# 

# *Preguntas auto-evaluación Unidades 1 y 2*

## *1. Defina Ingeniería del Software. Utilice menos de 100 palabras.*

**La ingeniera del software es la aplicación de un enfoque sistemático. Disciplinado y cuantificable para el desarrollo, operación y mantenimiento del software; es decir, la aplicación de la ingeniería al software (Libro Pressman pag. 11 por IEEE).**

La ingeniería de software es el establecimiento y uso de principios fundamentales de la ingeniería con objeto de desarrollar en forma económica software que sea confiable y que trabaje con eficiencia en máquinas reales.

(Libro Pressman pag. 11 por Fritz Bauer).

*2. ¿Cuál es la principal característica que identificó a la Ingeniería de Software en la década del 50?*

- **Hacer software cómo hacer hardware.**

- La ingeniería de software es como la ingeniería de hardware (Tesis 1950 libro Barry Boehm).

- Basa todo el desarrollo en base al hardware.

DÉCADA 50

Basa todo el desarrollo en base al hardware.

Se basa en la época de la guerra fría, por lo tanto tenían un enfoque militar.

Hacer software cómo hacer hardware.

Todos eran de otras areas.

Medir 2 veces y cortar 1.

**DÉCADA 60**

Se empiezan a dar cuenta de que no es hardware entonces empiezan a programar y corregir

El código se corregía y se vuelven inmanejables

y se empiezan a hacer convenciones sobre el tema

Los programadores seguían su camino

El mantenimiento del hardware no es el mismo que el del software, el software envejece a otra velocidad que el del hardware

Las estimaciones y los costos son distintas en el software a las del hardware

**DECADA 70**

Los programas siempre se podían construir sin instrucciones de GOTO gracias al Movimiento Estructurado

El paradigma de guiado por Requerimientos adquiere mejor forma “Cascada”

El modelo de Cascada adquiere un enfoque un poco más iterativo

Se añade el concepto de “iteraciones confinada ” para completar la etapas, y el prototipado“build it twice” antes de desplegar a producción.

El conjunto de versiones remarca la verificación y la validación de los artefactos en cada etapa antes de ir a la siguiente en orden de detectar y corregir errores y defectos, sin salir de la etapa siempre que sea posible

Problema

el modelo de cascada fue interpretado más frecuentemente como un modelo puramente secuencial, en el cual una etapa no terminaba hasta que termine la siguiente

**DÉCADA 80**

Enfocada a la productividad y escalabilidad

se descubre que la prueba se hace más barata cuando se la hace temprano

Escalabilidad

se buscaba adaptar a las tecnologias de esa época

Muchas herramientas CASE, para facilitar el trabajo

Aparece la programación orientada a objeto

Empieza la reutilización de Software

**DÉCADA 90**

La POO toma gran fuerza, surgen UML y RUP. La captura de requisitos se da de manera concurrente. Se le presta gran importancia a llegar rápido al mercado (aparece el desarrollo rápido de aplicaciones) y a la reutilización de código (aparece el desarrollo basado en componentes). Surge el desarrollo open source. Se comienza a hacer software centrado en el usuario (dando gran importancia a la usabilidad y a las interfaces amigables).

**DÉCADA 2000**

Surgen las metodologías ágiles y la ingeniería basada en el valor (realizar aquellas cosas que le aportan mayor valor al cliente). Salir rápidamente al mercado cobra aún mayor importancia. El software comienza a transformarse en algo crítico de lo cual se depende. Cobran mayor importancia los COTS (software enlatado), el desarrollo conducido por modelos y la integración de sistemas.

**DÉCADA 2010**

Se da el gran auge de internet y los dispositivos móviles, dando lugar a la globalización y los sistemas de sistemas.

*3. Explique con sus palabras lo que se entiende por crisis del software. Utilice menos de 100 palabras.*

**La crisis del sw es un problema en donde no se sabía decir o cumplir con exactitud los requerimientos que se daban.** Programar y corregir el código, hasta que este se volvía inmanejable, y estimar los costos del mismo de manera errónea.

El término expresaba las dificultades del desarrollo de software frente al rápido crecimiento de la demanda por software, de la complejidad de los problemas a ser resueltos y de la inexistencia de técnicas establecidas para el desarrollo de sistemas que funcionaran adecuadamente o pudieran ser validados.

Algunos "síntomas" que indican que el software se encuentra en un periodo de crisis son:

• Baja Calidad del Software.

• Tiempo y Presupuesto Excedido.

• Confiabilidad Cuestionable.

• Altos Requerimientos de Personal para desarrollo y mantenimiento.

*4. ¿Qué es un proceso? Utilice menos de 50 palabras.*

**Conjunto de actividades, acciones y tareas que se ejecutan cuando va a crearse algún producto del trabajo. (Libro Pressman pag. 12).**

**Proceso:**

Un proceso es conjunto de actividades, acciones y tareas que se ejecutan cuando va a crearse algún producto del trabajo.

Una **actividad** busca lograr un objetivo amplio (por ejemplo, comunicación con los participantes) y se desarrolla sin importar el dominio de la aplicación, tamaño del proyecto, complejidad del esfuerzo o grado de rigor con el que se usará la ingeniería de software.

Una **acción** (diseño de la arquitectura) es un conjunto de tareas que producen un producto importante del trabajo (por ejemplo, un modelo del diseño de la arquitectura)

Una **tarea** se centra en un objetivo pequeño pero bien definido(por ejemplo, realizar una prueba unitaria) qué produce un resultado tangible.

En el contexto de la ingeniería de software, un proceso no es una prescripción rígida de cómo elaborar software de cómputo. Por el contrario, es un enfoque adaptable que permite que las personas que hacen el trabajo (el equipo de software) busquen y elijan el conjunto apropiado de acciones y tareas para el trabajo. Se busca siempre entregar el software en forma oportuna y con calidad suficiente para satisfacer a quienes patrocinaron su creación y a aquellos que lo usarán.

*5. Explique cómo la elección del ciclo de vida influye en la gestión de proyectos.*

La elección del ciclo de vida elegido (proceso de desarrollo del software establecido) influye directamente en la gestión del proyectos, ya que este define, quién hará cada actividad o tarea, cuando lo hará (asignándole un orden o secuencia) y cómo hará la actividad (que metodología y herramientas utilizara). La gestión de proyecto se hará teniendo en cuenta como se definió el proceso de desarrollo, respetando/corrigiendo lo definido.

El ciclo de vida influye en la manera en que hago las cosas, en la manera en que me voy a comportar en la gestión de proyectos, cómo se debe actuar.

Si influye, cada metodología indica un conjunto de cosas que se pueden y que no se pueden hacer.

Si estamos en un ciclo de vida por cascadas,no vamos a pasar de una etapa (planeación por ejemplo) a la siguiente hasta que hayamos terminado dicha etapa

Si estamos en un enfoque incremental,en la primera iteración vamos presentar determinadas funcionalidades y en las iteraciones siguientes vamos a ir añadiendo otras.

Si estamos en un enfoque iterativo, desde la primer iteración vamos a tener todas las funcionalidades lo que vamos a ir haciendo en cada iteración es ir mejorando y terminando dichas funcionalidades.

*6. Explique la diferencia entre procesos de control predictivos y empíricos. Utilice menos de 100 palabras.*

(Libro Pressman pag. 30 al 33).

Los procesos de control predictivos definen todas las variables y predicen su comportamiento al inicio de un proyecto (buscando reducir los riesgos). Primero se define el análisis de riesgos, luego va el diseño y por último el desarrollo (MÁS ESTRUCTURADO).

El proceso de control empírico tiene en claro la complejidad que existe, por lo cual mide constantemente los resultados y adapta según este, su comportamiento. (MENOS ESTRUCTURADO).

Predictivos: quiere saber de antemano cómo va a ser todo respecto a las variables de proyecto con inf detallada.

La principal diferencia de los procesos de control predictivo y empírico es la siguiente:

Los **controles predictivos:** asumen que es posible predecir y detallar a largo plazo las variables del proyecto,en este enfoque se busca solucionar la complejidad de un proyecto esforzándose en una planificación detallada.

En cambio los **controles empíricos:** asumen que hay un horizonte de predicción de las variables del proyecto dado que siempre hay cambios en el contexto del proyecto debido a la indeterminación y complejidad propios. Se basa en la **inspección y adaptación regular** en función de los resultados que se van obteniendo y del propio contexto del proyecto

*7. Describa las principales características del ciclo de vida en cascada. Utilice menos de 100 palabras***.**

* ciclo orientado a documentación, sé que una etapa terminó por los documentos
* secuencial
* especializado
* roles definidos
* equipos grandes
* altamente burocráctico

Principales Características

1. **Orientado a la Documentación**
2. **Se basa en dividir un proyecto en fases:** Es decir que el proyecto se encuentra formada por etapas o fases, por ejemplo, de manera simplificada, requisitos, diseño y construcción, y que cada una de estas fases no comience hasta que termine con éxito la anterior
3. **Se le llama predictibilidad porque cada fase intenta predecir lo que pasará en la siguiente.**
4. **Cada una de estas fases realiza una única vez por lo tanto se trataba de ser lo más consistente posible** en cada una de estas fases para tratar de evitar errores en futuras etapas del desarrollo.
5. **Fases están claramente diferenciadas,** esto quiere decir que por ejemplo que el personal de una etapa no va a ser el mismo que en la etapa siguiente del desarrollo ya que lo que se buscaba era tener profesionales claramente diferenciados y especializados en cada una de ellas.
6. **Cada fase concluye con un entregable documental que sirve de entrada a la siguiente fase**, porque lo que buscaba la documentación era tratar de reducir los gaps semánticos existentes entre las distintas etapas y como lineamientos para saber qué hacer y cómo hacerlo.

*8. ¿Cuál es el beneficio de implementar un ciclo de vida ágil con respecto utilizar un ciclo de vida en cascada? Describa también riesgos y costes.*

**Beneficios**

1. Las entregas de software funcional se reducen a unas cuantas semanas, contral el modelo en cascada que demoraba años
2. Es posible saber a las pocas semanas de iniciado el proyecto, si estamos haciendo lo que el cliente quiere o no
3. El equipo de Desarrollo no pierde tiempo realizando documentación que nadie va a utilizar
4. Con cada iteración podemos estar más seguro de cuánto tiempo nos va a llevar el proyecto, y cuánto nos falta para terminarlo

**Riesgos**

1. **Los riesgos de hacer algo que no es lo que el cliente quiere se ve reducido** enormemente ya que ahora al entregar valor en etapas temprana del proyecto se puede detectar si el equipo está bien encaminado y en caso de no estarlo, se puede corregir el rumbo el proyecto; mientras que en un ciclo en cascada el mismo riesgo es mucho más grande, ya que recién se va a determinar si el producto es lo que el cliente quiere cuando estemos en la etapa de implementación, y un cambio a esta altura del proyecto es muy difícil de realizar
2. **Los riesgos de que el cliente pierda interés en el proyecto y lo termine cancelando también se ven reducidos** por la misma justificación que en el anterior, el cliente recibe desde etapas temprana del proyecto, software funcional lo que hace que se sienta parte del proyecto mientras que en los modelos tradicionales el cliente solo participaba en los requerimientos y al final recibía un software el problema es que pasaba mucho tiempo entre ambos sucesos por lo que el cliente podía dar de baja el proyecto
3. **Los costos de tener que realizar un cambio en algún aspecto del software son mucho más baratos** de realizar que en el modelo en cascada; ya que en el ciclo de vida ágil lo que tiene un alto nivel de detalle es lo que se está desarrollando en este momento y todo lo que no se desarrolló hasta el momento su nivel de detalle son solo lineamientos generales, por lo tantos los costos relacionados con el cambio de los mismos es bajo. Mientras que en el modelo en cascada los costos tienen una forma exponencial, es decir cuánto más avanzados estemos en el proyecto más costoso sería el cambio.

*9. Explique qué entiende por Gestión de Proyectos de Software y mencione los pilares a tener en cuenta para que la gestión de proyectos sea eficaz*

“Gestión de Proyectos de Software” se entiende como la planificación, supervisión y control del personal, el proceso y los eventos que ocurren mientras el software evoluciona desde una idea preliminar hasta una implementación operativa.

Los pilares que se tiene en cuenta en la gestión de un proyecto para que este sea éxito y eficaz son las 4 P: PERSONAL, PRODUCTO, PROCESO y PROYECTO, en este orden definido. (No debemos olvidar que el trabajo de ingeniería está hecho enteramente por personas).

*10. ¿Cuál es el pricipal inconveniente con la gestión de proyectos?*

El principal inconveniente con la gestión de proyectos es que se quiere controlar las 4 P (PERSONAL, PRODUCTO, PROCESO y PROYECTO) y no se pueden controlar. Se quiere saber de antemano todo lo que se debe hacer y quién lo debe hacer, cosas que se quieren saber de antemano y no se pueden controlar.

Principales Inconvenientes Gestión de Proyectos:

1. El personal del software no entiende las necesidades del cliente.
2. El ámbito del producto está pobremente definido.
3. Los cambios se gestionan pobremente.
4. Cambia la tecnología elegida.
5. Las necesidades empresariales cambian [o están mal definidas].
6. Las fechas límite son irreales.
7. Los usuarios son resistentes.
8. Pérdida de patrocinio [o nunca obtenido adecuadamente].
9. El equipo del proyecto carece de personal con habilidades adecuadas.
10. Los gerentes [y profesionales] evitan mejores prácticas y lecciones aprendidas.

[Pressman 566]

*11. Mencione los organigramas de equipos propuestos en la Ingeniería del Software clásica y explique uno de ellos*

Mantei sugiere 3 organigramas de equipo genéricos:

1. **Descentralizado democrático (DD):** No existe un jefe permanente sino coordinadores de tareas a corto plazo. Las decisiones se hacen por consenso del grupo. La comunicación entre los miembros es horizontal. Características:

* Recomendado en problemas difíciles
* Producen moral más alta y satisfacción por el trabajo. Es para equipos con tiempos de vida largo
* Problemas de modularidad baja porque implica gran cantidad de comunicación

1. **Descentralizado controlado (DC):** Se define un jefe (líder de grupo permanente) que coordina tareas específicas y jefes secundarios para subtareas. La resolución de problemas sigue siendo por consenso grupal pero la implementación de soluciones se reparte entre subgrupos por el jefe de equipo, es decir, la implementación de subtareas es asignada por el líder. La comunicación entre subgrupos e individuos es horizontal, y a lo largo de la jerarquía de control, es vertical. Características:

* Recomendado en problemas complejos fácilmente modularizables en problemas sencillos
* ↑ Cantidad de comunicación = ↓Rendimiento → Proyectos grandes con formación de subgrupos

1. **Centralizado controlado (CC):** El jefe de equipo se encarga de la resolución de problemas a alto nivel y la coordinación interna del equipo. La comunicación entre el jefe y los miembros del equipo es vertical.

Para problemas sencillos, una estructura centralizada para una resolución de problemas más rápida es la más adecuada, por lo tanto lo son los equipos DC y CC. Para problemas más complejos DD es mejor, ya que la descentralización genera más y mejores soluciones.

Para proyectos más grandes son mejores los equipos CC o DC ya que es menor la cantidad de comunicación necesaria. Para cuando los miembros deben permanecer juntos durante mucho tiempo una estructura DD produce una moral más alta y más satisfacción. Los equipos DD requieren generalmente más tiempo para completar un proyecto.

*12. ¿Cuáles son las características de un equipo ágil? Explique una de ellas*

Las características de un equipo ágil son:

* Multifuncional
* Auto – Organizado
* Pequeño

*Multifuncional*

No significa que todos deben saber hacer todo, pero debe haber al menos 1 experto y 1 que tiene conocimientos sobre cada tema. Implica responsabilidad y adaptación.

Un equipo multifuncional (puedes ampliar información en los equipos modernos de desarrollo son multifuncionales, así disparan la velocidad y la productividad) es aquel equipo que posee todas las competencias necesarias para lograr completar el trabajo, sin depender (o dependiendo mínimamente) de otros equipos, áreas, o roles fuera del mismo.

No significa que todo el mundo dentro del equipo sepa hacer de todo, significa que el equipo en su conjunto es autosuficiente y tiene el conocimiento y habilidades necesarias para construir la parte del producto que le toca y que la especialidad de cada miembro puede ser complementada por algún otro miembro del equipo.

*Auto-Organizado*

El equipo en conjunto decide quién hace, qué hace y cómo lo hace.

1. **Autónomo:** No existe un único centro de toma de decisiones o de autoridad. El control se distribuye conjuntamente por todo el equipo.
2. **Adaptable:** El equipo se ajusta dinámicamente según sea necesario, con el fin de resolver sus propios problemas y mejorar su propio desempeño.
3. **Responsable:** El equipo comparte colectivamente la responsabilidad de los resultados.

*Pequeño en número ( personas)*

Los resultados de Putnam evidenciaron que cuando el tamaño del equipo crecía el esfuerzo, lógicamente, al haber más gente, aumentaba, pero el tiempo de proyecto no se reducía. Las razones:

La **coordinación mayor** que requerían esos equipos grandes, las mayores y más numerosas actividades de gestión.

El **incremento de las rutas** de comunicación entre sus miembros, lo cual crea más errores. A más personas, más líneas de comunicación aparecen.

[**https://www.javiergarzas.com/2015/03/tipologias-de-equipos-software-equipo-agil.html**](https://www.javiergarzas.com/2015/03/tipologias-de-equipos-software-equipo-agil.html)

*13. Cite los 4 valores del manifiesto ágil.*

* 1. Individuos e interacciones sobre procesos y herramientas
  2. Software funcionando sobre documentación extensiva
  3. Colaboración con el cliente sobre negociación contractual
  4. Respuesta ante el cambio sobre seguir un plan

*14. Describa 3 principios ágiles*

1. **Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.**

Lo que se busca no es presentar un montón de documentación que no sirve al final del proyecto, sino software funcionando desde el principio para saber si las cosas se están haciendo bien.

1. **Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.**

El principal problema de las metodologías clásicas era que al cambiar un requisito en una etapa tardía del proyecto, el costo de afrontar dicho cambio era altísimo; en estas nuevas prácticas se prevé que las cosas van cambiar y está organizado de tal manera que un cambio en una etapa temprana del proyecto, tenga el mismo impacto que una vez avanzado el proyecto.

1. **La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.**

Si no tiene calidad, es muy probable que se tenga que volver a hacer más adelante, y el rehacer consume tiempo, esfuerzo y dinero, por lo tanto lo que se busca desde el principio es no tener que rehacer las cosas, por el sentido técnico

*15. ¿Qué es un proyecto ágil?*

Ambler, define a un proyecto ágil como aquella manera de enfocar el desarrollo de software mediante un ciclo iterativo (en extremo) e incremental, con equipos que trabajan de manera altamente colaborativa y auto-organizados. Es decir, un proyecto ágil es un desarrollo iterativo más la segunda gran característica implicada directamente por la iteración extrema: equipos multifuncionales y auto-organizados.

*16. ¿Qué significa que el Equipo sea multifuncional?*

Dentro de las personas y proyectos, se tienen todas las capacidades que se necesitan para llevarlo a cabo. Se debe tener por tema por lo menos una persona experta y otra que tenga los conocimientos suficientes de todos los temas. Implica responsabilidad y adaptación.

Un equipo multifuncional es aquel equipo que posee todas las competencias necesarias para lograr completar el trabajo, sin depender (o dependiendo mínimamente) de otros equipos, áreas, o roles fuera del mismo.

No significa que todo el mundo dentro del equipo sepa hacer de todo, significa que el equipo en su conjunto es autosuficiente y tiene el conocimiento y habilidades necesarias para construir la parte del producto que le toca y que la especialidad de cada miembro puede ser complementada por algún otro miembro del equipo.

*17. Describa las principales diferencias en el equipo si utilizamos un ciclo de vida en cascada y uno ágil.*

|  |  |  |
| --- | --- | --- |
|  | **Ciclo en Cascado** | **Ciclo Ágil** |
| **Nivel de Especialización sobre un área** | Cada miembro del equipo se encuentra altamente especializado sobre un aspecto particular del desarrollo y tiende a carecer de conocimientos sobre otras áreas | Cada miembro del equipo posee conocimiento específico sobre un aspecto del desarrollo, pero también posee conocimientos para comprender y colaborar con profesionales de otras áreas |
| **Conocimiento del Estado del Proyecto** | Solo van a conocer el estado del proyecto desde el punto de vista del área en que se encuentran desconociendo el estado real del proyecto; por ejemplo si un empleado es del área de requerimientos, y ya se terminó la etapa de análisis él va a saber que el análisis está completo pero no va a conocer cómo sigue el proyecto | Como todo el proyecto es llevado a cabo por un proyecto multifuncional cada miembro del grupo va a conocer el estado general del proyecto; por ejemplo un empleado va a saber que se implementaron un determinado porcentaje de las funcionalidades y que porcentaje se está desarrollando |
| **Peso en la toma de Decisiones relacionadas con el proyecto** | Sus opiniones y decisiones no son tenidas en cuenta en el proyecto ya que son solo miembros operativos de la organización | Se considera que todos tienen la misma jerarquía y que su opinión es importante para tomar decisiones |

Cascada

- Son de gran tamaño

- No son multifuncionales. Cada etapa tiene definido la cantidad de personas.

- No son auto – organizados. Las decisiones las toma otra persona.

Agiles

- El cliente forma parte del equipo

- El equipo toma las decisiones (auto-organizados)

*18. ¿Qué significa que el equipo sea Autónomo, Adaptable y Responsable?*

Significa que el equipo es Auto-Organizado. Característica de las más mencionadas al hablar de agilidad, así como de las más malinterpretadas y peligrosas cuando mal.

En síntesis, un equipo auto-organizado tiene tres características principales:

**Autónomo:** No existe un único centro de toma de decisiones o de autoridad. El control se distribuye conjuntamente por todo el equipo. La toma de decisones no está.centralizada

**Adaptable:** El equipo se ajusta dinámicamente según sea necesario, con el fin de resolver sus propios problemas y mejorar su propio desempeño. Me voy adptando al cambio

**Responsable:** El equipo comparte colectivamente la responsabilidad de los resultados.

*19. ¿Cómo deben ser las historias de usuario?*

Deben ser como el método INVEST. El método INVEST descrito por Bill Wake nos ayuda a escribir historias de usuario, consiste en cumplir las siguientes características:

* **Independiente**: Cada historia de usuario pueda ser planificada e implementada en cualquier orden. No dependen unas de otras, si ocurre se deben dividir o combinar.
* **Negociable**: Las historias deben ser negociables ya que sus detalles serán acordados por el cliente/usuario y el equipo durante la fase de «conversación».
* **Valor**: Una historia de usuario tiene que ser valiosa para el cliente o el usuario.
* **Estimable**: Una buena historia de usuario debe ser estimada con la precisión suficiente para ayudar al cliente o usuario a priorizar y planificar su implementación. Si no podemos estimarla debemos indicir en la fase de conversación o dividirla.
* **Pequeña**: Atómicas. Solemos hacerlas de tal modo que ocupen como máximo un sprint.
* **Testeable**: La historia de usuario debe poderse probar (Hemos trabajado con anterioridad en la fase “confirmación” de la historia de usuario). Tanto el usuario como el equipo de desarrollo tienen que poder probarla para saber cuándo está finalizada.
* **Cerradas**
* **Periodos de tiempo pequeños**, de tal forma de que se realicen en 2 o 3 días de desarrollo
* Deben recordar **QUÉ** hay que hacer, no CÓMO hay que hacerlo

*20. ¿Qué es un punto de historia?*

Los **Puntos Historia** son una medida relativa que indica la complejidad de un ítem, típicamente, una Historia de Usuario, frente a otra. Se utilizan para establecer el “tamaño” de una historia de usuario en función de complejidad – riesgo.

*21. Explique la Técnica Planning Poker*

La técnica más conocida para definir la estimación de puntos de historia es el Planning Poker. En ella, todos los participantes del equipo deben trabajar juntos en todas las estimaciones.

1. Cada persona toma un mazo de cartas numeradas con la secuencia vista antes.
2. Se presentan las historias de usuario (o tareas) a estimar, haciendo una descripción de las mismas.

Cada integrante del equipo muestra la carta seleccionada (evitar influencias). Se llega por consenso a la estimación de las historias de usuario.



La mecánica es que el cliente (en entornos Scrum sería el Product Owner) lee una historia, se aclara y cada miembro del equipo técnico escribe su estimación en una tarjeta. Una vez que todos han escrito su estimación, las tarjetas se ponen sobre la mesa. Si hay acuerdo, no es necesario debatir. Si no lo hay, el equipo discute para intentar llegar a un consenso Con todo ello, la idea es que todos miembros del equipo son jugadores en el juego y no solo algunos o los de siempre. Y, además, el equipo tarda muy poco en estimar las historias más familiares o sencillas, llegando a un consenso sin discusión.

*22. ¿QUÉ ES LA VELOCIDAD Y POR QUÉ ES LA CLAVE DE LA ESTIMACIÓN ÁGIL?*

La **velocidad es la manera de medir la productividad de un equipo en un sprint.** Se calcula sumando el número de puntos historia (según la unidad y escala que uses) que se estimaron para cada historia de usuario que se ha terminado durante una iteración o sprint.

Es **clave para la estimación** porque nos va a indicar cuantos sprints van a ser necesario para terminar con todas las historias de usuario qué se encuentran en el product backlog

Cantidad de trabajos realizados en un sprint. Mide el *trabajo realizado* en función de la *estimación inicial*. La velocidad se calcula **sumando** el número de **puntos de historia** de cada **historia de usuario terminada** durante la iteración. Ej: si el equipo completó 3 historias, de 5 puntos cada una, entonces su velocidad sería 15. Si el equipo completó 2 historias de 5 puntos, su velocidad sería 10. Si **sumamos los puntos de historia** de todas las historias de usuario tendremos una estimación del **tamaño total del proyecto**. Si conocemos la **velocidad del equipo**, podremos dividir el tamaño entre la velocidad para llegar a una **estimación del número de iteraciones necesarias**.



*23. ¿Para qué sirve el método INVEST?*

El **método INVEST sirve para asegurar la calidad de las historias de usuario.** Son un conjunto de características que sirven para verificar si una historia de usuario está bien escrita ya que se parte del supuesto de que para que una historia de usuario sea correcta debe cumplir con cada una de esas características:

**Independiente**: las historias pueden completarse en cualquier orden.

**Negociable:** los detalles de la historia son co-creados por los programadores y los clientes durante el desarrollo.

**Valiosa:** la funcionalidad es valiosa para los clientes o los usuarios del software.

**Estimable:** los programadores pueden encontrar una estimación razonable para construir la historia.

**Pequeña:** las historias deberían construirse en poco tiempo, generalmente alrededor de "días/persona". Se tienen que poder construir muchas historias en una iteración.

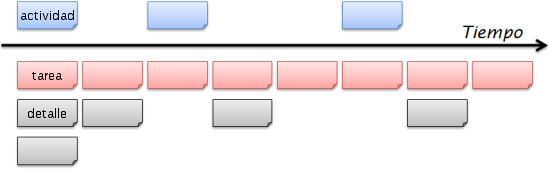
**Testeable:** se debe poder escribir pruebas que verifiquen que el software de la historia funcione adecuadamente.

*24. ¿Qué es un Mapa de Historias de Usuario?*

Los mapas de historia de usuario son una forma de organizar e implementar las historias de usuario. Es una aproximación a la organización y a la priorización de las historias de usuario; se hacen para comprender el producto que se está construyendo, ordenando las historias en el tiempo. Sirven para comprender hacia dónde se irá, para ver la vida del producto, y para determinar cosas que, en principio, parecen prioritarias pero dependen de otras que son menos prioritarias.

Cómo se organiza un mapa de historias de usuario

Les presento el **mapa de historias de usuario**: una versión mejorada del backlog del producto. Los mapas de historias de usuario nos permiten visualizar todo el sistema: representan el producto como un "todo", en vez de quedar miopes observando historias individuales.



Arriba de todo (azul) se ubican las "grandes historias",

Estos elementos son las tareas de usuario. Una tarea de usuario es lo que hacen los usuarios para lograr un objetivo.

Así, se organizan estas "pequeñas historias" debajo de las "grandes historias" para ir armando una grilla

Al ordenar las historias en el mapa, si la persona que usa el sistema en general hace una cosa antes de otra, ubicamos la primera historia a la izquierda, y la otra a su derecha.

Las historias superiores son justamente la columna vertebral del sistema, el motivo de ser y de donde se desprende toda la funcionalidad. Las historias inferiores son el esqueleto, unido por estas historias superiores

*25. ¿Qué propone la técnica MoSCoW?*

**Es un método que propone priorizar las historias de usuario en función del valor aportado al cliente.**

**Técnica MoSCoW**

El método MoSCoW es una técnica de priorización de requisitos basada en el hecho de que, aunque todos los requisitos se consideren importantes, es fundamental destacar aquellos que permiten darle un mayor valor al sistema, lo que permite enfocar los trabajos de manera más eficiente.

Lo que la diferencia de otras técnicas tradicionales como por ejemplo calificar los requisitos como de prioridad alta, media o baja es que en este caso la escala utilizada tiene un significado intrínseco, de manera que el usuario responsable de asignar la prioridad conoce el efecto real que producirá su elección.

**M (Must)**: Requisito que tiene que estar implementado en la versión final del producto para que la misma pueda ser considerada un éxito.

**S (Should):** Requisito de alta prioridad que en la medida de lo posible debería ser incluido en la solución final, pero que llegado el momento y si fuera necesario, podría ser prescindible si hubiera alguna causa que lo justificara.

**C (Could):** Requisito deseable pero no necesario, se implementaría si hubiera posibilidades presupuestarias y temporales.

**W (Won’t):** Hace referencia a requisitos que están descartados de momento pero que en un futuro podrían ser tenidos de nuevo en cuenta y ser reclasificados en una de las categorías anteriores.

Esta clasificación puede ser modificada durante el proceso de desarrollo y definirse, en el caso de desarrollos iterativos incrementales, prioridades a nivel de iteración.

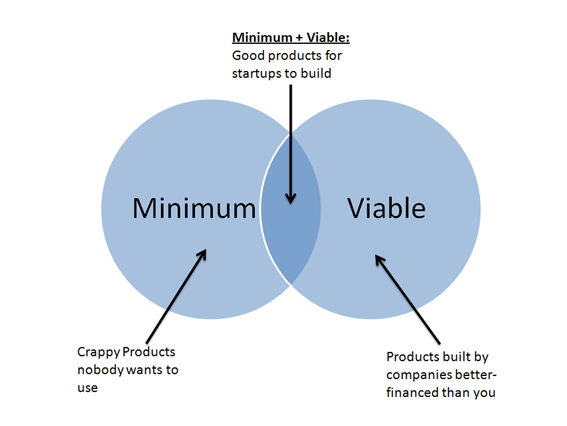
*26. ¿Qué es el MVP? Explique en menos de 80 palabras.*

Este acrónimo viene de las palabras en inglés**: “Minimum viable product”,** es decir, aquel producto mínimo viable que te permita validar por lo menos una parte de tu negocio. ***La mínima cantidad de funcionalidad que el cliente está dispuesto a pagar. Suele ser único y evoluciona a lo largo del tiempo, pero dependiendo del tipo de producto, se pueden tener más.*** Este será el primer prototipo que llevarás al mercado para encontrar las primeras reacciones de tus clientes.

Esta fase es muy importante por 2 razones:

– **Te dará las primeras pistas** para pivotar el modelo o perseverar en él (a menos que seas un genio tocará pivotar)

– **Entrarás en un escenario “invertible”**: con esto me refiero a que hoy día, el 98% de los inversores te van a pedir que hayas pasado por esta fase. Sino no podrás mostrar números y métricas de tu negocio, y eso es muy importante para el inversor.



*27. Explique la principal diferencia entre release y prototipo*

La principal diferencia es que en la **release**, se le entrega algo de valor a un cliente, es decir se pone en producción, es usado por el usuario. Mientras que un **prototipo**, en este contexto, representa un modelo que se le presenta al cliente para que lo apruebe o no, así como también para qué brinde sus sugerencias, y en base a esa información terminar el prototipo. El prototipo no está implementado, tiene como propósito que el Product Owner valide la funcionalidad desarrollada.

El prototipo es algo que no está en producción, mientras que la reléase sí está en producción.

Un prototipo puede ser una pantalla de un login(solo la parte gráfica). Mientras que una release va a ser todo el sistema de login funcionando

*28. ¿Qué es la velocidad del equipo?*

La **velocidad es la manera de medir la productividad de un equipo en un sprint.** Se calcula sumando el número de puntos historia (según la unidad y escala que uses) que se estimaron para cada historia de usuario que se ha terminado durante una iteración o sprint.

*29. Aplicando técnicas agiles, ¿se planifica más que utilizando técnicas tradicionales? Justifique su respuesta en no más de 150 palabras*

Si, porque al aplicar técnicas ágiles se planifica constantemente en cada sprint.

Se planifica mucho más que en las metodologías tradicionales, en las metodologías tradicionales, se planificaba mucho al principio, pero una vez que se planificó era muy poco probable de que se vuelva a planificar, por más que lo que esté sucediendo con el proyecto no coincida con la planificación.

Por otro lado, en las técnicas Ágiles, se planifica a lo largo de todo el proyecto, en cada una de las interacciones llamadas (Sprints) existen varias reuniones enfocadas solamente para planificar

**Sprint Planning:** El trabajo a realizar durante el Sprint se planifica en la Planificación de Sprint. Este plan se crea mediante el trabajo colaborativo del Equipo Scrum completo.

**Scrum Diario:** El Equipo de Desarrollo usa el Scrum Diario para evaluar el progreso hacia el Objetivo del Sprint y para evaluar qué tendencia sigue este progreso hacia la finalización del trabajo contenido en la Lista de Pendientes del Sprint.

**Revisión de Sprint (Sprint Review**): Al final del Sprint se lleva a cabo una Revisión de Sprint para inspeccionar el Incremento y adaptar la Lista de Producto si fuese necesario.

**Retrospectiva de Sprint (Sprint Retrospective)** La Retrospectiva de Sprint es una oportunidad para el Equipo Scrum de inspeccionarse a sí mismo y de crear un plan de mejoras que sean abordadas durante el siguiente Sprint.

Como podemos observar, en las metodologías clásicas, no existía ninguna reunión, documentación u algo parecido cuyo objetivo sea el de planificar cómo mejorar la eficiencia del equipo, mientras que en las Técnicas Ágiles si, y es la Retrospectiva del Sprint.

*30. ¿Qué es estimar?*

Según McConnell “una estimación software es una predicción de cuánto tiempo durará o costará un proyecto”. El propósito de una estimación software es determinar si los objetivos, los tiempos de que disponemos para realizar el proyecto, son suficientemente realistas.

**31. ¿Qué se necesita conocer para realizar una buena estimación?**

En ágiles, se deben conocer 2 variables:

* Tamaño, es decir, la complejidad
* Velocidad del equipo

Conocer el QUÉ, el TAMAÑO (unidad), conocer el TIEMPO REAL E IDEAL

*32. ¿Cuáles son los parámetros que se usan para estimar?*

En ágiles, se deben conocer 2 variables:

* Tamaño, es decir, la complejidad
* Velocidad del equipo

*39. ¿Qué entiende por SCRUM?*

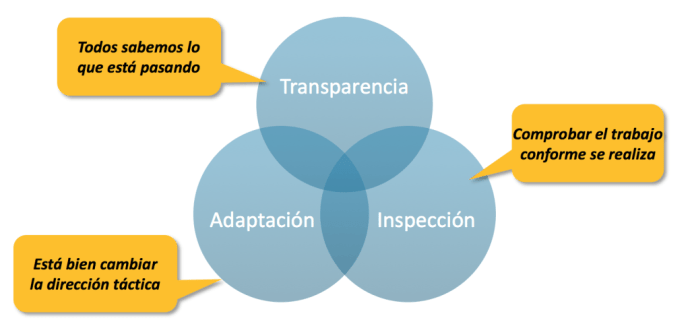
**Scrum**: es un Framework para la gestión completa de proyectos. Un marco de trabajo por el cual las personas pueden abordar problemas complejos adaptativos, a la vez que entregar productos del máximo valor posible productiva y creativamente.

*40. Explique uno de los pilares de SCRUM*

- **Transparencia:** todo el mundo sabe lo que se está haciendo, por medio de los tableros. Implica dar visibilidad a todo lo que está pasando, ya que los aspectos significativos del proceso deben ser visibles para aquellos que son responsables del resultado. La Reunión de Planificación proporciona visibilidad al Equipo Scrum acerca de aquello que va a hacer en el sprint; el Scrum Diario, proporciona visibilidad sobre las tareas diarias, los impedimentos y cómo marcha el trabajo; la Revisión del Sprint ofrece visibilidad sobre los logros, resultados y el progreso. Por último, la Retrospectiva del Sprint contribuye con la inspección y la adaptación del proceso.

- **Inspección:** constantemente estoy evaluando lo que estoy haciendo. Scrum promueve la inspección frecuente de los Artefactos y del progreso para identificar y corregir las variaciones no deseadas. La inspección tiene lugar durante: la Reunión de Planificación de Sprint, el Scrum Diario, la Revisión del Sprint y la Retrospectiva del Sprint.

- **Flexibilidad:** adaptación, me voy adaptando a los cambios.  implica hacer los ajustes en los procesos y artefactos para minimizar la desviación.



*41. ¿Cuáles son las funciones del SCRUM Master?*

El Scrum Master es un líder que está al servicio del Equipo Scrum. Se asegura que se aplique el Scrum adecuadamente.

El Scrum Master se encarga de gestionar y asegurar el proceso Scrum, que éste se lleva a cabo correctamente y de facilitar la ejecución del proceso y sus mecánicas, siempre atendiendo a los tres pilares del control empírico de procesos. Además, se encarga de eliminar impedimentos que puedan afectar a la entrega de producto. También se encarga de hacer coaching al resto del equipo Scrum cuando lo necesitan, además de facilitar reuniones y eventos si es necesario. El Scrum Master puede ser compartido entre varios equipos, aunque su disponibilidad afectara al resultado final del proceso Scrum.

El Scrum Master tiene dos funciones dentro del marco de trabajo. La primera es ***gestionar el proceso Scrum*** y la segunda es ayudar a eliminar impedimentos. Vamos a echarles un vistazo en profundidad:

* Gestionar el proceso Scrum: El Scrum Master es el encargado de mantener Scrum al dia, de proporcinar coaching, mentoring y formacion a la organizacion en caso de que lo necesita y de verlar porque Scrum favorezca la entrega de valor en lugar de ser una herramienta de microgestion.
* Ayudar a eliminar impedimentos: Esta funcion del Scrum Master indica la necesidad de ayudar a eliminar progresiva y constantemente impedimentos que van surgiendo en la organizacion y que afectan tanto a la integridad de Scrum como impiden a la organizacion entregar valor. Hay que remarcar que estos impedimentos no son del equipo, sino de la organizacion, y donde el Scrum Master aporta es a nivel organizativo, no reparando teclados para los miembros del Development Team. El Scrum Master es el responsable de velar porque Scrum se lleve adelante y se pueda facilitar la implementacion.

*42. ¿Qué es el Product Backlog?*

La **Lista de Producto** es una lista ordenada de todo lo que se conoce que es necesario en el producto. Son priorizadas u ordenadas por el valor que aportan al cliente.

Es la única fuente de requisitos para cualquier cambio a realizarse en el producto. El Dueño de Producto (Product Owner) es el responsable de la Lista de Producto, incluyendo su contenido, disponibilidad y ordenación.

El desarrollo más temprano de la misma solo refleja los requisitos conocidos y mejor entendidos al principio. La Lista de Producto evoluciona a medida de que el producto y el entorno en el que se usará también lo hacen. La Lista de Producto es dinámica; cambia constantemente para identificar lo que el producto necesita para ser adecuado, competitivo y útil. Si un producto existe, su Lista de Producto también existe.

*43. ¿Qué es un sprint? Explique brevemente como funciona.*

El corazón de Scrum es el **Sprint**, es un bloque o iteración de tiempo breve (time-box) de un mes o menos (1 a 4 semanas) durante el cual se crea un incremento de producto “Terminado” utilizable y potencialmente desplegable. Se parte de un sprint backlog acotado, del cual el equipo deberá trabajar. Al finalizar el sprint se va a entregar un producto “potencialmente viable”.

Es más conveniente si la duración de los Sprints es consistente a lo largo del esfuerzo de desarrollo. Cada nuevo Sprint comienza inmediatamente después de la finalización del Sprint anterior.

Los Sprints contienen y consisten en la **Planificación del Sprint** (Sprint Planning), los **Scrums Diarios** (Daily Scrums), el **trabajo de desarrollo**, la **Revisión del Sprint** (Sprint Review), y la **Retrospectiva del Sprint** (Sprint Retrospective).

Durante el Sprint:

1. No se realizan cambios que puedan afectar al Objetivo del Sprint (Sprint Goal);
2. Los objetivos de calidad no disminuyen; y,
3. El alcance puede clarificarse y renegociarse entre el Dueño de Producto y el Equipo de Desarrollo a medida que se va aprendiendo más.

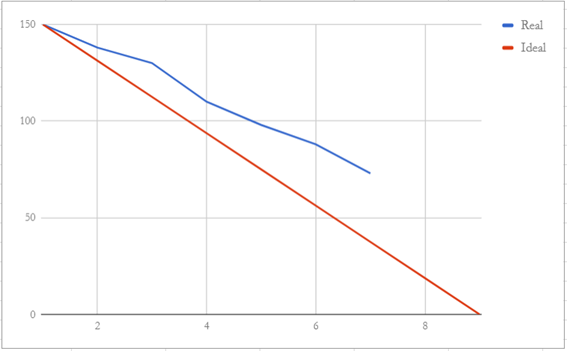
Cada Sprint puede considerarse un **proyecto** con un horizonte no mayor de un mes. Al igual que los proyectos, los Sprints se usan para lograr algo. Cada Sprint **tiene una meta** de lo que se construirá, un diseño y un plan flexible que guiará su construcción, el trabajo del equipo y el **incremento de producto resultante**.

*44. ¿Cómo se mide el avance del proyecto en SCRUM?*

El avance del proyecto en SCRUM se mide con las historias de usuario realizadas o la cantidad de historias pendientes que me van quedando.

Existen varias formas de medir el progreso del trabajo total hasta acabar el proyecto. Al menos una vez por sprint el dueño del producto mide cuánto ha sido trabajo que se ha terminado y cuánto queda para acabar el proyecto y lo anota.

Esta forma de contar el trabajo restante para acabar el proyecto se le llama Burndown y si se pinta en una gráfica será la llamada gráfica de Burndown. Se le llama así porque se va anotando la cantidad de trabajo que queda pendiente en lugar de la cantidad de trabajo o puntos que vamos sumando al realizar más trabajo. El trabajo va disminuyendo o quemándose hasta llegar a 0 horas o puntos de trabajo pendiente, momento en el cuál toca suelo en el eje horizontal.



*45. ¿Por qué decimos que uno de los beneficios de SCRUM es que se mejoran las estimaciones?*

Porque estamos estimando constantemente. Además, las técnicas de estimación que utilizamos permiten llegar a consensos más rápidamente, porque no nos centramos en estimar teniendo en cuenta una hora, sino que nos centramos a estimar comparando complejidades, es decir, usamos medidas relativas.

Decimos que las estimaciones mejoran cuando implementamos scrum porque al final de cada sprint tenemos lo que se llama lo que se llama como “Retrospectiva del Sprint” en la cual el equipo debate sobre cuáles fueron sus puntos débiles y los fuertes para definir un plan de mejora para el siguiente sprint; por lo tanto si en un 1 sprint definieron que iba a poder desarrollar el 10 historias de usuario pero, no lo pudieron alcanzar ese objetivo, en el próximo sprint se van a plantear desarrollar 7 y si en ese sprint alcanzaron las 7 y les sobro tiempo, para el siguiente se van a plantear 8 y así hasta que determinen cuál es su velocidad, y este proceso duro como máximo 3 meses y dio como resultado la estimación de cuanto software son capaces de diseñar en un determinado periodo de tiempo, cosa que no pasaba en las metodologías anteriores que por lo general se planteaban un determinado periodo de tiempo para realizar el software y la gran mayoría de las veces eran incapaces de presentar el software a tiempo, o con todas las funcionalidades, o con los costos planteados en un principio.

*46. Describa las principales funciones del product owner.*

* Es el representante del cliente
* Responsable de definir las historias de usuario con calidad de manera correcta
* Tiene que estar disponible todo el tiempo para el cliente durante el desarrollo, o bien, cuando éste lo necesite.
* Priorizar las historias de usuario
* Define los MVP (Producto Mínimo Viable)

1) Escribir buenas historias de usuario

2) Decidir qué construir… ¡y qué no!

3) Fijar criterios de aceptación para cada historia de usuario.

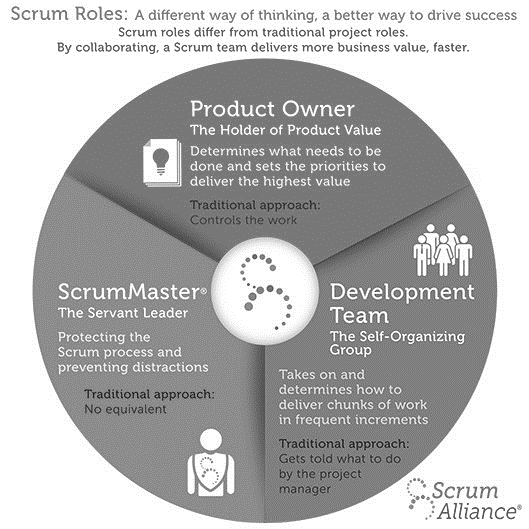
4) Definir “productos mínimos viables”.

5) Priorizar historias de usuario, y para ello tener claro el valor de las mismas y el valor que necesita el producto en cada momento (que diferirá a lo largo de su vida).

6) Estar accesible, y disponible, para explicar al equipo técnico dudas funcionales, para validar entregas y participar en reuniones.

7) Definir el plan de releases (o la visión estratégica).

*47. Explique, por medio de un ejemplo, qué pasaría si en un proyecto ágil sólo colaborar el equipo técnico y el Scrum Master.*



- Si solo colaboran el Equipo con el Scrum Master, terminaré el producto de manera correcta en el tiempo correcto, pero puedo no estar haciendo el producto que el cliente quiere

- Si sólo colaboran el Equipo y el Product Owner, probablemente hago las cosas con calidad y lo que el cliente quiere, pero sin respetar lo que me dice scrum (no controlo tiempo, iteraciones, etc)

- Si sólo colaboran el Scrum Master y el Product Owner, voy a tener el producto correcto en los tiempos adecuados, pero sin calidad.

Si en un proyecto ágil solo colaborasen el equipo técnico y el scrum master; lo que pasaría es que el software que van a diseñar para la compañía no tendrá todas las características que debería tener; por ejemplo

La empresa quiere que le desarrollen un auto y contratan a una empresa para que le desarrollen un auto, pero como cada vez que al equipo le surge una duda como, por ejemplo, el color, el tamaño, el número de asientos, etc. Lo que se va a terminar construyendo es un auto familiar genérico, de 4 puertas, negro etc. Pero quizás la empresa lo que quería era una camioneta 4x4 con todos los gadgets que existen y toda pintada por fuera, pero como el product owner nunca participo, no dio su opinión y el equipo hizo lo que les parecía dentro de su rango de aplicación.