UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

TIA
Tecnología, Investigación y Academia

El papel de la Ingeniería de Software en el desarrollo de aplicaciones

The role of software engineering in application development

Jenny Carolina Bayona Zubieta[1], Olga Lucía Pineda Samacá[2], Oscar Daniel Pardo Mahecha[3]

Para citar este artículo: Bayona, J. C.; Pineda, O. L.; Pardo, O. D. (2016). El papel de la Ingeniería de Software en el desarrollo de aplicaciones, 4(1), 3-14.

ARTÍCULO
DE INVESTIGACIÓN

Fecha de recepción:
29-05-2014
Fecha de aceptación:
14-01-2015

ISSN: 2344-8288
Vol. 4 No. 1
Enero - Junio 2016
Bogotá-Colombia

Resumen

La Ingeniería de Software es una nueva área de la ingeniería y es considerada como una disciplina que se encarga de crear y mantener las aplicaciones de software haciendo uso de tecnologías, prácticas, métodos y técnicas para el desarrollo de programas informáticos con calidad, apoyándose en las herramientas y los procedimientos que provee la informática para su aplicación.

Palabras clave: desarrollo de software, métricas, dirección, control, organización, planificación, diseño, pruebas.

Abstract

Software engineering is a new area of engineering and it is considered as a discipline responsible for creating and maintaining software applications using technologies, practices, methods and techniques for quality software development, relying on tools and procedures that provides computing for its application.

Keywords: software development, metrics, management, control, organization, planning, design, testing.

[1]Ingeniera en Telemática. Correo electrónico: jennycarob@gmail.com
[2]Ingeniera de Sistemas, Universidad Católica de Colombia. Correo electrónico: opineda11259@hotmail.com
[3]Ingeniero en Telemática. Correo electrónico: oscar_dpm@hotmail.com

INTRODUCCIÓN

En la actualidad, para el desarrollo del software se requiere de una mayor productividad y calidad tanto de los productos como de los procesos involucrados, así como la gestión de los mismos mediante la planificación, las estimaciones, el control y el seguimiento de las tareas en curso; es aquí donde la Ingeniería de Software juega un papel importante en el desarrollo de aplicaciones, pues es una disciplina que se encarga del manejo de proyectos informáticos.

I. Ingeniería de software

Una definición precisa aún no ha sido contemplada en los diccionarios, sin embargo se pueden citar las enunciadas por algunos de los más prestigiosos autores[1]:

- Ingeniería de software es el estudio de los principios y metodologías para el desarrollo y mantenimiento de sistemas software (Zelkovitz, 1978).
- Ingeniería de software es la aplicación práctica del conocimiento científico al diseño y construcción de programas de computadora y a la documentación asociada requerida para desarrollar, operar y mantenerlos. Se conoce también como desarrollo de software o producción de software (Bohem, 1976).
- Ingeniería de software trata del establecimiento de los principios y métodos de la ingeniería a fin de obtener software de modo rentable, que sea fiable y trabaje en máquinas reales (Bauer, 1972).
- Es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software; es decir, la aplicación de la ingeniería al software (IEEE, 1993).

En la Ingeniería se involucran actividades como el análisis previo de la situación, el diseño del proyecto, el desarrollo del software, las pruebas necesarias para confirmar su correcto funcionamiento y la implementación del sistema.

En esta disciplina surge un rol importante: el "Ingeniero de software", quien se encarga de crear aplicativos informáticos que den solución a problemas de información y automatización, mediante la gestión, el desarrollo, la operación, el mantenimiento, la adquisición, la utilización y reutilización de servicios y productos de software en general.

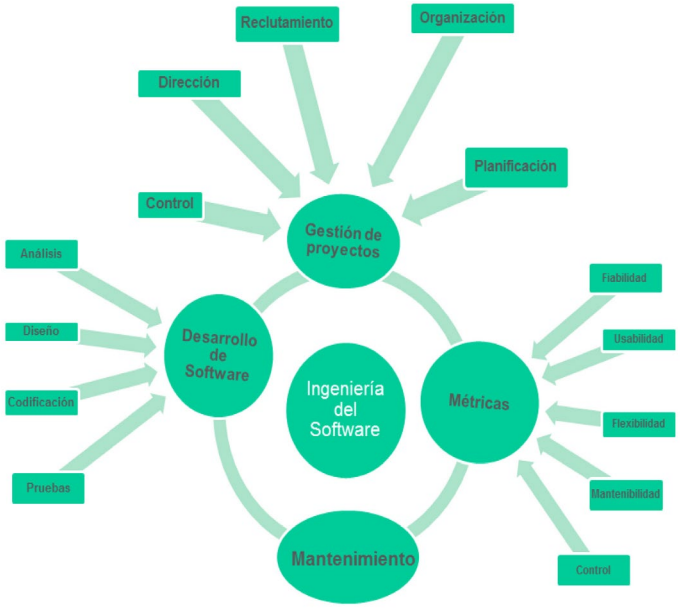

Figura 1. Ingeniería de software [10]

Para la creación de software, el "Ingeniero del software" genera modelos sistémicos aplicando métodos, herramientas y técnicas computacionales que le permiten construir paquetes informáticos de acuerdo con los requerimientos y las necesidades de los clientes en los diferentes contextos nacional e internacional.[4]

II. Gestión de proyectos

La gestión del proyecto de software es el primer nivel del proceso de ingeniería de software, porque cubre todo el proceso de desarrollo. Para conseguir que un proyecto de software sea fructífero se deben tener en cuenta:

- El ámbito del trabajo a realizar.
- Los riesgos en los que se puede incurrir.
- Los recursos requeridos.
- Las tareas a llevar a cabo.
- El esfuerzo (costo) a consumir.
- El plan a seguir.

El objetivo de la gestión de proyectos es canalizar el trabajo de los desarrolladores de forma eficiente y productiva, de tal manera que conduzca al éxito del proyecto. El éxito de los proyectos incluye la gestión activa de los procesos (gestión de proyectos), a fin de cumplir exitosamente con los requisitos de los interesados como lo es el cliente. Para entender la gestión de proyectos informáticos en la Ingeniería de Software se debe tener claro los siguientes términos:

Gestión

Son todas las actividades y tareas ejecutadas por una o más personas con el propósito de administrar (planificar y controlar) las actividades para alcanzar un objetivo, mejorar los resultados o completar una actividad.

Proyectos

Es la integración de una serie de procesos y actividades haciendo uso de una metodología con el fin de lograr objetivos y metas de la manera más eficiente y efectiva. En cualquier tipo de proyecto se debe realizar gestión (planificación y seguimiento durante la duración del proyecto) de los recursos que se tienen disponibles como son dinero, tiempo, personas, herramientas, entre otros.

A. Control

El objetivo de aplicar control en la gestión de proyectos es mantener alineado el proyecto con los objetivos establecidos en su inicio. El control del proyecto se debe realizar en tiempo, recursos, costes, alcances y demás procesos que pueden impactar positiva o negativamente el proyecto.

La importancia de tener control es que quien está a cargo de liderar el proyecto puede saber el estado actual del mismo y, en caso de que algo ande mal, puede tomar las acciones correctivas pertinentes.

B. Dirección

La dirección de proyectos informáticos está compuesta por una serie de actividades las cuales el director del proyecto debe dirigir o gestionar, como:

- Tomar decisiones.
- Impartir instrucciones.
- Liderar grupos (coordinación de personas).
- Asumir compromisos.
- Supervisar el cumplimiento de las actividades o tareas asignadas.
- Adoptar medidas para la corrección de desviaciones.
- Programar las actividades a desarrollar y asignar los recursos necesarios.
- Gestionar información y comunicación necesarias para el proyecto.

4 http://www.eam.edu.co/site/presentacioningenieria-software.php

- Gestionar compras y contratos asociados al proyecto.

C. Reclutamiento

Es un proceso en el cual se hace una búsqueda del personal necesario para poder llevar a cabo la realización del proyecto. De este proceso de selección dependerá en gran parte el éxito o el fracaso de los proyectos, razón por la cual se debe buscar personas competentes y altamente calificadas en las funciones que van a desempeñar y poder así crear grupos de trabajo multidisciplinarios.

D. Organización

La organización de los proyectos informáticos se encarga de planificar las actividades de tal forma que facilite la realización y el control de cada una de las tareas generadas. Además la organización ayuda a definir los límites de autoridad y la asignación de obligaciones y responsabilidades de cada uno de los integrantes del equipo.

E. Planificación

El objetivo de la planificación del proyecto de software es proporcionar un marco de trabajo que permita al gerente o director estimar recursos, costos y actividades o tareas de trabajo. Además, las estimaciones deben intentar definir los escenarios de mejor y peor caso de modo que los resultados del proyecto se puedan acotar.

III. Metodologías de desarrollo de software

De manera general, una metodología de desarrollo de software describe el marco de trabajo realizado para estructurar, planificar y controlar el desarrollo de los sistemas de información.

Las metodologías de desarrollo de software han sido desde los inicios de la construcción del software uno de los puntos principales de estudio dentro del ciclo de vida de desarrollo de software. Las evoluciones en las metodologías de desarrollo del software han traído consigo nuevos retos y formas de pensar alrededor de la problemática que se ha generado. De igual forma, se han introducido fortalezas y debilidades alrededor del desarrollo del software.

Uno de los puntos más influyentes en el cambio y la evolución de las metodologías de desarrollo de software es cómo los sistemas de información y los dominios de tecnología de información están avanzando rápidamente con respecto a las tecnologías disponibles, y atado a esto, la demanda de aplicaciones con nuevos objetivos y propósitos, además de los entornos en los que son ejecutadas y el volumen de usuarios que demandan mayor cuidado, no solo en los requerimientos funcionales sino también en los no funcionales o atributos de calidad.

Estas evoluciones se han convertido en un reto para los profesionales de la ingeniería de software para mantener el ritmo de las tecnologías de la información disponibles y su aplicación en diversos dominios.

Dentro de las metodologías de desarrollo del software deben estar consideradas las distintas áreas de conocimiento del ciclo de vida del software. La siguiente figura muestra las principales áreas:

A. Requerimientos de software

Se puede definir un requisito de software como una propiedad que debe ser realizada por el software desarrollado o modificado para resolver un problema particular; teniendo en cuenta que el problema puede ser una necesidad del usuario en cuanto a la automatización de una tarea de quien va a utilizar el software, el apoyo en un proceso de negocio, la corrección de un defecto en el software, o para controlar un dispositivo, etc. Lo anterior hace que los requisitos de software en muchas ocasiones sean algo complejo porque pueden ser originados por diferentes usuarios y a diferentes niveles dentro de la organización dependiendo del entorno donde se use el sistema.

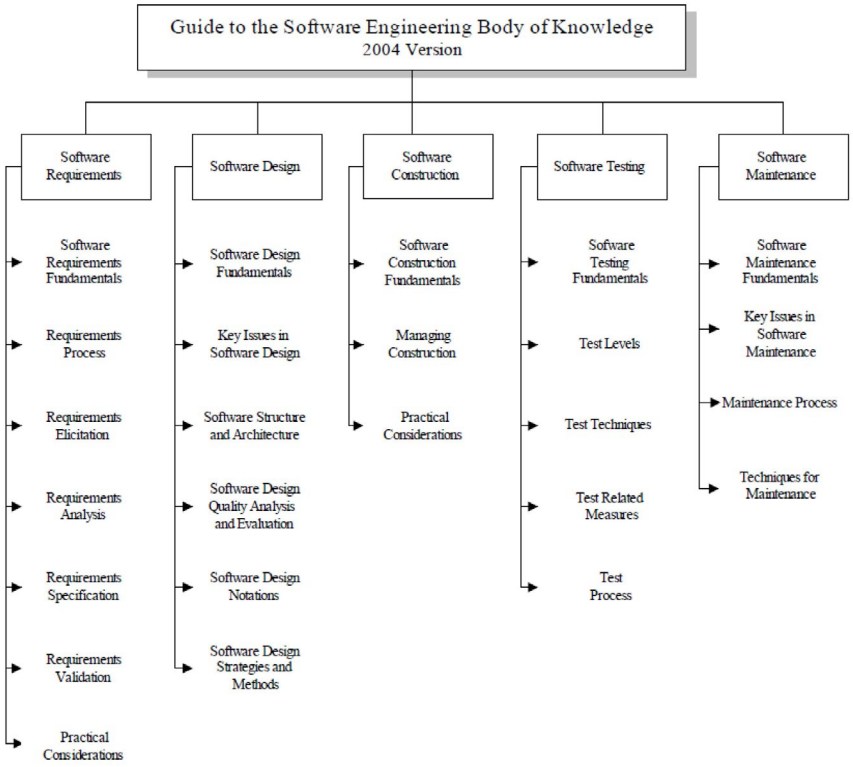

Figura 2. Guide of the software engineering body of knowledge

Una de las características fundamentales de los requisitos de software es que sean verificables y esto hace que en algunas ocasiones esta verificación sea costosa.

Los requisitos de software también tienen otros atributos, adicionales a los de comportamiento, y que permiten una clasificación, por ejemplo para su atención por áreas de desarrollo. En muchos casos estos atributos permiten realizar una priorización de acuerdo a los que tengan mayor valor para el usuario o sean mandatorios. Otro ítem importante dentro de los requerimientos es la relación o dependencia de unos con otros indicando además la secuencia de implementación.

A continuación se relacionan los aspectos más relevantes en cuanto a requerimientos de software:

- *Parámetros de producto y parámetros de proceso*. Los parámetros del producto son requisitos a desarrollar; por ejemplo, una funcionalidad o validación del sistema, y los parámetros del proceso son restricciones sobre el desarrollo del software; por ejemplo, en cuanto a la tecnología a usar.
- *Requerimientos funcionales y no funcionales*. Funcionales son los que describen lo que el sistema debe realizar, también conocidos como capacidades del sistema, y no funcionales son los que están dados para limitar la solución y por las restricciones o atributos de calidad como: usabilidad, seguridad, desempeño, integridad, interoperabilidad, fiabilidad, entre otros.
- *Propiedades emergentes*. Hace referencia a requerimientos que no pueden ser abordados por un solo componente del sistema pero que para satisfacción del usuario dependen de cómo inter-operan todos los componentes de software; por ejemplo, una respuesta a una solicitud ingresada por un usuario. Las propiedades emergentes dependen totalmente de la arquitectura del sistema.
- *Requerimientos cuantificables*. Este aspecto indica que el requerimiento debe expresarse lo más claro posible, de tal manera que no refleje

ambigüedad; por ejemplo: "que el software sea fácil de usar", esta definición no está indicando claramente lo que se está esperando en la usabilidad. Esta característica se requiere especialmente para cubrir los requerimientos no funcionales; por ejemplo: número de transacciones por segundo, nivel de seguridad requerido, tiempo de respuesta esperado, etc.

- *Requerimientos del sistema y requerimientos de software*. En este punto, se cita la definición: "*an interacting combination of elements to accomplish a defined objective. These include hardware, software, firmware, people, information, techniques, facilities, services, and other support elements"*, según lo definido por el Consejo Internacional de Ingeniería de Sistemas, International Council on Systems Engineering (INCOSE00)[8]. Dada esta definición, se debe aclarar que existen diferencias entre los requerimientos del sistema —como requisitos del sistema como un todo— y los requerimientos de software son derivados de los requisitos del sistema.

B. Diseño de software

En términos generales, se puede describir el diseño de software como un mecanismo de resolución de problemas. Este diseño debe contemplar objetivos, limitaciones, alternativas, representaciones y soluciones. El diseño de software es considerado un proceso de dos pasos:

- *El diseño arquitectónico*, describe cómo está formado el software a través de sus componentes y sus interrelaciones (arquitectura de software).
- *El diseño detallado*, el cual describe el comportamiento específico de esos componentes.
- Existen unos principios o unas técnicas de diseño de software como son:
- Abstracción.
- Cohesión y acoplamiento.
- Descomposición y modulación.
- La encapsulación / ocultar información.
- Separación de interfaz e implementación.

Suficiencia, completitud y primitivismo.

Existe una serie de aspectos fundamentales que deben considerarse en el diseño y que están atados a la calidad del software, como por ejemplo el rendimiento. Entre las propiedades que deben contemplarse en el diseño se encuentran:

- Concurrencia.
- Control y manejo de eventos.
- Distribución de componentes.
- Control de excepciones, manejo de errores y tolerancia a fallos.
- Interacción y presentación.
- Persistencia de datos.

C. Construcción de software

La construcción del software hace referencia a la creación detallada de trabajo y sentido del software a través de la combinación de la codificación, la verificación, las pruebas unitarias, las pruebas integrales y la depuración.

El área de conocimiento de la construcción de software está vinculada a las demás áreas de conocimiento del ciclo de vida del software; sin embargo, está relacionada con más fuerza al diseño de software y a las pruebas del software.

La construcción de software normalmente produce muchos elementos de configuración que deben ser administrados dentro de un proyecto de software, como por ejemplo: código fuente, contenido, casos de prueba, etc.; y por esto también está relacionada estrechamente con el área de gestión de la configuración del software. También está relacionada con la gestión de proyectos, en la medida en que la gestión de la construcción puede presentar retos considerables.

Los fundamentos o principios la construcción del software incluyen:

- Minimizar la complejidad.
- Anticiparse al cambio.
- Construcción para la verificación.
- Estándares en construcción.

D. Pruebas de software

En el área de conocimiento de las pruebas de software debe diferenciarse entre un mal funcionamiento del sistema —conocido como falla o defecto— y un efecto no deseado que se observa en la utilización del sistema.

Una prueba puede ser considerada como una observación de una serie de ejecuciones del programa.

En las pruebas cuyo objetivo es la identificación de defectos, se dice que esta es exitosa si el resultado de su ejecución es que el programa falle, pero si las pruebas son para demostrar que el software cumple con las especificaciones entonces los casos de prueba son exitosos si no se presentan fallas en su ejecución.

Las pruebas de software deben ser realizadas en diferentes niveles a lo largo de los procesos de desarrollo y mantenimiento de software. Tres fases de pruebas deben ser consideradas en el proceso de desarrollo de software: las pruebas unitarias, las pruebas de sistema y las pruebas integrales.

Los objetivos de las pruebas pueden estar enfocados en comprobar distintas propiedades de un sistema; por ejemplo los casos de prueba pueden estar diseñados para verificar que las especificaciones funcionales hayan sido implementadas correctamente, estas se denominan pruebas de conformidad.

Sin embargo, el objetivo de las pruebas puede estar encaminado a garantizar ciertos comportamientos del sistema determinados por requerimientos no funcionales o atributos de calidad; en este caso, la prueba puede estar enfocada a garantizar que un sistema soporte cierto número de usuarios concurrentes realizando transacciones.

Dado lo anterior, el objetivo de las pruebas determina el diseño de los casos de prueba que se deban plantear.

E. Mantenimiento de software

Un proyecto de software se dice que no termina con la entrega del producto, para algunos autores realmente es ahí donde inicia porque una vez se entrega, comienzan las iteraciones para atender la solución a las fallas que se presenten y las nuevas necesidades que se generen sobre el sistema y la evolución del mismo, dada por los nuevos requerimientos del usuario o por la obsolescencia tecnológica que un software pueda llegar a presentar a futuro.

El mantenimiento del software se define en la Norma IEEE 1219, como la modificación de un producto de software después de la entrega para corregir los fallos, para mejorar el rendimiento u otros atributos, o para adaptar el producto a un entorno modificado. El estándar también aborda las actividades de mantenimiento antes de la entrega del producto de software, pero solo en un apéndice de la información del estándar[8].

F. Gestión de la configuración del software

Una de las áreas de conocimiento del ciclo de desarrollo de software que tiene una gran relevancia es la gestión de la configuración del software.

Gestión de la configuración (CM) es la disciplina de identificar la configuración de un sistema en puntos distintos en el tiempo con el propósito de controlar sistemáticamente cambios en la configuración y el mantenimiento de la integridad y la trazabilidad de la configuración de todo el ciclo de vida del sistema (Ber97). Se define formalmente (IEEE610.12-90) como "una disciplina que aplica la dirección técnica y administrativa y la vigilancia para: identificar y documentar las características funcionales y físicas de un elemento de configuración, controlar los cambios de esas características, registrar y reportar el proceso de cambios y el estado de implementación, y verificar el cumplimiento de los requisitos especificados."[8]

La gestión de la configuración del software para un mejor logro de su objetivo debe apoyarse en herramientas especializadas en esta área, además de la correcta definición de una estrategia de configuración que permita garantizar:

- La sincronización de los elementos de configuración creados por los diferentes desarrolladores.
- Evoluciones de los componentes sobre versiones estables y oficiales.
- Manejo de líneas base del software.
- Manejo de ramas de configuración del software para gestión de proyectos de mayor duración al mismo tiempo que la sincronización de cambios en los elementos por requerimientos del día a día garantizando los Merge en las diferentes versiones de los componentes.
- La promoción de los cambios para generar versiones en los diferentes ambientes de trabajo del software: desarrollo, pruebas, preproducción y producción.

Teniendo en cuenta que las metodologías de desarrollo del software han ido evolucionando en la medida en que el software evoluciona, también se debe considerar que gran parte de la evolución de las metodologías está dada por los avances en la tecnología y por los entornos cambiantes del negocio que cada vez demandan requerimientos cuyo ciclo de vida sea más corto para poder reaccionar de forma rápida ante la competencia, es por esto que dentro de las evoluciones de las metodologías de desarrollo de software surgen las metodologías ágiles.

G. Metodologías ágiles de desarrollo de software

Una metodología ágil de desarrollo de software es un método de ingeniería del software basado en el desarrollo iterativo e incremental, en el cual los requerimientos y soluciones evolucionan mediante la colaboración de grupos auto-dirigidos y multidisciplinarios.

Los valores de una metodología de desarrollo ágil son:

- Individuos e interacciones en lugar de procesos y herramientas.
- Software funcionando en lugar de documentación detallada.
- Colaboración hacia el cliente en lugar de negociación del contrato.
- Responder al cambio en lugar de seguir un plan.

Una de las metodologías ágiles de desarrollo es precisamente Scrum, esta palabra no es una sigla, el término se originó de una jugada de Rugby, cuyo estudio dio origen al nombre de la metodología.

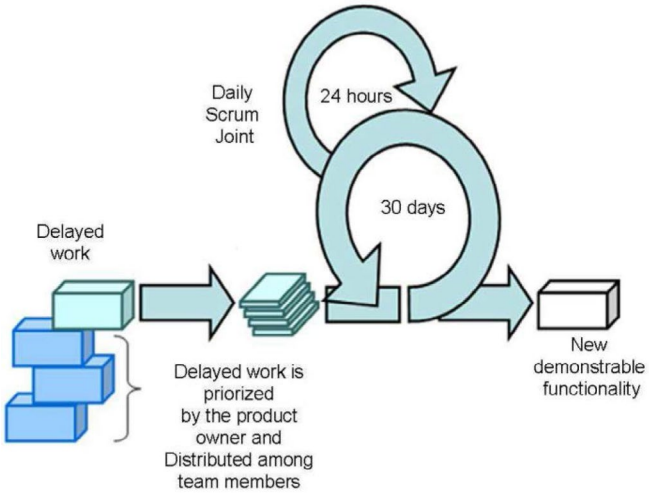

Figura 3. SCRUM, un ejemplo de una metodología ágil

¿Qué es Scrum?

Es una metodología ágil y flexible para gestionar el desarrollo de software, cuyo principal objetivo es maximizar el retorno de la inversión para una empresa (ROI). Se basa en construir primero la funcionalidad de mayor valor para el cliente y en los principios de inspección continua, adaptación, auto-gestión e innovación.

Scrum es una metodología que ha sido adoptada principalmente por compañías grandes y que ha dado lugar a cambios en las prácticas, como por ejemplo la duración de un sprint, que la metodología sugiere sea entre una y cuatro semanas; de acuerdo a la experiencia en estas compañías se ha modificado a seis semanas la entrega del conjunto de requerimientos funcionando en el software[9].

IV. Métricas del software

Un concepto importante dentro de la ingeniería del software es las métricas, por cuanto el avance en dicha disciplina paulatinamente ha buscado darle más relevancia al tema de calidad. Una métrica de software relata de alguna forma las medidas individuales sobre algún aspecto (por ejemplo: el número medio de errores encontrados por revisión o el número medio de errores encontrados por persona y hora en revisiones)[2].

Es importante mencionar el estándar ISO 9126 que se usa para la evaluación de software. Dentro de dicho estándar se encuentran dos tipos de métricas, a saber: internas y externas. Las internas son las que no dependen de la ejecución del software (medidas estáticas), mientras que las externas son las que se aplican al software en ejecución[3]. Es importante implementar métricas, pues lo que se busca es evaluar la calidad y ello puede hacerse de una mejor manera a través de mediciones objetivas.

En un contexto como el colombiano, en el cual todavía en algunos escenarios se vislumbra el desarrollo de software como una actividad 'artesanal' y no se tiene control de aspectos de calidad, es importante poder involucrar mediciones que permitan determinar de una forma objetiva los aspectos en los cuales se está fallando a la hora de desarrollar el software. En cuanto a las métricas de software se refiere, éstas buscan abarcar un aspecto fundamental: controlar la calidad del software, para ello se sugieren definiciones y medidas para algunos aspectos relevantes[4]:

- Corrección: es el grado en el que el software lleva a cabo su función requerida. La medida más común de corrección es defectos por KLDC (miles de líneas de código), donde un defecto se define como una falta verificada de conformidad con los requisitos.
- Facilidad de mantenimiento: es la facilidad con que se puede corregir un programa en caso de encontrar un error, o mejorarse en caso de un cambio de requisitos. Una simple métrica orientada al tiempo es el tiempo medio de cambio (TMC), es decir el tiempo que se tarda analizar la petición de cambio, diseñar una modificación adecuada, implementar el cambio, probarlo y distribuir el cambio a todos los usuarios.
- Integridad: este atributo mide la capacidad de un sistema para resistir ataques (accidentales o planeados) contra su seguridad. El ataque puede ir dirigido a programas, datos o documentos. Para este atributo se tiene una medida definida de la siguiente manera:

$$\text{integridad} = \sum [(1 - \text{amenaza}) \times (1 - \text{seguridad})]$$

Figura 4. Definición de Integridad

Amenaza se define como la probabilidad de que un ataque de un determinado tipo ocurra en un determinado tiempo. La seguridad es la probabilidad de que se pueda repeler el ataque de un determinado tipo.

- Facilidad de uso: es un intento de cuantificar qué tan 'amigable' puede ser el producto de software con el usuario. Se puede medir en función de cuatro características, a saber: (1) habilidad intelectual o física requerida para aprender el sistema; (2) tiempo requerido para llegar a ser moderadamente eficiente en el uso del sistema; (3) aumento neto en productividad (sobre el enfoque que el sistema reemplaza), medida cuando alguien utiliza el sistema moderada y eficientemente; y (4) valoración subjetiva (a veces obtenida mediante un cuestionario).

Estos elementos a evaluar permiten mejorar ostensiblemente la calidad del software; es importante que se pueda evaluar calidad por cuanto no se puede percibir la calidad simplemente a través de elementos de subjetividad de los usuarios, sino que se debe buscar que sean las mediciones las que arrojen los resultados pertinentes de forma objetiva.

V. Métricas en el contexto colombiano

Hay un mercado creciente, apoyado en gran medida por políticas gubernamentales que buscan potencializar el acceso a las TIC en ámbitos como el educativo o la conectividad para regiones apartadas, entre otros[5]. Si bien en este contexto de oportunidades para el crecimiento tecnológico surgen grandes oportunidades para los profesionales en diversos roles de la ingeniería de software, también se generan grandes retos y una responsabilidad que consiste en que el avance tecnológico sea de calidad, que la tecnología que llegue a cada uno de los ámbitos de la sociedad colombiana sea de gran valor y utilidad.

En el caso concreto de la industria del software se ha visto que muchas veces se desarrolla con poco manejo conceptual de calidad y dando mayor prelación a los tiempos de entrega. En un estudio publicado por la Revista Lasallista de Investigación[6] se pudo constatar (al menos a nivel preliminar) que en la mayoría de casos en empresas del suroccidente del país no se conoce sobre métricas de software, y quienes conocen el concepto no utilizan las métricas para evaluar la calidad ni para ningún otro fin.

Este resultado muestra una tendencia preocupante, más aún si se complementa con una investigación enfocada en empresas de Bogotá[10], pues se encontró que en un 85% de estas empresas encuestadas no existe un grupo de aseguramiento de la calidad.

Estas investigaciones han mostrado de manera evidente que hay mucho por avanzar en los temas de calidad del software, y que si bien en muchas ocasiones se puede llegar a ver en el entorno laboral que se habla de calidad y de su importancia, este concepto no puede quedar solamente en retórica sino que debe ser llevado a la organización de manera real.

La pregunta que surge es: ¿cómo pueden implementarse realmente controles de calidad? La respuesta va de la mano de una implementación de metodologías de desarrollo de software en las organizaciones. Por tomar un ejemplo, en Microsoft Solution Framework uno de los principios es invertir en calidad[11], en principio se establece así de acuerdo a Microsoft Consulting Services:

> *Muchas organizaciones defienden la calidad, a menudo un término vagamente definido, pero carecen de los conocimientos para cuantificar la calidad. La calidad es algo que se debe incorporar proactivamente en el ciclo de vida de la entrega de la solución, simplemente no sucede de forma espontánea*.

Así pues, se tiene que la calidad no se genera de la noche a la mañana ni puede concebirse como un concepto abstracto, las organizaciones deben invertir en ella para lograrla y, de ser posible, siempre debe desarrollarse bajo una metodología. No es conveniente 'casarse' con *x* o *y* metodología, pues no todos los casos son iguales, pero sí es fundamental contar con alguna metodología.

CONCLUSIONES

Algunos de los objetivos de la *Ingeniería de Software* son:

- Mejorar la calidad en el desarrollo de las aplicaciones (del software), suministrando a los desarrolladores las bases necesarias para construir software de alta calidad y eficiente.
- Aumentar la productividad del equipo que conforma el proyecto y en especial de los ingenieros del software.
- Facilitar el control del proceso de desarrollo de software.
- Definir un modelo y una metodología que garanticen la producción y el mantenimiento de los productos de software desarrollados en el plazo fijado y dentro del costo estimado.
- La gestión de proyectos es una disciplina encargada de organizar y administrar los recursos asignados de tal forma que se puede llevar a cabo el proyecto con esos recursos (tiempo, presupuesto, personas, herramientas, etc.).
- En la planificación de los proyectos, se fijan los objetivos a corto, mediano y largo plazo, y la forma como serán alcanzados.
- La organización en la Ingeniería de Software es en la cual los gestores determinan de forma detallada el procedimiento para alcanzar los objetivos. En esta fase se crea la disposición, las relaciones de trabajo y se asigna la persona que liderará la actividad.
- El control implica revisar si la planificación realizada al inicio del proyecto está siendo llevada a cabo y además si los objetivos propuestos se han cumplido. En esta etapa, el líder del proyecto debe ser capaz de realizar las correcciones respectivas.
- El plan de un proyecto se debe adaptar y actualizar conforme avance el proyecto, con el fin de tener una visión clara del estado de este.
- Las metodologías de desarrollo de software han sido construidas o generadas mediante el mejoramiento continuo, no solo demandado por la construcción del software sino además por los avances tecnológicos y las necesidades de las áreas de negocio de las compañías.

El control de calidad de guiado a través de las métricas es muy importante para el desarrollo de software, puesto que permite tener elementos objetivos para determinar si el trabajo se hace o no de una manera apropiada.

En el contexto colombiano hay mucho camino por recorrer, si bien hay elementos iniciales que permiten inferir que dentro de las organizaciones ya hay elementos formales de ingeniería de software, estos avances se quedan cortos en aspectos como implementación de metodologías y manejo de control de calidad.

REFERENCIAS

[1] TechNet–Microsoft. *¿Qué es la ingeniería del software?* Recuperado de: http://social.technet.microsoft.com/Forums/es-ES/7dc2cf80-a6ad-4271-b4db-a1e3edb946fb/-que-es-la-ingenieria-software-

[2] Pressman, R. (2002). *Ingeniería del software. Un enfoque práctico*. México: Mc GrawHill. (p. 54).

[3] Sicilia, M. Á. *Estándar ISO 9126 del IEEE y la mantenibilidad*. Recuperado de: http://garciagregorio.webcindario.com/ms/iso9126.pdf

[4] Pressman, R. (2002). *Ingeniería del software*. Un enfoque práctico. México: Mc GrawHill. (p. 63).

[5] Ministerio de Tecnologías de la Información y las Comunicaciones de Colombia. *Proyecto Nacional de Fibra Óptica*. Recuperado de: http://www.mintic.gov.co/portal/vivedigital/612/w3-propertyvalue-647.html

[6] Moreno, J. J.; Bolaños, L. P.; Navia M. A. (2010). Un acercamiento a las prácticas de calidad de software en las MiPyMESPS del suroccidente colombiano. *Revista Lasallista de Investigación*, 7(1). Recuperado de: http://www.scielo.org.co/scielo.php?pid=S1794-44492010000100003&script=sci_arttext

[7] Dahiya, D.; Jain, P. (2010). *Enterprise Systems Development: Impact of Various Software Development Methodologies*.

[8] IEEE Computer Society. (2004). *Guide to the Software Engineering Body of Knowledge, A project of the IEEE Computer Society Professional Practices Committee*. Recuperado de: http://www.math.unipd.it/~tullio/IS-1/2007/Approfondimenti/SWEBOK.pdf

[9] Heikkilä, V.; Paasivaara, M.; Lassenius, C. (2013). *ScrumBut, But Does it Matter? A Mixed-Method Study of the Planning Process of a Multi-team Scrum Organization*.

[10] Abuchar, A.; Cárdenas, B.; Alfonso, D. (2012). Observatorio de prácticas de desarrollo de software en MinPyme y pymes de Bogotá. *Revista Científica, 15*. Recuperado de: http://revistas.udistrital.edu.co/ojs/index.php/revcie/article/view/3949

[11] Microsoft (2013). Descripción general de Microsoft Solution Framework. Recuperado de: http://msdn.microsoft.com/es-es/library/jj161047.aspx

[12] Pérez-Torres, J. A.; Mejía, M. (2005). *Software Development Using Agile Methodologies: An Airline Case*.

[13] Rasmin, R.; Paige, R. F. (2010). *Iterative Criteria-Based Approach to Engineering the Requirements of Software Development Methodologies*.