Universidad Autónoma de Centroamérica

Bachillerato en Ingeniería de Sistemas

Análisis y Diseño de Sistemas I

Martes 6pm-8pm

Proyecto de Investigación

Historias de Usuario (SCRUM) y el Levantamiento de Requerimientos (Desarrollo en Cascada).

Integrantes:

Andrés Rivera Espinoza.

Esteban Córdoba Cedeño.

Johan Cubillo Salazar.

Profesor: Michael Hudson Baltodano.

Fecha de entrega: 09 de agosto del 2022.

II Cuatrimestre 2022

Introducción:

En los últimos años la tecnología ha avanzado rápidamente con grandes progresos tanto en hardware como en software. Así mismo, las necesidades de las organizaciones en cuanto a sistemas de información demandan productos de software cada vez más grandes y complejos sin comprometer la calidad. En estos tiempos, donde la disponibilidad e integridad de la información es un factor crítico, es necesario contar con metodologías para tener sistemas de software confiables que sean entregados a tiempo y con costo reducido.

En el desarrollo de software, una metodología hace cierto énfasis al entorno en el cual se plantea y estructura el desarrollo de un sistema, siendo uno de los sectores tecnológicos más competitivos y no es algo nuevo, ya que durante muchos años lo ha sido, sin embargo, ha tenido una evolución constante en lo que se refiere a las metodologías o bien, las formas en las cuales se realiza la planeación para el diseño de cualquier software con el uso correcto de las diversas herramientas, técnicas, métodos y modelos para el desarrollo.

Así mismo, conociendo más a fondo las metodologías de Scrum y Cascada, podremos implementar dichas metodologías dentro de un proyecto a futuro, ya que, siendo alumnos de Ingeniería en Sistemas, es fundamental conocer su uso adecuado.

Desarrollo:

¿Qué es SCRUM y cómo funciona la documentación de requerimientos por medio de historias de usuario?

Scrum es una metodología de desarrollo utilizada en el desarrollo de Software basada en un proceso iterativo e incremental. Scrum es un marco ágil, adaptable, rápido, flexible y eficaz que está diseñado para ofrecer valor al cliente durante todo el desarrollo del proyecto. El objetivo principal de Scrum es satisfacer la necesidad del cliente a través de un entorno de transparencia en la comunicación, responsabilidad colectiva y progreso continuo.

Las historias de usuario son una forma de definir un requisito con los objetivos que cada una sea:

- Independiente: Debe ser totalmente atómica en su definición
- Negociable: Ambiguas en su enunciado para poder ser debatidas. La concreción vendrá dada por los criterios de validación.
- Valorable: Es importante conocer el valor que aportan al proyecto, poder valorar la cantidad de trabajo que suponen para poder planificar y gestionar el proyecto.
- Pequeña: Para poder hacernos una idea rápida una historia de usuario debería poderse llevar a cabo durante un tiempo superior a dos días e inferior a una semana.
- Verificable: Es obligatorio verificar que el software cumpla con la funcionalidad acordada, se hace mediante los criterios de validación.

Ventajas

- ✓ Ayuda a establecer metas cuantificables y, así, mantener al equipo ocupado todo el tiempo para garantizar la productividad.
- ✓ La flexibilidad del método Scrum te permite modificar el hilo conductor de los hechos en cualquier momento, lo que favorece la resolución de conflictos u obstáculos sobre la marcha.
- ✓ Evita el perfeccionismo innecesario.
- ✓ Muestra una visión completa del proyecto.

Desventajas

- ✓ Si por algún motivo quedan tareas sin finalizar, el resto de los pendientes comenzará a postergarse indefinidamente, pues a menudo hay una relación lógica y secuencial entre las actividades del tablero.
- ✓ Al momento de trabajar con Scrum, todo el equipo debe conocer a fondo sus principios y marco teórico. Tiene que haber conocimiento de los **roles de Scrum** o podría haber dispersión y choque de funciones.

¿Qué es una historia de Usuario y como se relaciona a la metodología SCRUM?

Las historias de usuario son pequeñas descripciones de los requerimientos de un cliente. Su utilización es común cuando se aplica marcos de entornos ágiles como Scrum. Al redactar las historias de usuario se debe tener en cuenta describir el Rol, la funcionalidad y el resultado esperado en una frase corta.

Una historia de usuario sigue el siguiente formato
Como **<quién>** Quiero <**qué**> Para <**objetivo**>.

Ejemplo: Como Vendedor, quiero registrar los productos y cantidades que me solicita un cliente para crear un pedido de venta.

Aspectos para tomar en cuenta a la hora de escribir una historia de Usuario:

Definir quién utilizara la funcionalidad a desarrollar
Es útil imaginarnos qué características tienen las personas que usarán el producto, y detallar su necesidad y problemas actuales para dar lugar al entendimiento sobre sus expectativas reales.
Especificar qué producto quiere el usuario
Las historias de usuario deben describir qué se espera como salida de la implementación, y cómo se ve beneficiado el usuario final.

Para qué utilizará el producto
En este sentido es importante definir el contexto donde surge la historia que se está creando.

Los criterios de aceptación
Aquí se especifica qué salidas obtendremos cuando finalice el proceso de ejecución de la funcionalidad, y nos sirve para verificar que está terminada la funcionalidad.

Y finalmente, los comentarios
Las historias de usuarios facilitan la interacción permanente con el cliente para verificar que lo que estamos construyendo está de acuerdo con sus expectativas.

Cómo se estima la duración y/o esfuerzo asociado a cada historia de usuario de un Sistema?

Debemos tener en cuenta muchos aspectos entre ellos:
Saber que cada equipo, cada producto tiene su contexto y tendrá sus propios números, prácticas emergentes y conclusiones.
Recordemos que el tiempo requerido para el DONE de una historia de usuario debe incluir todas las tareas técnicas que sean relevantes y requeridas, por ejemplo:

- Análisis
- Diseño
- Implementación
- Revisión Par (esta es una buena práctica)

- Despliegue
- Pruebas
- Corrección
- Despliegue
- Pruebas
- Actualización de documentación relevante para el equipo.

Luego de realizar todo esto podemos realizar una estimación del tiempo necesario para realizar el proyecto con respecto a la historia de Usuario.

ESFUERZO SUGERIDO PARA LAS HISTORIAS DE USUARIO*

	Duración del Sprint							
	1 Semana		2 Semanas		3 Semanas		4 Semanas	
Días Hábiles	5		10		15		20	
Cantidad de Historias	10	6	10	6	10	6	10	6
Días-EQUIPO requeridos para el Done	0.5	0.8	1.0	1.7	1.5	2.5	2.0	3.3
Días-PERSONA** requeridos para el Done - Equipo de 3 personas	2	3	3	5	5	8	6	10
Días-PERSONA** requeridos para el Done - Equipo de 5 personas	3	4	5	8	8	13	10	17
Días-PERSONA** requeridos para el Done - Equipo de 7 personas	4	6	7	12	11	18	14	23
Días-PERSONA** requeridos para el Done - Equipo de 9 personas	5	8	9	15	14	23	18	30

* Basado en el post: http://agileforall.com/patterns-for-splitting-user-stories/
en el que se propones que las historias deben tener de 1/6 a 1/10 de la velocidad del equipo por sprint
** Los números fueron aproximados al entero superior

Basado en la sugerencia de **Thomas Wallet @WalletThomas,** en el que me mostraba que **tener historias gigantes no es buena buena práctica** puse la clasificación amarillo, naranaja,y roja mostrando que hay tamaños grandes de historias de usuario que posiblemente se constituya en unas épicas susceptibles de ser divididas

El esfuerzo invertido en esta historia es grande, se sugiere hacer partición de la Historia de Usuario

El esfuerzo invertido en esta historia de usuario vs el tamaño del sprint lo pone en riesgo que se logre en el tiempo comprometido, es un tamaño de historia riesgoso, se sugiere realizar división del a historia

Definitivamente no se recomiendan historias de usuario de este tamaño ya sea por que están cerca, iguales o exceden el tamaño del sprint, o por que su tamaño es lo suficientemente grande y es altamente factible que pueda ser dividida en historias de usuario más pequeñas

Elaborada por: **Jorge H. Abad L.**
Blog: **www.lecciones-aprendidas.info**
Twitter: **@jorge_abad**

¿Se puede emplear el uso de UML (completo o parcial) en la documentación de historias de usuario?

Lo recomendable es que no ya que la Historia de Usuario está escrita en lenguaje coloquial al ser, simplemente, el recordatorio de la conversación con el cliente. Y un acuerdo formal de mínimos para dar por buena la funcionalidad descrita y esperada.

¿Qué es la metodología Cascada y cómo se documentan tradicionalmente los requerimientos del sistema?

Consiste en el desarrollo de un proyecto de manera secuencial con el fin de ordenar de forma lineal las distintas etapas que se deben de seguir al momento de desarrollar cualquier software. Originalmente fue propuesto en 1970 por Winston W. Royce, es también conocido como modelo lineal o modelo de ciclo de vida de un programa, y millones de personas lo han incorporado a sus planes en las últimas cinco décadas.

Las diferentes fases de un proceso de desarrollo se suceden una detrás de otra como en una cascada. Cada una de las fases concluye con un resultado provisional (hito) como, por ejemplo, un catálogo de requisitos en forma de pliego de condiciones, la especificación de una arquitectura de software o una aplicación a nivel alfa o beta.

- Análisis: planificación, análisis y especificación de los requisitos.
- Diseño: diseño y especificación del sistema.
- Implementación: programación y pruebas unitarias.
- Verificación: integración de sistemas, pruebas de sistema y de integración.
- Mantenimiento: entrega, mantenimiento y mejora.

Ventajas

- ✓ Una estructura sencilla gracias a unas fases de proyecto claramente diferenciadas.
- ✓ Buena documentación del proceso de desarrollo a través de unos hitos bien definidos.
- ✓ Los costes y la carga de trabajo se pueden estimar al comenzar el proyecto.
- ✓ Aquellos proyectos que se estructuran en base al modelo en cascada se pueden representar cronológicamente de forma sencilla.

Desventajas

- ✓ Por norma general, los proyectos más complejos o de varios niveles no permiten su división en fases de proyecto claramente diferenciadas.
- ✓ Poco margen para realizar ajustes a lo largo del proyecto debido a un cambio en las exigencias.
- ✓ El usuario final no se integra en el proceso de producción hasta que no termina la programación.
- ✓ En ocasiones, los fallos solo se detectan una vez finalizado el proceso de desarrollo.

¿Qué es un requisito de sistema y cómo se relaciona a la metodología Cascada?

Los requerimientos/requisitos de un sistema describen los servicios que ha de ofrecer el sistema y las restricciones asociadas a su funcionamiento.

Se dividen en requerimientos funcionales y no funcionales.

Requerimientos funcionales: Expresan la naturaleza del funcionamiento del sistema (cómo interacciona el sistema con su entorno y cuáles van a ser su estado y funcionamiento).

Requerimientos no funcionales: Restricciones sobre el espacio de posibles soluciones. Rendimiento del sistema: Fiabilidad, tiempo de respuesta, disponibilidad… Interfaces: Dispositivos de E/S, usabilidad, interoperabilidad… Proceso de desarrollo: Estándares, herramientas, plazo de entrega…

Se podría deducir de los términos anteriores que los requisitos funcionales definen que debe hacer un sistema, mientras que los no funcionales definen como debe ser el sistema.

A los requisitos no funcionales se les suele llamar coloquialmente “cualidades” del sistema y pueden dividirse en dos categorías:
Cualidades de ejecución, como la seguridad o la usabilidad, observables en observables en tiempo de ejecución.
Cualidades de evolución, como la “testabilidad”, mantenibilidad, extensibilidad o escalabilidad, determinadas por la estructura estática del software.

La distinción entre requerimientos funcionales y no funcionales no siempre resulta evidente.

Ejemplo: La seguridad puede interpretarse inicialmente como un requerimiento no funcional al principio. No obstante, su elaboración puede conducir a nuevos requerimientos funcionales, como la necesidad de autentificar a los usuarios del sistema.

Más allá de si decidimos incluir este tipo de requisitos en una sección u otra, lo importante es identificarlos correctamente.

La relación de los requerimientos del sistema y la metodología cascada se da debido a que en el modelo original en cascada una de sus siete fases consiste en los requisitos de sistema, inclusive en la primera fase del modelo, en la cual se ocupa de los requisitos que no están relacionados con el producto digital en sí, sino más bien con aspectos relevantes para la empresa como el precio y la disponibilidad. Aquí también se especifican los aspectos de documentación y seguridad. En general, aquí se mencionan los requisitos no funcionales.

Cómo se estima la duración y/o esfuerzo asociado a cada requerimiento.

Las tareas por realizar en un proyecto de software se miden en términos de esfuerzo requerido en horas/hombre, por lo que poseen un costo económico y forman parte del presupuesto. En el vínculo contractual que une al cliente con la empresa encargada de desarrollar el software, se acuerdan tres variables importantes: alcance del producto a desarrollar, precio y tiempo incurrido para el desarrollo.

La estimación del esfuerzo de cada tarea debe ser lo más precisa posible, ya que de esto depende, en gran parte, el éxito del proyecto. La entrega del producto en el tiempo estipulado, el uso eficiente del presupuesto asignado y el cumplimiento de las funcionalidades requeridas, inciden en la calidad del producto y muestran la eficacia del proceso de desarrollo.

Es muy difícil que la estimación del esfuerzo sea exacta, dado que hay muchas variables que afectan al costo final del proyecto y al esfuerzo aplicado para desarrollar el producto. No obstante, la estimación de esfuerzo de un proyecto de software dejó de ser un “arte” para convertirse en un modelo sistemático que proporciona valores con un grado aceptable de riesgo.

Ante un proyecto de software, luego de establecerse el objetivo y los requerimientos que debe cumplir el producto, debe seleccionarse una alternativa para estimar el esfuerzo requerido, entre ellas:

a) Basarse en la experiencia y similitud de proyectos anteriores.
b) Utilizar un modelo empírico.

La primera opción puede resultar positiva en casos en que el proyecto posea similitudes con proyectos anteriores, o se reutilice diseño o código de un proyecto pasado. Sin embargo, la realidad marca que todos los proyectos difieren unos con otros, y el contexto (clientes, recursos humanos, tecnología, situación político-económica, entre otros) en que ellos se desarrollan es diferente. Así, en la mayoría de los casos, basarse en una experiencia anterior no ha sido indicador de buenos resultados.

La segunda opción utiliza fórmulas derivadas empíricamente que predicen el esfuerzo requerido basándose en variables de entrada tales como calidad y experiencia del equipo del proyecto, tecnología a utilizar, complejidad del dominio, datos de salida, volatilidad de los requerimientos, entre otras. Estos modelos están basados en un conjunto de proyectos que fueron utilizados como casos de estudio o proyectos de muestra. Los modelos deben ser aplicados con prudencia y ajustados al contexto local de la organización que los aplica, en función de sus proyectos históricos.

Algunos de las principales metodologías son: Lines of Code o líneas de código (LOC), Constructive Cost Model o Modelo del costo de construcción (CoCoMo I), CoCoMo II y Function Points o Puntos Función.

¿Se puede emplear el uso de UML (completo o parcial) en la documentación de requerimientos?

La utilización del lenguaje unificado de modelo debe ser completa en la documentación de requerimientos ya que esto facilita la compresión no solamente para nosotros como ingenieros, sino también para los clientes que no estén solicitando la realización del sistema.

- Modelo de casos de uso: El modelo de casos de uso depende de los actores, los cuales son entidades externas al sistema los cuales no son afectados por el proceso actual de diseño de este, pero tienen una fuerte interacción posterior con él. El modelo de caso de uso es una explicación detallada de cómo se llevará a cabo esa interacción entre el sistema y los actores.

- Diagramas de clases: Muestran las entidades que necesitan trabajar juntas para asegurarse que el sistema realiza cada uno de los requerimientos especificados en los casos de uso. Los diagramas de clases contienen los requerimientos funcionales de sistema.

- Diagramas de secuencia: El diagrama de secuencia muestra cómo los objetos en el diagrama de clases colaboran entre sí para llevar a cabo los casos de uso; cada caso de uso está relacionado con uno o más diagramas de secuencia.

- Diagramas de actividad: Son utilizados para visualizar el comportamiento dinámico de una parte del sistema.

- Diagramas de estados: Pueden ser utilizados para representar el comportamiento de una clase, un caso de uso o un sistema. Los componentes clave de los diagramas de transición de estrados son los estados, los cuales interactúan por medio de eventos que pueden llevar de un estado a otro mediante una transición o que permanezca en el mismo estado como resultado del evento.

- Especificación de requerimientos de software (ERS): El ERS es el “artefacto” final del proceso de desarrollo de requerimientos. El ERS contiene además de los requerimientos funcionales de cada una de las clases mencionadas en el previo diagrama de clases, como también los no funcionales,

Conclusiones:

Scrum no es ni la mejor metodología ni la única, pero es una de las metodologías con un crecimiento fuerte por la facilidad de producción de software y por su agilidad en cuanto a cambios y lo que propiamente aporta en comparación con otras metodologías. Por un lado, Scrum evita la generación documental. No es que con Scrum no se deba o no se pueda documentar, si no que con Scrum no se exige documentar nada para iniciar un proyecto, algo que en otras metodologías es impensable.
Es más rápido que las metodologías tradicionales en obtener resultados y también su proceso es más flexible, aunque no es tan recomendable en equipos muy numerosos y que no están acostumbrados a asumir responsabilidades permanentes.

Por otro lado, el modelo en Cascada es un método clásico de desarrollo de software orientando a proyectos que tengan bien definidos sus requerimientos y que no sean entregados en tiempo corto. Si se analiza el modelo en cascada se enfoca puntos importantes, uno de esos puntos es que este método nació para para la construcción de hardware y que con el pasar el tiempo se acoplo al desarrollo de software.

Bibliografía:

https://elvex.ugr.es/idbis/db/docs/design/2-requirements.pdf

https://www.softwaretestinghelp.com/what-is-sdlc-waterfall-model/

https://pdfs.semanticscholar.org/6414/65f37add5bbff9dd4a8d13f0cf8a9e866e91.pdf

https://robertohernandez18.files.wordpress.com/2014/09/uso-de-uml-en-los-requerimientos-de-software.pdf

http://go.microsoft.com/fwlink/p/?LinkId=255141

https://www.genbeta.com/desarrollo/historias-de-usuario-una-forma-natural-de-analisis-funcional

http://www.lecciones-aprendidas.info/2017/05/Tamanos-sugeridos-de-historias-de-usuario.html

https://proyectosagiles.org/que-es-scrum/