Reflexiones Unidad 2

Arturo Cadena Méndez

CU vs US

Para saber cuál es la diferencia entre la especificación de casos de uso y las historias de usuario, primero hay que definir cada uno.

La especificación de casos de uso se refiere al modo en que un actor(cosa externa que se comunica con el producto) interactúa con el sistema, es decir, una narración que habla acerca del rol desempeñado por el actor en su interacción con el producto. Esta igual se puede representar a través de un Diagrama de Casos de Uso.

Las historias de usuario son descripciones generalmente pequeñas e informales las cuales explican funciones de software vistas desde la perspectiva de un usuario. Estas suelen seguir el siguiente formato: **“Como [perfil], quiero [objetivo del software], para lograr [resultado]”.**

Ahora bien, la diferencia radica en que las historias de usuario representan el qué se va a hacer o qué quiere el cliente, y los casos de uso el cómo va a hacer, ya que las historias de usuario son características y marcadores de lo que se debe construir y en cambio los casos de uso son visiones contextuales de lo que se va a construir. Otra diferencia muy importante entre ambas es que las historias de usuario pueden representar mejoras y correcciones en requisitos, mientras que los casos de uso solo son representaciones de los requisitos funcionales. También es que para la aceptación de cada una, los casos de uso requieren de matrices de seguimiento de requisitos con porcentajes y las historias de usuario solo utilizan la aceptación de forma binaria, es decir vale o no vale.

En cuestión de en qué casos utilizar cada uno, los casos de uso son altamente recomendables en sistemas interactivos, debido a que muestran la intención que tiene el actor al hacer uso del sistema, también cuando se busca evitar que el

sistema se base únicamente en criterios tecnológicos, ya que los casos de uso permiten que el analista se centre en las necesidades del usuario.

Las historias de usuarios se recomiendan cuando se utilizan técnicas de metodología ágil como lo es Scrum, porque permiten agilizar los procesos. Por último las historias son muy útiles cuando se quiere facilitar el entendimiento entre personas que normalmente no tienen la misma perspectiva.

Referencia:

Scrum México. 2018. Escribiendo Historias de Usuario. Recuperado de https://scrum.mx/informate/historias-de-usuario

Seguridad Para IS

Hoy en día la seguridad en la Ingeniería de Software es de suma importancia, ya que permite a los sistemas seguir trabajando a pesar de los posibles ataques que puede haber y que lleguen afectar a los usuarios. Por lo tanto, es vital que un Ingeniero de Software tenga conocimientos sobre seguridad para completar su formación académica. Dicho esto, los conocimientos que un Ingeniero de Software debe elegir para su formación académica son los siguientes:

- **Fundamentos de seguridad**: En esta parte se incluyen los aspectos a tratar, el propósito y papel de la seguridad, sus objetivos, principios básicos de la seguridad, mecanismos y políticas, y ataques o amenazas. El principal propósito de estos fundamentos es conocer los aspectos básicos de seguridad en los sistemas y la importancia que tiene en la sociedad abierta y conectada.
- **Seguridad organizativa**: El objetivo de esta competencia es conocer la importancia de la seguridad dentro de las organizaciones a través del conocimiento de la seguridad organizativa, de las políticas y procedimientos de seguridad en la organización, la clasificación y controlo de activos y seguridad personal.
- **Requisitos de seguridad**: La Ingeniería de Requisitos de Seguridad se define como la parte que proporciona técnicas, métodos y normas para conseguir sistemas de software seguros. Todo esto incluye el concepto de requisito no funcional, la ingeniería de requisitos, definición y clasificación de requisitos de seguridad, técnicas y modelos de Ingeniería de Requisitos de Seguridad.
- **Seguridad en el desarrollo de Software:** Tiene como fin capturar los principales conceptos y aspectos más relevantes en cuanto a la incorporación de la seguridad en los procesos de desarrollo de software. Para lograr esto se requiere saber sobre desarrollo de software, importancia de la seguridad en el desarrollo de software y propuestas de seguridad en procesos de desarrollo.

- **Seguridad en Sistemas de Información:** Tiene como propósito dar una visión general de la problemática de la seguridad en los sistemas de información, dando a conocer cuáles son las técnicas, mecanismos, políticas, protocolos, etcétera más utilizadas en los sistemas de información. Para obtener está visión general se requieren conocimientos de seguridad lógica, criptografía, seguridad en internet, seguridad en sistemas operativos y seguridad en bases de datos.
- **Riesgos de Seguridad:** Es identificar todos los activos importantes para la seguridad de los sistemas de información, las amenazas que pueden afectarles, calcular el riesgo existente de un posible impacto sobre el activo. Con toda esta información será posible tomar decisiones pertinentes para implantar medidas de seguridad optimizando el factor riesgo-inversión.
- **Servicios de seguridad:** Su objetivo es mejorar la seguridad de los sistemas de procesamiento de datos y la transferencia de información en las organizaciones con el conocimiento de servicios de seguridad básicas y avanzados.
- **Gestión de Seguridad:** Se refiere a la administración de la seguridad en los sistemas de información usando la gestión, planificación y técnicas para la gestión. de la seguridad TI.
- **Certificación, normas y estándares para la seguridad:** Significa gestionar serie de datos y recursos de forma competente y efectiva identificando los riesgos a los que se someten adoptando medidas adecuadas y proporcionadas. Todo esto se hace conociendo las certificaciones de seguridad y especificaciones y estándares de seguridad.

Referencia:

UCLM. Universidad Castilla-La Mancha: La Seguridad como una asignatura indispensable para un Ingeniero del Software. Recuperado de: https://upcommons.upc.edu/bitstream/handle/2099/11778/a25.pdf

Mantenimiento de Software como Competencia

El programa de estudios de Ingeniería de Software define a la competencia de Mantenimiento de Software como el conjunto de conocimientos y habilidades que tiene un Ingeniero de Software para mantener productos de software heredados en diferentes dominios de aplicación, optimizando los recursos humanos, materiales, económicos y de tiempo, y atendiendo las necesidades de la organización. Ahora bien, ¿será correcto considerar al Mantenimiento de Software como una competencia de la licenciatura? El programa de estudios de la UADY la incluye como competencia, debido a que se basó en la información del Institute of Electrical and Electronics Engineers (IEEE), que a su vez elaboró junto con la Association for Computing Machinery el SWEBOK V3.0, que es una guía que presenta un esquema de conocimientos y competencias que todo ingeniero de software debe conocer por ser relevante para su actividad profesional. Aparte eso, se utilizaron los Modelos Curriculares del ANIEI para determinar cada competencia, incluyendo la de Mantenimiento de Software. Dicho esto, las razones que se dan por las que el mantenimiento es fundamental son las siguientes:

El mantenimiento ayuda a corregir errores, mejorar el diseño, implementar, mejorar y adaptar programas para que todo lo que incluye el sistema pueda ser usado sin ningún problema, entre otras cosas. No solo eso, sino igual que el mantenimiento perfecciona las funciones ya existentes, identifica amenazas de seguridad, repara las seguridades vulnerables e impide que el rendimiento del software se degrade a niveles inaceptables, etc.

Entonces, podemos decir que es correcto que el programa de LIS-FMAT incluya al mantenimiento de software como una competencia, ya que le permite al egresado realizar un sinfín de tareas con el propósito de mejorar o reparar un sistema, porque un software sin ninguna de estas dos sería un trabajo anticuado y poco útil.

Referencia:

UADY. 2016. Universidad Autónoma de Yucatán. Plan de estudios de Licenciatura en Ingeniería de Software. Recuperado Octubre 29, 2022 de https://www.matematicas.uady.mx/files/documents/programas/lis/LIS_Aprobado_12-ago-2016.pdf

Políglotas

Hoy en día, aprender varios lenguajes de programación significa mejorar en el mundo laboral y empresarial, ya que eso permite tener conocimientos amplios en software y la posibilidad de trabajar en proyectos de cualquier tipo. La cantidad de lenguajes de programación es cada vez más grande y aprenderlos todos o la mayoría puede sonar imposible, pero si se aplican las estrategias correctas será mucho más sencillo un aprendizaje y una transición rápida entre diferentes lenguajes.

Una de las estrategias es la de aprender los fundamentos de programación, que sería el conjunto de conocimientos básicos que se pueden aplicar a cualquier lenguaje, como por ejemplo el concepto de función. Conociendo los fundamentos es más fácil entender lo que se busca hacer y facilita el aprendizaje. Básicamente sería como manejar autos; los fundamentos de los autos serían los pedales, el movimiento del volante y las precauciones que uno necesita saber al manejar, conociendo todo eso, va a ser más sencillo manejar en coches con características distintas al que usualmente manejamos, porque ya conocemos las bases de cómo conducir cualquier tipo de coche y solo nos faltaría saber unos cuantos detalles del vehículo.

Otro punto muy importante, es el saber inglés. Aunque suene muy obvio, el dominar este idioma nos facilita a la hora de programar en cualquier lenguaje, ya que en su gran mayoría son en inglés, aparte de que igual el contenido y libros que hay sobre los lenguajes son en inglés.