Ingeniería de Software: Desarrollar es mucho más que programar.

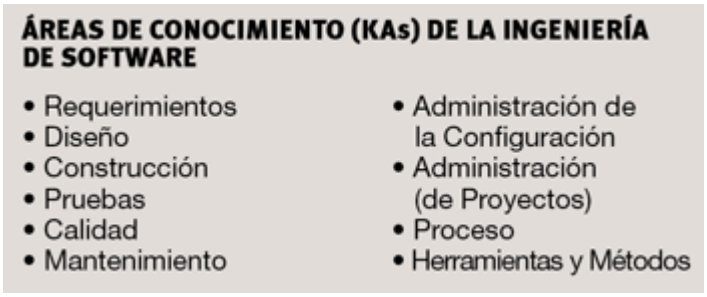
Publicado en: SG #01 (/REVISTA/01)

FUNDAMENTOS (/SECCION-REVISTA/FUNDAMENTOS)

Y entonces, ¿qué conocimientos debe tener un ingeniero de software? Precisamente esto es lo que busca responder el Software Engineering Body of Knowledge (SWEBOK), un proyecto auspiciado por la IEEE para lograr un consenso mundial de lo que es esta disciplina y el lugar que tiene junto a otras ingenierías. El SWEBOK define diez Knowledge Areas (KAs) o áreas de conocimiento.

Requerimientos de Software

Los requerimientos de software expresan las necesidades y restricciones que debe satisfacer un producto para contribuir a la solución de un problema real. Esta área de conocimiento considera la obtención, análisis, especificación y validación de los requerimientos, así como el rol que juegan dentro del proceso de desarrollo de software. Un especialista en requerimientos tiene conocimiento y experiencia en técnicas para obtener, cuantificar, negociar, clasificar, priorizar, modelar, documentar y validar los requerimientos de software. Además debe saber administrar adecuadamente los cambios a éstos.



Diseño de Software

El diseño juega un rol clave en el desarrollo de software ya que es donde se generan modelos que sirven como “planos” para la construcción. Típicamente se divide en dos tipos:
• Diseño arquitectónico - Describe la estructura y organización de alto nivel de un sistema. Identifica los componentes e interfaces entre éstos.
• Diseño detallado - Describe individualmente cada componente con suficiente detalle para ser construido.

Esta área concentra una gran cantidad de conocimiento. Para empezar, el diseño de software requiere entender a fondo principios como la abstracción, acoplamiento, cohesión, descomposición y encapsulación, ya que son la base para diseñar sistemas robustos. También es necesario saber resolver aspectos prácticos como la persistencia de datos, sistemas distribuidos, peticiones concurrentes, manejo de eventos, recuperación a fallas, etc. Por último, un diseñador que no quiera “reinventar la rueda” cada vez que se le presente un problema, debe estar familiarizado con “patrones”, soluciones documentadas a problemas comunes.

Construcción de Software

Esta área de conocimiento se refiere a la creación de software útil a través de la programación, depuración (debugging), pruebas unitarias e integración de componentes. La construcción lidia con la creación y aplicación de algoritmos para la resolución de problemas, así como su implementación utilizando algún lenguaje de programación. Todo esto se debe hacer buscando minimizar la complejidad y cumpliendo estándares para que el código generado sea entendible y extensible por otros, además de que esté optimizado para consumir la menor cantidad de recursos posible.

Pruebas de Software

Las pruebas de software consisten en la verificación dinámica del comportamiento real de un programa comparado con su comportamiento esperado en un conjunto finito de casos de prueba seleccionados de un dominio de ejecuciones típicamente infinito. Las pruebas se realizan para evaluar la calidad de un producto a través de la detección de fallas en éste. Sin embargo, las pruebas de software han evolucionado para dejar de ser consideradas como algo que comienza sólo hasta que la programación termina, con el limitado propósito de detectar fallas. Es aceptado

que las pruebas deben abarcar el proceso completo de desarrollo, que su planeación comienza durante las primeras etapas del proceso de requerimientos, y que los planes y procedimientos de prueba se deben desarrollar y refinar durante el ciclo completo de desarrollo.

Las pruebas pueden ser de diferentes tipos, ya sea por su alcance (unitarias, integrales, de sistema) o su objetivo (funcionalidad, confiabilidad, desempeño, regresión, aceptación, beta, etc.). Para esto se utilizan diferentes técnicas como tablas de decisión, análisis de fronteras, máquinas de estados, y la experiencia misma. Por último, para cuantificar la calidad de un producto de software es necesario entender métricas como la densidad de defectos y la evaluación de confiabilidad.

Calidad del Software

El área de conocimiento de calidad se enfoca en la aplicación de técnicas estáticas para evaluar y mejorar la calidad del software. Esto difiere del acercamiento utilizado en pruebas, donde las técnicas utilizadas son dinámicas, ya que requieren la ejecución del software. El área de conocimiento de calidad involucra los subprocesos de aseguramiento de calidad, verificación, validación, revisión y auditoria. Además considera tópicos como la clasificación de defectos, control estadístico de calidad, modelos de predicción y análisis de tendencias.

Mantenimiento de Software

El mantenimiento se refiere a las modificaciones a un producto de software previamente liberado para prevenir fallas (preventivo), corregirlas (correctivo), mejorar su desempeño (perfectivo) o adaptarlo a cambios en el ambiente (adaptativo). Históricamente no se le ha dado tanta atención al mantenimiento como al desarrollo de software. Sin embargo, esto está cambiando debido a que las empresas buscan sacar el mayor jugo posible a sus productos existentes. Algunos temas clave en el mantenimiento son la reingeniería, análisis de impacto, pruebas de regresión, y outsourcing del mantenimiento. Además hay que tener en cuenta que el proceso a seguir en el mantenimiento es diferente al que se sigue para el desarrollo, por lo que involucra actividades y artefactos diferentes.

Administración de la Configuración del Software (SCM)

La configuración de un sistema se refiere al conjunto de elementos de hardware y software que lo forman. SCM es la disciplina de identificar la configuración en distintos puntos del tiempo con el propósito de controlar los cambios a ésta, manteniendo su integridad y rastreabilidad durante el ciclo completo de vida del software. SCM va más allá del simple control de versiones, y requiere saber identificar los elementos de configuración, definir un proceso de control de cambios, auditar y reportar el estatus de la configuración, y administrar la integración y liberación del sistema completo.

Administración de la Ingeniería del Software

Esta área de conocimiento es lo que típicamente llamamos Administración de Proyectos o Project Management. Consiste en la aplicación de actividades administrativas —como la planeación, coordinación, medición, monitoreo, control y reporte— para asegurar que el desarrollo y mantenimiento de software se lleva a cabo de manera sistemática, disciplinada y cuantificable. Entre los tópicos más importantes de esta área de conocimiento están la planeación de proyectos, estimación de esfuerzo, asignación de recursos, administración de riesgo, manejo de proveedores, manejo de métricas, evaluación, y cierre de proyectos.

Proceso de Ingeniería de Software

Cada área de conocimiento considera un proceso para las actividades técnicas y administrativas que deben realizarse para adquirir, desarrollar, mantener y retirar software; éste es considerado como un primer nivel de procesos. Adicionalmente existe un segundo nivel, o meta-nivel, que se enfoca en la definición, implantación, evaluación, mejora y administración del cambio de los procesos de primer nivel. A éste es al que se refiere el área de conocimiento de proceso de ingeniería de software.

Herramientas y Métodos de Ingeniería de Software

Las herramientas permiten la automatización de tareas repetitivas y bien definidas, habilitando al ingeniero de software para que se concentre en los aspectos creativos del proceso. Existen una gran cantidad de herramientas para asistir todas las áreas de conocimiento, desde la administración de requerimientos hasta las pruebas automatizadas.

Los métodos de ingeniería de software establecen una estructura para sistematizar las actividades con el objetivo de aumentar las posibilidades de éxito. Esta área de conocimiento se enfoca en los métodos que abarcan múltiples KAs, ya que los que son relativos a una sola área de conocimiento se incluyen en el área correspondiente. Los métodos pueden aplicar técnicas heurísticas (informales), formales, y basadas en prototipos.

Estas son las diferentes áreas del conocimiento de la ingeniería de software. Si quieres conocer más sobre estos temas, no te pierdas futuras ediciones de esta revista, ya que nuestros artículos ahondarán en cada uno de ellos.

Referencia

1. Guide to the SWEBOK. http://www.swebok.org (http://www.swebok.org)