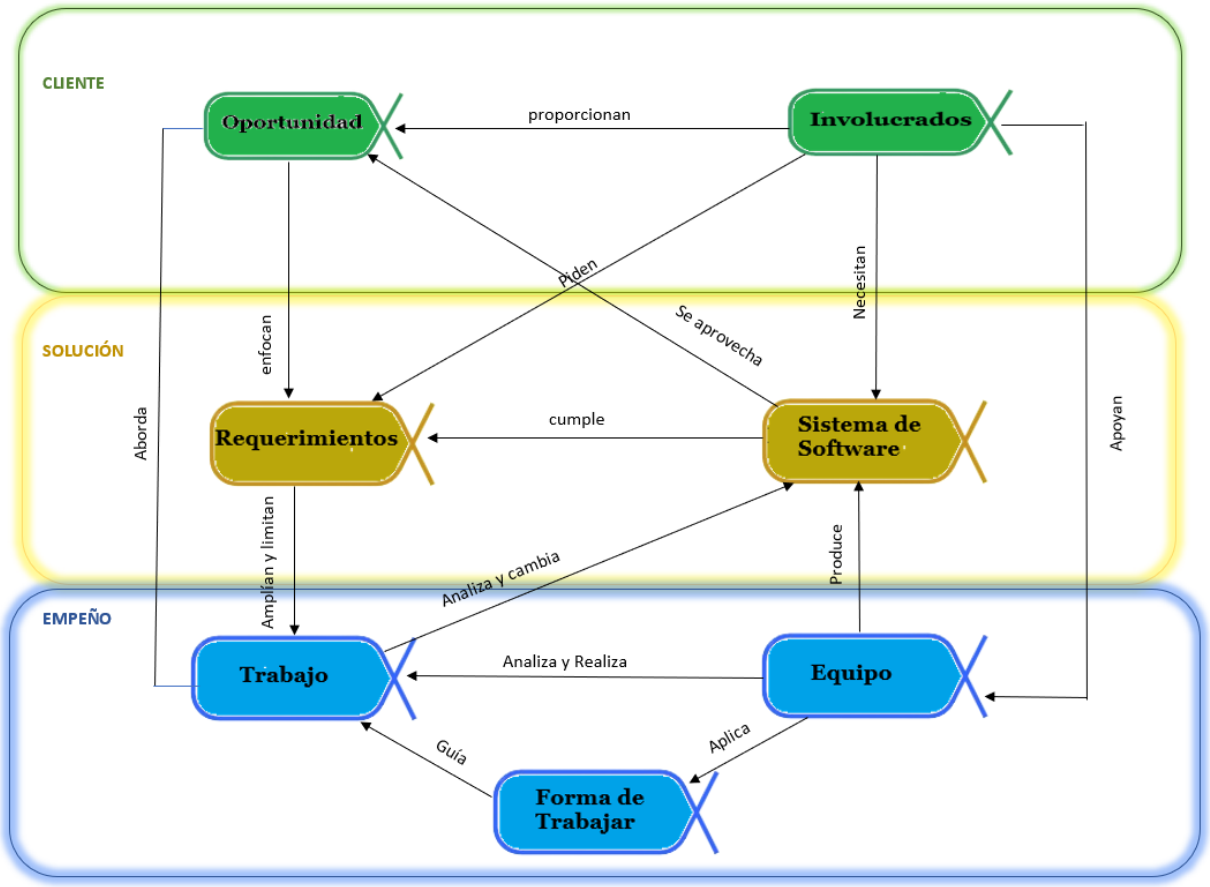
# Tarea en Grupo Metodología de Desarrollo de Software

Mariana Acevedo González
Pablo Andrés Arboleda Bolívar

Corporación Universitaria Remington
Ingeniería de Software I
Docente: Yury Montoya Pérez

Cali, Colombia
18 de febrero de 2024

1.




2. Las ventajas y beneficios del modelamiento de un proyecto de desarrollo de software son las siguientes:

   2.1. Se logra una exposición del proyecto más clara y fácil de entender para todos los miembros del equipo de trabajo.

   2.2. Se establecen estrategias de trabajo para materializar de manera más eficiente las ideas pensadas para el proyecto.

2.3. Permite una mejor organización y orientación del equipo de trabajo el cual tendrá un accionamiento más eficiente respecto a su labor debido a que conocerá el proyecto a grandes rasgos y simplificado.

La capacidad de un ingeniero de software para modelar de forma correcta un proyecto de software es necesaria e importante para poder manejar de manera acertada un equipo de trabajo y así tener un orden y una orientación de acuerdo a los objetivos que se tienen.

De acuerdo a un artículo publicado por la Universidad de Veracruz sobre el Desarrollo de software, un modelado " Debe incluir sólo aquellas actividades, controles y productos del trabajo que sean apropiados para el equipo del proyecto y para el producto que se busca obtener." todo proceder que se encuentre fuera de este marco esquemático del modelo será visto como algo que no contribuya significativamente sobre la realización del proyecto, y un ingeniero de software debe ser un líder capacitado para mantener a su equipo de trabajo centrado en un marco eficiente para lograr una entrega de valor significativo al cliente.

3. Las ceremonias de scrum están compuestas por los siguientes ítems:

   3.1. Sprint planning: Es la ceremonia de inicio del sprint, en ella se establece el objetivo del sprint y el compromiso de terminarlo. Para un sprint de un mes el sprint planning suele tener una duración máxima de 8 horas.

   3.2. Daily Scrum: Esta ceremonia de realiza cada día con el fin de que el equipo de desarrollo sincronice sus actividades. Suele tener una duración máxima de 15 minutos

   3.3. Sprint review: cerca de la finalización del sprint se muestran los avances e incrementos del producto para su inspección, para sprints de 1 mes tiene una duración máxima de 4 horas.

3.4. Sprint Retrospective: Es la ceremonia de cierre del Sprint, después del sprint review el equipo de desarrollo se reúne para identificar accionables de mejora y así poder aplicarlos en el siguiente sprint.

4. Las historias de usuario son pequeñas fracciones de información que suelen ser elaboradas por la persona a cargo del rol de product Owner compuestas a su vez de otros elementos que son parte del concepto de la triple C que es el siguiente:

   4.1. Card: Haciendo alusión a una pequeña nota donde se ejemplifica la petición de un usuario de una funcionalidad específica en el producto. ejemplo: Yo como usuario quiero un botón para compartir en la aplicación web. Esto se usa con el propósito de exponer lo más directa y claramente posible la funcionalidad que se desea tener en el producto.

   4.2. Conversación: Se refiere a un pequeño lapso de tiempo donde se explica de manera verbal los aspectos de la funcionalidad que sean complicados de resumir en una nota. Esto suele llevarse a cabo durante o después de exponer los "criterios de aceptación" que son los requisitos que deben tomarse en cuenta a la hora de diseñar la funcionalidad antes expuesta. Ejemplo: para poder diseñar un botón de compartir debe tomarse en cuenta las apis para enlazar a una aplicación con otra y así poder transferir datos entre ellas.

   4.3. Confirmación: Es la última fase de la historia de usuario donde se decide si se lleva a cabo o no el desarrollo de la funcionalidad.

Las historias de usuario se usan para una exposición clara, rápida y directa de las funcionalidades que un cliente quiere implementar. Sus beneficios son mayor facilidad de entendimiento a la hora

de presentar los conceptos al equipo de trabajo y mayor eficiencia a la hora de coordinar el equipo.

5. El testing es una actividad en un sprint donde se analizan las funcionalidades que se planea desarrollar donde se verifica si estas son viables o no. Las historias de usuario se testean sometiéndolas a un análisis donde se toman en cuenta los criterios de aceptación propuestos por el equipo de desarrollo, si dicha historia de usuario logra un resultado positivo luego de contraponerla con los criterios, entonces la historia pasa a la etapa de confirmación, donde se convertirá en una tarea.
6. Los "casos de prueba" son representaciones hipotéticas de las acciones que realizaría un usuario al utilizar determinado programa, estas representaciones deben ser tomadas en cuenta antes de hacer una entrega de valor al cliente, debido a que el software debe prepararse para no arrojar fallos de gravedad o inentendibles por el usuario si este llevara a cabo un mal proceso o si el software no provee la información necesaria. Los casos de prueba son documentaciones que deben abarcar capos como:

   6.1.Identificador único.

   6.2.Objetivo.

   6.3.Prioridad.

   6.4.Trazabilidad.

   6.5.Precondiciones.

   6.6.Entradas.

   6.7.Resultados esperados.

   6.8.Resultados actuales.

7. El manifiesto para el desarrollo de software ágil es una declaración de principios sobre nuevas formas de desarrollar software que surgió en el 2001, esta se reunión se realizó porque en ese momento existían metodología que eran demasiado rígidas en las que involucraba mucho tiempo planearlas, lo que se quiso fue dar una alternativa a estas metodologías, los principios del agilismo son los siguientes:

   7.1. Satisfacer al cliente mediante la entrega temprana y continua: Este principio favorece tanto al cliente como al equipo, por el lado del cliente este recibe el aplicativo rápido lo que hace que se ponga a trabajar en el y por otro lado favorece al equipo de desarrollo ya que el cliente puede dar una retroalimentación del aplicativo y en caso tal el equipo de desarrollo hacer la mejoras sin tener que esperar hasta el final en una entrega completa para realizar los ajustes del aplicativo.

   7.2. Aprovechar el cambio como ventaja competitiva: En este principio se pueden aceptar los cambios del cliente sobre el software aún estando en etapas finales esto es bueno ya que se estará desarrollando un aplicativo que se adapte a lo que desea el cliente.

   7.3. Entregar valor frecuentemente: Este principio especifica que se deben hacer entregas menores del aplicativo por ejemplo ir entregando módulos esto es bueno porque los cambios que se deben hacer en la retroalimentación que de el cliente se pueden ir desarrollando.

   7.4. Cooperación negocio-desarrolladores durante todo el proyecto: Este principio une al equipo de negocio junto con el equipo de desarrollo de software y los pone a trabajar de la mano.

7.5. Construir proyectos en torno a individuos motivados: Este principio enseña que se debe hacer participes al equipo de desarrollo de software en las decisiones que se toman así ellos se apropiarán del trabajo y se sentirán motivados.

7.6. Utilizar la comunicación cara a cara: Desde la pandemia la forma de trabajo tuvo un cambio se pasó en algunas empresas del trabajo físico al trabajo remoto igual independiente de como sea el trabajo la comunicación se debe de dar a fin de desarrollar el software esperado.

7.7. Software funcionando como medida de progreso: Este principio enseña que si el software no funciona no ha progresado el equipo y va más allá sino se ha entregado al cliente, aunque sea un módulo tampoco se ha avanzado.

7.8. Promover y mantener un desarrollo sostenible: Este principio enseña que el equipo de desarrollo debe de mantener un ritmo juntos evitando sobrecargar a los desarrolladores.

7.9. La excelencia técnica mejora la agilidad: Este principio enseña que hay que cuidar los aspectos técnicos a fin de aportar agilidad esto es que el desarrollo debe ser ordenado en cado de requerir una actualización.

7.10. La simplicidad es fundamental: Al cliente no le interesa el esfuerzo que se realice en el desarrollo lo que le interesa es recibir un aplicativo que atienda sus necesidades.

7.11. Equipos auto-organizados para generar más valor: El darle cierta libertad al equipo de desarrollo sin estar continuamente encima de las personas hacen que se sientan motivados para el trabajo.

7.12. Reflexión y ajustes frecuentes del trabajo de los equipos: Se refiere a la revisión continua a fin de ajustarlo y mejorar el rendimiento.

**Referencias**

Tecnologías para la integración de soluciones. (2020). Procesos para la ingeniería de software.

https://www.uv.mx/personal/ermeneses/files/2018/02/Clase8-

Modelos_de_procesos_de_desarrollo_de_software.pdf

SCRUMstudy. (2024). Guía de los fundamentos de Scrum.

www.scrumstudy.com

Sentrio. (2022). Metodologías Agile: los 4 valores y 12 principios del 'Manifiesto Ágil'

https://sentrio.io/blog/valores-principios-agile-manifiesto-agil/

SCRUMstudy. (2022). Guía de los fundamentos de scrum (guía del sbok).

https://www.scrumstudy.com/sbokguide/download-free-buy-sbok

Hanna Oktaba y Eréndira Jiménez. (2018). ESSENCE 1.1 UNAM.

https://www.uv.mx/personal/ermeneses/files/2018/02/ESSENCE_alfas.pdf

Essence – Kernel and Language for Software Engineering Methods Beta 2. (s.f)

https://www.omg.org/spec/Essence/1.0/Beta2/About-Essence