INTRODUCCIÓN A LA INGENIERÍA DE SOFTWARE ¿QUÉ ES?

Bellanich Nicolás
e-mail: nicolasbellanich@gmail.com
Campos Sebastián e-
mail:sebacampos169@gmail.com
García Rodrigo
e-mail:rodrigo.09garcia@gmail.com
Pascuali Lisandro
e-mail:lpascuali97@gmail.com
Tarasconi Gastón e-
mail:gaston.tarasconi@gmail.com

Universidad Tecnológica Nacional - Ingeniería en Sistemas de Información - Cátedra de Ingeniería de Software - Curso 4K4

RESUMEN: *En este informe realizaremos una introducción a la ingeniería de software. Ahondaremos en temas como el surgimiento, la definición y la actualidad de aquello que llamamos Ingeniería de Software. La creación de software es una actividad humana, y como tal se ha ido transformando. Como práctica, es relativamente joven, pero al estar ligada a los avances tecnológicos y a procesos de negocios, debió adaptarse a un ritmo de cambio vertiginoso. De software creado por una persona, como un trabajo más tedioso que creativo, con requerimientos fijos y no adaptables, a equipos de trabajo interdisciplinarios con habilidades más allá que las técnicas, que intentan desarrollar software funcional y con requerimientos cambiantes. Este cambio histórico, estuvo acompañado por cambios en las distintas disciplinas que integran la ingeniería de software como actividades de gestión de proyectos, y desarrollo de métodos y teorías de apoyo a la producción de software.*

PALABRAS CLAVE: software, ingeniería, actualidad.

1 INTRODUCCIÓN

Para introducirnos en el conocimiento de la ingeniería de software es importante comprender el pasado de la actividad para entender de dónde surge y aquello de lo que está formada. Podemos decir que la historia se comparte con la del desarrollo de software como actividad en sí. Inicialmente el software era construido por una única persona en quien se concentraban todos los conocimientos. No existía sistematización, existían cambios de requerimientos que no se podían afrontar y eran procesos tediosos más que otra cosa. Con el paso del tiempo, se adquirió la capacidad sistémica de creación de los productos de hardware quienes tenían un proceso mejor definido sobre etapas o procesos a seguir y que permitía la participación de grupos integrados, división de tareas y mejor distribución de la complejidad. También surgió la necesidad de crear grupos de trabajo más veloces, flexibles y mejor organizados, es decir mejorar la segunda

etapa que había tenido el software. Y allí surge el concepto de Ingeniería de software que no solo contempla procesos técnicos de desarrollo de software si no también actividades como gestión de proyectos de software y de desarrollo de herramientas, métodos y teorías de apoyo a la producción del software. La idea de este reporte es avanzar sobre estos conceptos que definen y distinguen a la Ingeniería de software.

2 UN POCO DE HISTORIA

La historia de la ingeniería del software es paralela a la historia del software.

Durante la década de los cincuenta, se aplica al desarrollo de software el mismo proceso que al desarrollo de hardware, se usa el método científico para aprender a través de la experiencia, el software se consideraba como un producto añadido y la programación de computadores era un arte para el que no existían métodos sistemáticos. El desarrollo de software se realizaba sin ninguna planificación una solo persona lo escribía, lo ejecutaba y, si fallaba, lo depuraba. El diseño era un proceso implícito, que se realizaba en la mente del programador y la documentación no existía.

En la década de los sesenta con el surgimiento de la multiprogramación y los sistemas multiusuario se introducen nuevos conceptos de interacción hombre – máquina, los sistemas en tiempo real recogían, analizaban y transformaban datos de múltiples fuentes y apoyaban la toma de decisiones. Como consecuencia de esto nació la primera generación de sistemas de gestión de bases de datos. Esta era se caracteriza por la aparición del software como producto y el nacimiento de las casas de software donde se producían programas de miles de líneas de código fuente que tenían que ser corregidos cuando se detectaban fallas y modificados cuando cambiaban los requisitos, entonces se fomentaba el proceso de desarrollo de software tipo *codifica y corrige*. Como consecuencia de esta práctica y debido a que los proyectos de software presentaban muchas fallas (dado que los desarrollos terminaban sobrepasando el tiempo y costos estimados al inicio del proyecto), no se obtenían los resultados esperados y el software era poco flexible. Aquí tuvo lugar un fenómeno conocido como la “crisis del software”, que dio pie a que en 1968 se realizara la primera conferencia sobre desarrollo de software en Múnich, financiada por la OTAN. Allí la “crisis del software” se convierte en el tema central y se utilizó por primera vez el término "ingeniería del software" para describir el conjunto de conocimientos que existían en un estado inicial. Así pues, nace formalmente la rama de ingeniería de software.

Para la década de los setenta la evolución de los sistemas distribuidos, las redes de área local y global y la creciente demanda de acceso instantáneo a los datos supuso una fuerte presión sobre los desarrollos de software incrementando notablemente la complejidad de los sistemas informáticos. Esto incidió en la identificación de las diferentes fases del desarrollo de software como requerimientos, análisis, codificación y pruebas. Y aquí fue cuando se introdujo la programación estructurada y los métodos formales para especificar software. Se identificaban principios de diseño, como modularidad, encapsulamiento, abstracción de tipos de datos, acoplamiento débil y alta cohesión, se publicó el modelo en cascada y se definieron los conceptos de verificación y validación.

La década de los ochenta se caracterizó por la productividad y escalabilidad de sistemas y equipos de desarrollo, la industria del software es la cuna de la economía del mundo donde las técnicas para el desarrollo de software de cuarta generación (4GLs) cambiaron la forma en que se construían los programas para incrementar la productividad. Se introdujo la tecnología de programación orientada a objetos a través de múltiples lenguajes de programación desplazando los enfoques de desarrollo

tradicionales. A finales de esta década se creó el primer modelo de madurez de capacidad de procesos (SW-CMM).

En los noventa y el nuevo siglo, la concurrencia (paralelismo y distribución) adquirió mayor importancia, la orientación a objetos se extendió a las fases de análisis y diseño, se implementó el lenguaje de modelado (UML) y se generó el primer proceso comercial de desarrollo orientado objetos (RUP). Los diseñadores y los arquitectos de software iniciaron su experiencia a través de patrones de diseño y de arquitectura. Se definió el modelo en espiral para el desarrollo basado en el análisis de riesgos y el desarrollo de software iterativo e incremental. Para esta década, el software era privado entonces surgió la necesidad por parte de un grupo de programadores de crear proyectos que impulsaran la creación de software libre y de código abierto. La usabilidad de sistemas se convirtió en el foco de atención e investigación, el software empezó a ocupar la posición crítica en el mercado competitivo y en la sociedad Web. **[4]**

2.1 NO SILVER BULLET

Durante décadas, resolver la crisis del software desencadenó en que compañías e investigadores produjeran más y más herramientas software. Cada nueva tecnología o práctica que apareció entre 1970 y 1990 fue tratada como una "*bala de plata*" (en inglés, *silver bullet*) que solucionaría la crisis del software.

En 1986, Fred Brooks publicó el artículo *No Silver Bullet*, argumentando que ninguna tecnología o práctica por sí misma podría mejorar en un diez por ciento la productividad en los siguientes diez años. El debate sobre las *balas de plata* continuó durante la siguiente década, dando lugar a numerosas interpretaciones sobre el artículo de Brooks.

Los defensores de lenguajes como Ada, o de los procesos software continuaron apostando por que su tecnología sería la que solucionaría la crisis. Sin embargo, hubo gente que interpretó el hecho de que no se encontrara una solución única y efectiva al cien por cien como un fracaso de la ingeniería del software.

Si bien es cierto que la búsqueda de una única solución no funcionó, también había que ser conscientes de que tampoco existían *balas de plata* en ninguna otra profesión. Así, con el transcurso de los años, casi todo el mundo aceptó que no se encontraría ninguna *bala de plata*, pero se tomó esto como una prueba de que la ingeniería del software finalmente había madurado y que los proyectos debían tener éxito gracias al trabajo duro y al esfuerzo. El campo de la ingeniería del software es demasiado complejo y diverso para que una única solución resuelva todos los problemas, pero el conjunto de todas las prácticas que surgieron y de las que surgen hoy en día son las que, bien aplicadas, permiten que la ingeniería del software desarrolle productos de calidad. **[5]**

3 ¿QUÉ ES SOFTWARE?

Ya nos introdujimos en el contexto histórico de la ingeniería de Software, pero antes de seguir desarrollando este concepto, creemos necesario definir qué es software. Normalmente se piensa que cuando se habla de software estamos hablando de programas, juegos, aplicaciones en los celulares; sin embargo, el software abarca mucho más que eso. En la actualidad estamos en contacto con software todo el tiempo, en un auto, en un avión, o en una simple cafetera. En los últimos años el software ha evolucionado de tal manera que hasta se encuentra en situaciones que si este llegara a fallar pueden ocurrir sucesos realmente críticos. Cuando hablamos de software, solemos asociar este concepto directamente al código, o a los programas o aplicaciones. Y es cierto, un software es un set de programas, pero es

también la documentación que lo acompaña, los archivos de configuración, y muchos otros artefactos que complementan el código que vemos funcionando. Es decir, el software abarca una gran cantidad de componentes y artefactos, y esto es la muestra de que se trata de algo complejo.

5 ACTIVIDADES PARA LA CREACIÓN DE SOFTWARE

Todo proyecto de software se desencadena por alguna necesidad de negocios: la de corregir un defecto en una aplicación existente, la de adaptar un sistema heredado a un ambiente de negocios cambiante, la de ampliar las funciones y características de una aplicación ya existente o la necesidad de crear un producto nuevo.**[2]**

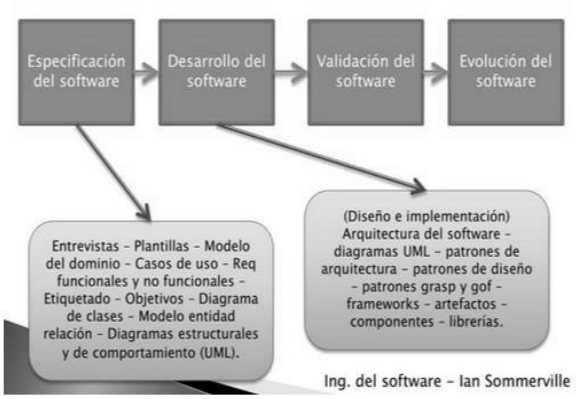

Figura 1. Actividades.

En la creación de un nuevo producto se llevan a cabo una serie de actividades fundamentales que son llevadas a cabo por los ingenieros de software. Estas actividades son:

- Especificación del software: Donde el cliente define el producto que quiere asi como tambien las restricciones
- Desarrollo del software: Diseño y programación del software.
- Validación del software: Se valida que el software sea realmente el que el cliente necesita.
- Evolución del software: Actualizaciones y modificaciones que se realizan para adaptarse a nuevos cambios.

5.1 PRODUCTOS DE SOFTWARE

A la hora de crear un nuevo producto, podemos distinguir entre dos tipos:

- Productos genéricos: Son sistemas aislados producidos por una organización de desarrollo y que se venden al mercado abierto a cualquier cliente que le sea posible comprarlo. Un ejemplo de estos pueden ser bases de datos, procesadores de texto, paquetes de dibujo y herramientas de gestión de proyectos.**[1]**
- Productos personalizados: Son sistemas requeridos por un cliente particular. Por ejemplo: sistemas de control para instrumentos electrónicos, sistemas desarrollados para llevar a cabo procesos de negocios específicos y sistemas de control de tráfico aéreo.**[1]**

De acuerdo con estos dos tipos de productos podemos concluir que la principal diferencia que existe es que en los primeros el software está desarrollado y el cliente ya lo compra hecho, con sus múltiples funciones, donde probablemente solo se utilizaran algunas, a diferencia de los productos personalizados donde estos se desarrollan en función de los requerimientos del cliente y de su forma de trabajar, es decir que busca complacer todas las necesidades y adaptarse de la mejor manera a lo que la empresa necesita .

5.2 ATRIBUTOS DE UN BUEN SOFTWARE

¿Por qué detenernos en desarrollar el concepto de software, si nuestra intención es hablar sobre una disciplina? Es que justamente, la Ingeniería de Software, tiene como fin último el desarrollo de un producto de software de calidad.

Entonces, el producto final, es el que termina definiendo que tan buena resultó ser la aplicación de la Ingeniería durante el desarrollo del mismo. Aquí describimos cuatro atributos que todo software debe tener para asegurar ser un software de calidad. Estos atributos no están directamente asociados con lo que el software hace. Más bien, reflejan su comportamiento durante su ejecución y en la estructura y organización del programa fuente y en la documentación asociada.

- **Mantenibilidad:** El software debe escribirse de tal forma que pueda evolucionar para cumplir las necesidades de cambio de los clientes. Este es un atributo crítico debido a que el cambio en el software es una consecuencia inevitable de un cambio en el entorno de negocios.
- **Confiabilidad:** La confiabilidad del software tiene un gran número de características, incluyendo la fiabilidad, protección y seguridad. El software confiable no debe causar daños físicos o económicos en el caso de una falla del sistema.
- **Eficiencia:** El software no debe hacer que se malgasten los recursos del sistema, como la memoria y los ciclos de procesamiento. Por lo tanto, la eficiencia incluye tiempos de respuesta y de procesamiento, utilización de la memoria, etcétera.
- **Usabilidad:** El Software debe ser fácil de utilizar, sin esfuerzo adicional, por el usuario para quien está diseñado. Esto significa que debe tener una interfaz de usuario apropiada y una documentación adecuada.

6 DEFINICIÓN DE INGENIERÍA DE SOFTWARE

Después de haber aclarado ciertos conceptos, y habiendo conocido un poco de la historia, podemos hablar sobre la Ingeniería de Software. La ingeniería de software es un conjunto de disciplinas de la ingeniería que se preocupa de todos los aspectos de la producción de un Software; desde las primeras etapas de la especificación hasta el mantenimiento del sistema.**[1]**

Podemos decir que son las “recetas” o pasos a seguir para crear un software de calidad. Así como los arquitectos realizan planos para construir un edificio, para construir un software también se realizan planos, diagramas, y se llevan a cabo distintos tipos de actividades.

Si nos remontamos al glosario de la IEEE, encontraremos la siguiente definición: La ingeniería de software es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software**. [7]**

6.1 DIFERENCIA INGENIERÍA DE SOFTWARE Y CIENCIA DE LA COMPUTACIÓN

Así como cuando hablamos de software, solemos asociar este concepto a un único aspecto del mismo, e ignoramos la complejidad que abarca; también puede pasarnos que al hablar de Ingeniería de Software, estemos pensando en otras disciplinas que aunque parezcan similares, difieren en varios aspectos. Esencialmente, la ciencia de la computación refiere a teorías y métodos referidos a computadoras y sistemas de software, mientras la ingeniería de software refiere a problemas prácticos de producir software, los ingenieros de software necesitan ciertos conocimientos de ciencia de la computación, pero los usan como herramienta, no como fin en sí mismos. La ingeniería de software es más práctica, muchas soluciones teóricas de las ciencias de la computación muchas veces no son factibles a la hora de implementar una solución de software.

6.2 DIFERENCIA INGENIERÍA DE SOFTWARE E INGENIERÍA EN SISTEMAS

La ingeniería de sistemas refiere a todos los

aspectos que comprenden desarrollo y evolución de sistemas complejos donde el software tiene un papel principal. Comprendiendo desarrollo de hardware, políticas y procesos de diseño y distribución de sistemas, así como la ingeniería de software. Los ingenieros en sistemas están involucrados en la especificación del sistema, en la definición de la arquitectura y en la integración de diferentes partes para crear el sistema final. Aunque están menos relacionados con la ingeniería de los componentes ndel sistema (Hardware, Software, etc.)

7 OBJETIVOS DE LA INGENIERÍA DE SOFTWARE

Es necesario comprender cuáles son los objetivos de esta disciplina. Tal como lo venimos desarrollando hasta aquí, la Ingeniería de Software es una rama que se desprende de la Ingeniería de Sistemas. Y nace justamente, a raíz de la necesidad de aplicar ingeniería no solo al producto, sino también la proceso, considerando que la mejora en la calidad del proceso, resulta en la mejora en la calidad del producto. Por lo tanto, esta disciplina se ocupa de los aspectos técnicos de la tecnología, pero además trata de abarcar también lo concerniente a la gestión de proyectos de desarrollo de software. A continuación, enumeramos una serie de objetivos que creemos que orientan el ejercicio de esa disciplina:

- Diseño de aplicaciones informáticas que se ajusten a las necesidades de las organizaciones.
- Dirección y control en el desarrollo de aplicaciones complejas.
- Intervención en todas las fases del ciclo de vida de un producto, realizando un seguimiento desde el análisis hasta las pruebas y el mantenimiento del producto.
- Estimación los costes de un proyecto y tiempos de desarrollo.
- Dirección de equipos de trabajo de desarrollo software.
- Utilización de procedimientos de calidad en los sistemas, evaluando métricas e indicadores y controlando la calidad del software producido.**[8]**

8 LA PRÁCTICA DE LA ING. DE SOFTWARE

Anteriormente hablamos sobre las actividades fundamentales para la creación de un nuevo producto, ¿pero cómo se incorpora la práctica de la ingeniería de software en la producción de software?

Podríamos decir que la esencia de la práctica de la Ingeniería de Software consiste en cuatro pasos:**[2]**

1. Entender el problema
2. Planear la solución
3. Ejecutar el plan
4. Examinar la exactitud del resultado

Figura 2. Práctica en la ingeniería de software.

De acuerdo a estos pasos tomados de *"Ingeniería de software. Un enfoque práctico"* podemos decir que, en el primer paso, solemos actuar apresuradamente, creyéndonos conscientes del problema que se plantea, cuando en realidad aún nos queda mucho por entender. Aquí, es donde la aplicación de la

Ingeniería de Software nos requiere hacer una pausa para poder indagar de manera más profunda sobre el problema, guiándonos de preguntas como: ¿Quiénes son los participantes? ¿cuáles son los actores que se verán beneficiados con esta solución? ¿Puede fragmentarse el problema? ¿Se puede crear un modelo del mismo, para poder analizarlo en profundidad?

El paso siguiente consiste en planear la solución. Y aquí se vuelve a reafirmar la necesidad de darle importancia al primer paso. Sería una gran pérdida de tiempo y esfuerzo si planeamos una solución para un problema que no es el que en realidad nos interesa resolver. Y en este paso, si queremos poner en práctica la Ingeniería de Software, la forma de proceder es la misma: detenernos, y no lanzarnos de manera repentina a escribir un código, o a crear una solución desde cero. Probablemente la solución que estamos necesitando, ya existe, y hasta tal vez, ya la hemos implementado en proyectos anteriores, para dominios similares. Aquí también se nos sugiere realizar una serie de preguntas: ¿Ha visto antes problemas similares? ¿Hay patrones reconocibles en una solución potencial? ¿Existen elementos reutilizables de una solución conocida?

Si nos tomamos el trabajo de realizar estos dos primeros pasos, nos estamos asegurando una mayor eficacia a la hora de ejecutar el plan de solución. El contar con un plan para la solución, nos permitirá seguir una “carretera” (más allá de las desviaciones que puedan presentarse), garantizando que el trabajo que vayamos realizando nos conduzca a la solución que intentamos alcanzar. A medida que avancemos en la ejecución, también podemos ir haciendo algunas preguntas que nos ayuden a reconocer si vamos por buen camino: ¿Se ajusta la solución al plan? ¿El código fuente puede apegarse al modelo del diseño? ¿Es probable que cada parte componente de la solución sea correcta?

Una vez ejecutado el plan, no podemos asegurarnos que la solución alcanzada sea la correcta. Pero sí es cierto que podemos asegurarnos de contar con una serie de casos de prueba que nos permitan garantizar que nuestra solución sea la correcta. Para esto, es necesario realizar un examen del resultado obtenido, contemplando cada componente y cada situación a la que pueda someterse el software creado.

9 RETOS A LOS QUE SE ENFRENTA LA INGENIERÍA DE SOFTWARE

La ingeniería de software se enfrenta a 5 retos fundamentales, estos son: **[6]**

1. El reto de los sistemas legados.
2. El reto de la heterogeneidad.
3. El reto de la entrega.
4. El reto de la confianza.
5. El reto de la formalidad.

Figura 3. Retos ingeniería de software.

En cuanto al primer reto, hace referencia a aquellos sistemas antiguos que un cliente tiene y estos deben ser mejorados, mantenidos e incluso adaptar un nuevo producto a un sistemas de estas características. A veces estos sistemas contienen una cantidad de información que es vital para la toma de decisiones y reemplazarlo puede ser de mucho esfuerzo y costo.

El reto de la heterogeneidad, hace referencia a que el software debe tener la capacidad y flexibilidad para poder integrarse a sistemas heredados más viejos escritos en diferentes lenguajes de programación.

El reto de la entrega marca a el tiempo como un

factor muy importante. Las grandes empresas que están en constante cambio, deben tener una gran capacidad de respuesta a estos cambios, así como también el software que le da el soporte a las mismas.

En cuanto a el reto de la confianza consiste en desarrollar técnicas que demuestren a los usuarios que pueden confiar en el software.

Por último, el reto de la formalidad consiste en que exista formalidad en el proceso de desarrollo de software.

10 IMPORTANCIA EN LA ACTUALIDAD

La principal importancia de la ingeniería de software es que construyen algo, que antes no existía. Es un campo único que recorre la línea fina entre lo creativo y lo científico, se necesita poder visualizar el producto y luego crearlo.**[3]**

Gradualmente nos estamos volviendo altamente dependientes de la tecnología, la cual es cada vez mejor en todos los aspectos de nuestras vidas, en entretenimiento, estudio, investigación, etc. Todo gira ahora en torno a las aplicaciones de software.

Para ello, los ingenieros de software deben desarrollar su autosuficiencia en el desarrollo de software para ser más capaces de crear aplicaciones útiles para dispositivos inteligentes.

Hoy en día la ingeniería de software cumple un papel muy importante ya que desarrollar un software de mala calidad puede llegar al caso de poner en riesgo la vida de las personas, como por ejemplo, que un airbag de un automóvil no se abra a tiempo, o que algún dispositivo de un hospital no funcione correctamente.

Todo lo que hace que esta era de la humanidad sea espectacular, es la ingeniería de software. Sus ingenieros son el cerebro detrás de la operación exitosa de toda máquina inteligente.

11 CONCLUSIÓN

En este informe se evidencia la importancia del software para nuestras vidas, ya que lo tenemos presente a toda hora en el dia a dia. Dado que el software está desarrollado e integrado en las máquinas para que pueda cumplir con todos los propósitos de los diferentes usuarios. Para ello la ingeniería de software es de gran aplicación y asistencia a la hora de desarrollar un nuevo producto y que este se desarrolle de una manera efectiva y eficiente.

La ingeniería de software es un área nueva dentro de la ingeniería pero el alcance que tiene es extremadamente amplio. Es un campo bastante prometedor, ya que está creciendo de una manera pronunciada y es probable que crezca mucho más rápido que el promedio, en comparación con otras profesiones.

12 REFERENCIAS

Plantilla de informe técnico disponible en: https://www.academia.edu/6936218/PREPARACI%C3%93N_DEL_REPORTE_INFORMATIVO_EN_FORMATO_DE_DOS_COLUMNAS_MANUSCRITO_ESTILO_PAPER_

[1] Ingeniería del software 7Ed. Ian Sommerville.

[2] Ingeniería del software. Un enfoque práctico. Roger S. Pressman.

[3]"*Importance of Software Engineering"*.[En línea]. Disponible en: https://www.castsoftware.com/glossary/importance-of-software-engineering-code-of-ethics-future

[4] William Frasser Acevedo " *Historia y Evolución de la ingeniería del Software*" Escuela de Ciencias Exactas e Ingeniería Universidad Sergio Arboleda.[En línea] Disponible en: https://ingswusa.files.wordpress.com/2014/08/historia-de-ing-software.pdf

[5] "*Ingenieria del Software*" [En línea] Disponible en: https://histinf.blogs.upv.es/2010/12/28/ingenieria-del-software/

[6]*"Ingeniería del Software"*.[En línea] Disponible en: https://issuu.com/ingenieriaarquitecturausat/docs/ingenieria_de_software/15

[7] IEEE Standard Glossary of Software Engineering Terminology ,[En línea] Disponible en: https://ieeexplore.ieee.org/document/159342

[8] Objetivos de Ingeniería del Software – Escuela Técnica Superior de Ingeniería de Sistemas Informáticos – Universidad Politécnica Madrid, [En línea]Disponible en http://www.etsisi.upm.es/estudios/grados/software/objetivos