CCB (QUE ES CCB???) Que version de item implementa un requerimiento de cambio aprobado o bien la diferencia entre dos versiones.

#### Plan de Gestión

**Un plan de gestión de configuración** incluye cosas como reglas de nombrado, herramientas a utilizar en el SCM, roles e integrantes del comité, plantillas de formularios, procesos de auditoría, etc.

### Evolucion de Gestión de Configuración de Software

En la **evolución de la gestión de configuración de software tenemos la integración entrega y despliegue continuo**

Ver gráfico pero por las dudas aca el transcript:

**Construcción --> Pruebas Unitarias --> Despliegue --> Pruebas de Aceptación --> Despliegue a producción**

*Donde integracion llega a pruebas de aceptación y entrega tiene una flecha manual al despliegue a producción*

En ambientes Ágiles siempre debemos recordar lo siguiente:

Los 4 valores del manifiesto ágil son **INTERACCIÓN Y PERSONAS** sobre procesos, **PRODUCTO ENTREGADO Y FUNCIONAL** por sobre documentación comprensiva, **PERSONAS INTERRELACIONADAS (COLABORACIÓN CON EL CLIENTE)** sobre contratos y **RESPONDER AL CAMBIO** sobre seguir al plan.

En Agile SCM Sirve a los practicantes y no al revés. Coordina el desarrollo, no lo maneja, responde a cambios, se esfuerza en ser transparente, elimina el desperdicio, es responsabilidad de todo el equipo y siempre se busca automatizar todo lo que se pueda. (Como vemos en el gráfico de arriba)

Qué pasa con el comité de control de cambios ???

Que items podemos tener ???

Qué pasa con las auditorías ???

Qué pasa con reportes de estado ???

http://www.scmpatterns.com/agilescm/

Ver con libro ELEMENTS OF SCM

**Roles:**

* Usuario
* Autor
* Administrador

**User story:**

Usuario

Registrar usuario

Cambiar membresia

Recibir confirmacion cambio de membresia

Comprar librio (credito)

Comprar librio (precio)

Descargar audiolibro

Escuchar audioñibro verr

periodo de prueba???

Iniciar periodo de prueba

---

Buscar libros en catálogo

Comprar libro

Añadir a la lista de deseos

Probar audiolibro

Devolver libro

Acreditar reembolso en crédito

Acreditar reembolso con tarjeta

Comprar créditos adicionales

Pagar automáticamente la membresía(pagar membresia)

Notificar tarjeta de crédito no vigente

Suspender membresía

Cancelar membresía automático(por 2 razones )

Descargar audiolibro

Ver biblioteca

Reproducir audiolibro

Cancelar membresía (con mail)

Actor

Recibir pago por libro

Gestionar membresía

Administrador

Registrar libro

Dar de baja libro libro

Agregar narración

**No se incluye:**

x Añadir a la lista de deseos

x Probar audiolibro

x Devolver libro

x Acreditar reembolso en crédito

x Acreditar reembolso con tarjeta

Debitar automáticamente la membresía(pagar membresia)

x Suspender membresía

x Iniciar periodo de prueba