ESTADO ACTUAL Y ANTECEDENTES. LA CRISIS DEL SOFTWARE

Benavidez Santiago Nicolás - Cravero Matias Agustín - Gómez Aguirre Miguel - Morano Antonella - Rojas Alexis Augusto

UNIVERSIDAD TECNOLÓGICA NACIONAL - FACULTAD REGIONAL CÓRDOBA
INGENIERÍA EN SISTEMAS DE INFORMACIÓN
Ingeniería de Software
Curso 4K4 – Año 2019

Abstract- **Para la sociedad actual, pensar en una vida sin software sería algo totalmente imposible. El software está presente en la realización de la mayoría de las actividades diarias de las personas.**
El presente informe técnico consiste en una investigación detallada y específica de los antecedentes de la Ingeniería del Software y su situación actual.
La investigación se basó en conceptos claves de la historia de la Ingeniería del Software, como sus inicios con la tan conocida "Crisis del Software", los grandes fracasos del software y el surgimiento de nuevas herramientas que cumplieron un papel fundamental en su evolución. Además, se incorporó un marco teórico con definiciones necesarias para abordar los temas mencionados anteriormente.
La metodología que se empleó fue la investigación de distintas fuentes mencionadas en la sección de Bibliografía y la recopilación de citas de autor relevantes para el desarrollo del informe, mencionadas en la sección de Referencias.
Los resultados obtenidos permitirán analizar la importancia de la Ingeniería de software actualmente en los sistemas y el papel fundamental que ejercerá en el futuro.

PALABRAS CLAVE- software, crisis del software, ingeniería de software, evolución del software.

1. INTRODUCCIÓN

"Debe hacerse ingeniería con el software en todas sus formas y a través de todos sus dominios de aplicación." **[1] Ingeniería del software; Un enfoque práctico- Roger S. Pressman**

Esta frase explica que la mejor forma de hacer frente a los retos del software en el siglo XXI (entender el problema planteado, aplicar foco en el diseño debido a su importancia y mantener la calidad de software) es a través de la aplicación de la ingeniería de software.

Friedrich Ludwig Bauer consideró a la ingeniería de software como *"El establecimiento y uso de principios de ingeniería robustos, orientados a obtener económicamente software que sea fiable y funcione eficientemente sobre máquinas reales."* **[2] NATO SOFTWARE ENGINEERING CONFERENCE 1968- Professor Dr. F. L. Bauer**

F. Bauer formuló esta definición debido a que, en ese momento, el término "ingeniería de software" se consideró como una posible solución a todas las dificultades en el desarrollo de sistemas de software lo qué se denominó el fenómeno de la Crisis del Software.

En este informe trataremos los distintos inconvenientes que existían en el desarrollo de sistemas durante la era de la crisis del software, como fue evolucionando la ingeniería de software para dar soporte a los mismos y la situación actual empleando las distintas metodologías que fueron surgiendo.

2. INGENIERÍA DE SOFTWARE

A. ORIGEN

En los años 1968 y 1969, el Comité de Ciencias de la OTAN patrocinó dos conferencias en las cuales muchos creen que fueron los inicios de la ingeniería de software.

Se propuso el enfoque de la ingeniería de software como antídoto para tratar la "Crisis del software", término acuñado para denominar a la era marcada por todos los inconvenientes que se encontraban en el desarrollo de sistemas complejos desde los años 50.

Se mantenía la esperanza de que la ingeniería de software pudiera reducir los costos de desarrollo y mantenimiento de software y lograra conducir a un software más confiable y de calidad.

i. Causas de la Crisis del Software

A finales de los años 60, el potencial de las computadoras comenzó a aumentar de forma considerable, dicho avance ocurrió con gran rapidez a nivel de hardware, pero no sucedió lo mismo a nivel de software, las técnicas de programación se retrasaron demasiado y no tuvieron el potencial necesario para acompañar el crecimiento que estaba transitando los avances de la tecnología.

Además, otro problema adicional fue que muchos programadores no estaban capacitados formalmente, debido a que el software era considerado un arte y no un oficio, por lo que los desarrolladores aprendían sobre la marcha.

Pero Hans van Vliet establece que *"Por el lado de las organizaciones, se llevaron a cabo intentos de solución a estos problemas, que consistían en añadir más y más programadores al proyecto, el enfoque llamado 'million-monkey"* **[3] Software engineering : principles and practice- Hans van Vliet**

Como resultado a todos estos problemas y a las frustradas soluciones por parte de las organizaciones, surgieron problemas en la gestión de proyectos de software tales como:

- Sobrepaso de tiempo planificado
- Mantenimiento casi imposible
- Software no adaptable a los cambios que surgían
- Exceso del presupuesto
- El producto no cubría las necesidades del cliente
- Muchos defectos se detectaban luego de que el software se entregara al cliente

En resumen, se puede decir que el software se desarrollaba con mala calidad, lo que generaba que nunca fuera utilizado, por lo que se incurría en pérdida de tiempo y aumento de costos.

B. ANTECEDENTES:

i. DÉCADA DEL 70

En la década de los 70 las organizaciones empezaron a tomar conciencia de que los costos del software superaban a los costos del hardware. Estos acontecimientos representan la transición hacía lo que conocemos actualmente como ingeniería de software.

La evolución de los sistemas distribuidos y la creciente demanda del software generó una fuerte presión sobre el desarrollo de software incrementando notablemente la complejidad de los sistemas informáticos.

Debido a esto, se identificó el enfoque sistemático que se usa en la ingeniería de software que se conoce como **proceso de software**.

"Un proceso de software es un enfoque adaptable que permite que las personas que hacen el trabajo (el equipo de software) busquen y elijan el conjunto apropiado de acciones y tareas para el trabajo." **[1] Ingeniería del software; Un enfoque práctico- Roger S. Pressman**

Con esta definición, Roger Pressman busca definir que un proceso no debe ser una especificación estricta de cómo realizar un producto, sino más bien un enfoque adaptable al equipo y al proyecto en particular.

Como respuesta al gran crecimiento en la demanda de software y de la complejidad que el mismo conlleva, surge el **ciclo de vida en cascada**, descripto por primera vez por Winston W. Royce.

Pero, por más que dicho modelo fue considerado como uno de los avances más importantes de esta primera etapa, Royce argumentaba que *"la implementación de este concepto es riesgosa e invita al error"* **[4] MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS - Winston W. Rovce**

Lo que generó que se considere culpable a Winston Royce por no defender el modelo y posteriormente fuera refinado por diversos autores.

ii. GRANDES FRACASOS DEL SOFTWARE -DÉCADA DEL 80

A pesar de los esfuerzos realizados en el desarrollo del software hasta el momento, la ingeniería de software no parecía producir los resultados esperados, el costo de propiedad y mantenimiento del software en la década de 1980 fue dos veces más caro que el propio desarrollo del software y seguían los problemas.

Durante la década del 80 surgieron posibles soluciones al incesante problema del desarrollo de software como por ejemplo Excelerator, las herramientas CASE (Computer Aided Software Engineering, entre otras herramientas. Pero no fueron suficientes, muchas de las posibles soluciones que se propusieron no han funcionado.

Como consecuencia de los persistentes inconvenientes en el desarrollo del software, lo largo de la década tuvieron lugar grandes errores en la implementación de software; uno de los más conocidos fue el caso de "Therac-25" un sistema medio que por un error de programación provocó la muerte de varias personas que se sometieron a tratamientos guiados por este sistema.

A finales de la década, En el 1987 Brooks publicó un artículo llamado **No Silver Bullet. Essence and Accidents of Software Engineering** donde hace una analogía entre el software y un hombre lobo.

Brooks argumentaba que *"No sólo no hay balas de plata a la vista, sino que la misma naturaleza del software impide que las haya."*

[5] No Silver Bullet - Essence and Accident of Software Engineering- 1987 Frederick P. Brooks, Jr.

Lo que intentaba afirmar Brook es que para acabar con dicho monstruo no existían balas de plata (soluciones mágicas) que por sí solas prometan una mejora en productividad, fiabilidad y simplicidad. Aunque Brooks insistía en que no había una "Bala de Plata", argumentaba que una serie de innovaciones que atacan complejidad esencial podría tener importantes mejoras.

iii. DÉCADA DEL 90/2000

En la década de los 90 se produce un gran avance en cuanto a nuevos lanzamientos que afectan directamente el desempeño del desarrollo de software.

Se consolida la programación orientada a objetos (OO) como aproximación para el desarrollo de sistemas informáticos y el surgimiento de Internet puso en auge el desarrollo de sistemas de software internacionales. El crecimiento del uso del navegador Web desarrollado en HTML cambió la manera en la que estaba organizada la visualización y la recuperación de la información, puesta al alcance de usuarios de todo el mundo.

Pero, a principios de la década del 2000, se agudizaron aún más los clásicos problemas del mantenimiento de software, lo que derivó en que se formulara el terminó "métodos ágiles" para definir aquellas nuevas metodologías que estaban naciendo y que se diferenciaban de las metodologías tradicionales, constituidas por métodos estrictos y procesos definidos.

Como resultado de esta nueva ola, en el año 2001 se firma el "Manifiesto Ágil" indicando como estandarte que *"Estamos descubriendo mejores formas de desarrollar software haciéndolo y ayudando a otros a hacerlo."* **[6] Manifiesto Ágil- 2001**

Con esta afirmación, los denominados "agilistas" justifican que su postura se basa primordialmente en las personas y sus interacciones con los demás y no en los procesos definidos, como sucede en las metodologías tradicionales.

El surgimiento de las metodologías ágiles presentó un gran avance en la ingeniería de software, presentando nuevos paradigmas que hasta entonces eran imposibles de implementar en organizaciones completamente burocráticas.

C. SITUACIÓN ACTUAL

En los últimos 20 años se han realizado grandes avances en la ingeniería de software. Existen lenguajes de programación más sofisticados, procesos de desarrollo más maduros, patrones aplicados al desarrollo ágil lo que elevó la calidad del desarrollo a niveles superiores y las aplicaciones que se construyen en la actualidad son más complejas.

Hoy vivimos en una época en la que somos testigos del auge del avance de tecnologías que están revolucionando nuestra forma de pensar e interactuar con el mundo, transformando a los individuos e interfiriendo en los negocios.

Bjarne Stroustrup afirmaba que *"Our civilization runs on software"* **[7] Programming: Principles and Practice Using C++)– Bjarne Stroustrup** cuando hacía referencia a que el software es una forma de alcanzar y cambiar el mundo, obviamente haciendo referencia a un mundo mejor. A pesar de que Bjarne Stroustrup estaba en lo cierto, debemos tener en cuenta que cada vez se construyen sistemas más complejos y dinámicos, para los cuales debemos estar capacitados y adaptarnos rápidamente.

¿Pero qué hay del futuro? Según Jan Bosch afirma que *"Ahora estamos experimentando un salto aún mayor a medida que avanzamos hacia un nuevo nivel de digitalización y automatización."* **[8] Towards a new digital business operating system: Speed, data, ecosystems, and empowerment (keynote) - Jan Bosch** por lo que podemos afirmar que el software sigue en constante evolución y la ingería de software se debe adaptar a estos cambios que van surgiendo.

3. CONCLUSIÓN:

Como pudimos investigar en el presente informe, los primeros problemas que surgieron en el desarrollo de sistemas de software se debían a la implementación de metodologías y procesos definidos donde se priorizaba el seguimiento de un plan y la confección de documentación detallada, lo que ocasionaba que los proyectos se excedan en tiempo y costo, además de arriesgarse a entregar un producto que nadie pidió.

A pesar de que **Frederick Brooks** afirmó que la bala de plata contra el monstruo del software no existía en su momento, ni va a existir, con la aparición de las metodologías ágiles, muchas personas siguen considerando que dichas metodologías pueden convertirse en la "bala de plata" que puede solucionar todos los problemas del software.

Pero si recordamos los conceptos de agilidad, estamos obviando el principio más importante del Manifiesto Ágil *"Las personas y sus interacciones sobre los procesos y herramientas"* **[6] Manifiesto Ágil- 2001,** esto quiere decir que ninguna metodología logrará por si misma resultados mágicos, se debe contar con un equipo de personas capacitadas para lograr resultados aceptables.

Con esta investigación pudimos aprender que cuando las metodologías, herramientas o procesos se imponen como solución mágica o como una religión, es probable que conduzcan al fracaso, por lo cual las afirmaciones de **No Silver Bullet, de Frederick Brooks**, siguen vigentes ya que, la razón por la que no existirán balas de plata es que la complejidad del desarrollo de software no reside actualmente en las herramientas, procesos o metodologías aplicadas, sino en los propios problemas que afronta el software, como por ejemplo la gran cantidad de cambios a los que se encuentra sometido.

La ingeniería de software debe considerar esos problemas y permitir enfoques adaptables para implementar los requerimientos cambiantes, logrando aportar valor a las personas que lo utilizan.

Además, nosotros también debemos adaptarnos y aceptar el cambio para tener ventaja competitiva frente al cliente, lo que nos coloca en una posición de permanente búsqueda de formas de aprendizaje para incorporar y crear valor en entornos más humanos.

4. REFERENCIAS

[1] Ingeniería del software; Un enfoque práctico 7ma.Edición – (Editorial Mc Graw Hill Año 2010)
Roger S. Pressman

[2] NATO SOFTWARE ENGINEERING CONFERENCE 1968-
Professor Dr. F. L. Bauer
http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF

[3] Software engineering: principles and practice
(Editorial: John Wiley & Sons, Año 2007)-
Hans van Vliet
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.128.2614&rep=rep1&type=pdf

[4] MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS - 1970
Winston W. Rovce
https://leadinganswers.typepad.com/leading_answers/files/original_waterfall_paper_winston_royce.pdf

[5] No Silver Bullet - Essence and Accident of Software Engineering- 1987 Frederick P. Brooks, Jr.
http://worrydream.com/refs/Brooks-NoSilverBullet.pdf

[6] Manifiesto Ágil- 2001
https://agilemanifesto.org/iso/es/manifesto.html

[7] Programming: Principles and Practice Using C++ (2nd Edition)– (Editorial:Addison-Wesley Professional; 2014)- Bjarne Stroustrup
https://ptgmedia.pearsoncmg.com/images/9780321992789/samplepages/9780321992789.pdf

[8] Towards a new digital business operating system: Speed, data, ecosystems, and empowerment (keynote) - Jan Bosch - IEEEExplore (traducción al español)
https://ieeexplore.ieee.org/document/8330190

El informe fue elaborado bajo el "formato de dos columnas (Manuscrito estilo PAPPER)", utilizando un template de manuscritos definido por IEEE eXpress Conference Publishing. El template elegido es el de Microsoft Word para A4, tomado de: https://www.ieee.org/conferences/publishing/templates.html

5. BIBLIOGRAFÍA

Ingeniería del software; Un enfoque práctico 7ma.Edición – (Editorial Mc Graw Hill Año 2010)- Roger S. Pressman

INGENIERÍA DE SOFTWARE - 9° Edición (Editorial Addison Wesley-Año 2011) Sommerville, Ian

NATO SOFTWARE ENGINEERING CONFERENCE 1968- Professor Dr. F. L. Bauer
http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF

MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS - 1970 Winston W. Rovce
https://leadinganswers.typepad.com/leading_answers/files/original_waterfall_paper_winston_royce.pdf

No Silver Bullet - Essence and Accident of Software Engineering- 1987 Frederick P. Brooks, Jr.
http://worrydream.com/refs/Brooks-NoSilverBullet.pdf

Manifiesto Ágil- 2001
https://agilemanifesto.org/iso/es/manifesto.html

El manifiesto ágil-2014
https://www.scrummanager.net/bok/index.php?title=El_manifiesto_%C3%A1gil

Programming: Principles and Practice Using C++ (2nd Edition)– (Editorial: Addison-Wesley Professional; 2014)- Bjarne Stroustrup
https://ptgmedia.pearsoncmg.com/images/9780321992789/samplepages/9780321992789.pdf

Crisis del Software- 2016
https://www.scrummanager.net/bok/index.php?title=Crisis_del_software

An Investigation of the Therac-25 Accidents
http://www.cse.msu.edu/~cse470/Public/Handouts/Therac/Therac_1.html

Towards a new digital business operating system: Speed, data, ecosystems, and empowerment (keynote) - Jan Bosch - IEEEExplore (traducción al español)
https://ieeexplore.ieee.org/document/8330190