La evolución del *software*, puntos de crisis

Juan Manuel Bournissen[1]

Resumen

El objetivo de este artículo es mostrar cómo fue evolucionando el *software* desde su nacimiento hasta la actualidad. Intenta mostrar cuáles fueron sus inconvenientes más importantes y cuál es el peor problema que enfrentará con el cambio de siglo. Finalmente, plantea una serie de puntos a tener en cuenta para paliar la actual crisis del *software*.

Palabras clave: *software* - ingeniería del *software* - planificar

Summary

The objective of this article is to show how software has evolved from its beginning until the present, to try to show what the most important difficulties were and what the worst problems will be with the change of the century. Finally, it presents a series of points to keep in mind in order to deal with the current software crisis.

Keywords: Software - Software engineering - Software project management

Résumé

L'objectif de cet article est demontrer comment évolua le software depuis sa naissance jusqu'aujourd'hui. Il tâche de montrer quels furent ses difficultés les plus importantes, et quel est le plus grand problème qu'il aura avec l'arrivée du nouveau siècle. Pour finir il présente une série de thèmes qu'il faut considérer pour amoindrir la crise actuelle du sofware.

Mots clefs: software - systèmes - informatique - crise

Introducción

Al llegar a su oficina un gerente de sistemas sabe que le espera una larga lista de tareas por realizar en el día. Esto no es nada raro, todos los

[1] Juan Manuel Bournissen es Ingeniero en Sistemas y se desempeña como Gerente de Sistemas y Profesor de la Universidad Adventista del Plata.

gerentes las tienen; pero la de un gerente de sistemas, por lo general, está compuesta por quejas de los usuarios. Algunos de ellos no sienten satisfechas sus necesidades, hay sistemas que han producido fallas, o proyectos que no se pueden terminar en término y se exceden en los presupuestos, o tal vez la red no funciona correctamente y hay un *server* fuera de servicio, y encima no se tiene el *backup* del día anterior para ser restaurado en otro *server* de repuesto. Y para complicar más la situación, de la alta dirección se quejan diciendo que la gente de sistemas pasa mucho tiempo sin entregar los productos que fueron pedidos para desarrollarse en forma urgente.

Sin duda le espera un día largo y duro, y por más que se esfuerce no podrá solucionar todos los problemas en el día y se le acumularán para el siguiente y así en forma sucesiva.

Esto indica que su empresa está pasando por una aflicción crónica o una enfermedad prolongada de *software* o como es más conocido, está viviendo lo que se denomina "la crisis del *software*".

Este artículo tiene por objetivo mostrar cómo fue evolucionado el *software* a través de los años y, como cierre, algunos consejos para tratar de salir de la encrucijada que presenta este flagelo.

Evolución del *software*

En la actualidad, las compañías que tienen más éxito son aquellas capaces de comercializar sus productos más rápidamente que su competencia de igual tamaño, abarcando mayor número de clientes y llevando sus productos a destino en el menor tiempo posible.

A todo esto debemos agregarle la reingeniería realizada en estas empresas que colaboran para mejorar sus relaciones con sus proveedores y sobre todo, con los clientes. Pero al analizar sus inversiones en tecnología y desarrollo de *software*, se han dado cuenta de que éstas no han llenado sus expectativas en tiempo y dinero. No responden a las necesidades del mercado.

Las economías occidentales han realizado una transición de una fase industrial de fabricación, reconocida por su capacidad de producción y fiabilidad, a una economía basada en la información, que se caracteriza por una creciente adopción de las necesidades del usuario y la búsqueda de nuevos mercados. Sin embargo, el *software* sigue siendo de producción artesanal, y es creado en forma manual por los programadores; cada aplicación

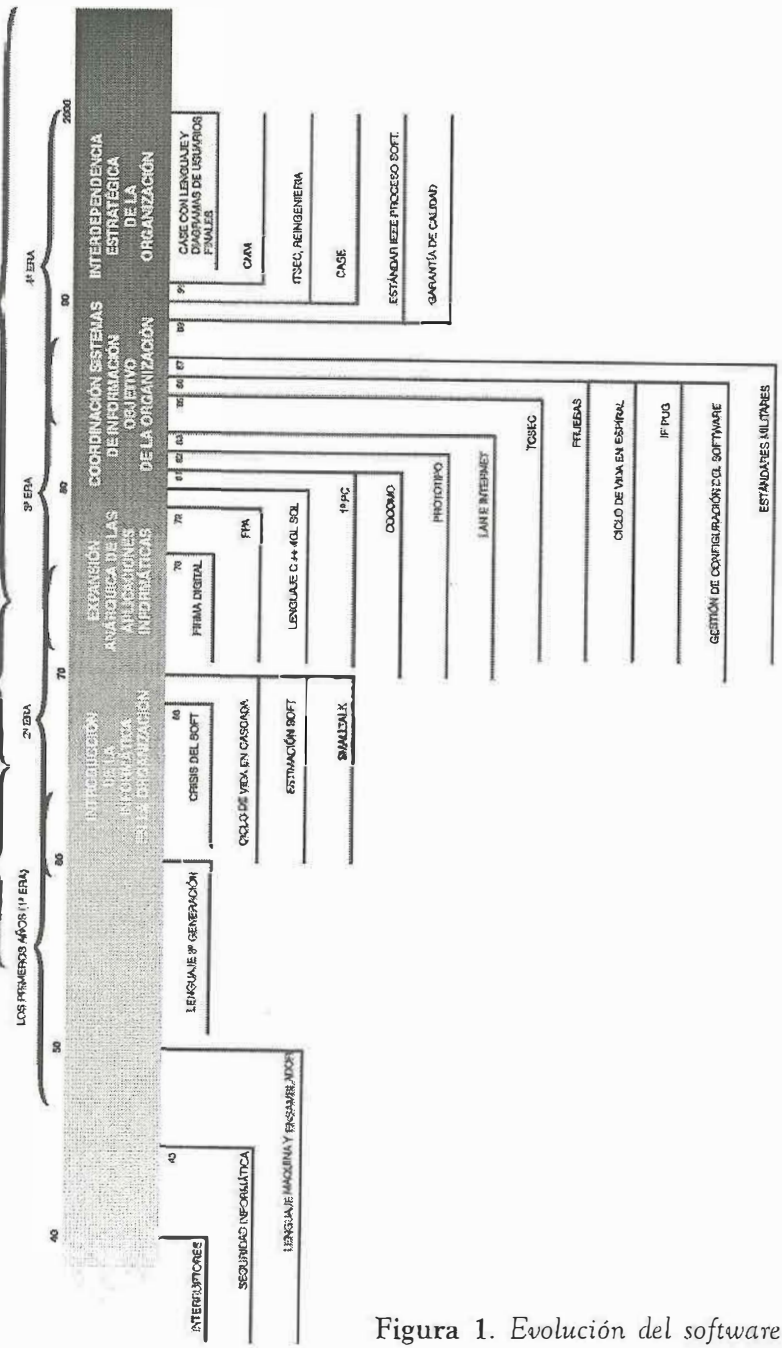

Figura 1. *Evolución del software.*

es construida prácticamente desde cero, por lo tanto, no marcha al mismo ritmo que los demás productos.

Estudios realizados demuestran que los proyectos de *software* rebasan en un 50% el tiempo previsto para su finalización, y en forma conjunta aumentan sus costos. Además, de cada seis nuevos sistemas de gran escala que entran en funcionamiento, dos deben ser cancelados.[2]

Según estudios realizados por IBM en veinticuatro compañías norteamericanas, el 55% de los proyectos de *software* cuesta más de lo previsto, un 68% rebasa los plazos establecidos, y un 88% tiene que ser rediseñado en forma sustancial.[3]

Con el pasar de los años se han implementado distintas herramientas para la creación del *software*, que se trata de reflejar en el gráfico (ver Figura 1) que luego se explica brevemente.

1940

Establecimientos de interruptores (*switches*), paneles de cableado para la programación.

1943

Muy antiguamente se usaba la encriptación y guardado de información para que ésta no fuera interpretada por todos.

En la época del telégrafo se debió aplicar sistemas criptográficos para evitar los escuchas que se enganchaban en las líneas.

En la Segunda Guerra Mundial se usó también el criptosistema como forma de evitar que la información fuera descifrada por el enemigo. En este caso se construyó la computadora Colossus, por medio de la cual los criptógrafos ingleses descifraron los mensajes emitidos por la máquina alemana ENIGMA.

[2] Natalia Juristo Juzgado, "Introducción a la ingeniería del *software*", inédito, p. 7.

[3] System Software Associates, "Informe especializado sobre tecnología de objetos: De la orientación a objetos del software basado en componentes". <http://www.ssamexico.com.mx/html/bpcs/informe/doc1.html> (14 junio 1999).

Primeros años[4]

Desde los años 1950 hasta mediados de la década del '60. En los primeros años de la informática, la creación de *software* era de propósitos específicos. No se construía para ser vendido a otros usuarios con una necesidad igual o similar. Era usado por la persona u empresa que lo generaba, y mantenido por la misma persona; no existía por lo general la documentación del mismo.

1950

Aparece el lenguaje de máquina y el lenguaje ensamblador.

1955

Surge la programación de cualquier modo (1955-1965): la programación era totalmente empírica, con pequeños programas. No existía la gestión.

1960

- **Llegan los lenguajes de tercera generación:** como el FORTRAN, COBOL, PL/1, etc.
- **Introducción de la informática en la organización:** En la década del sesenta aparece la informática en las organizaciones, con grandes máquinas (en volumen, pero de poca capacidad comparadas con cualquier PC de escritorio actual), que necesitaban una estructura especial (edificios especiales con grandes equipos de aire acondicionado), y el lenguaje críptico de los especialistas hizo que el departamento de procesamiento de datos (DPD) se aislara de todos y se formara como una cúpula de cristal. Su función principal era satisfacer las necesidades de automatización de los procesos contables, y tenía como único objetivo la reducción de costos.

  El DPD se encontraba dependiendo de los servicios administrativos, por lo tanto se creó una barrera que impedía la comunicación entre el personal de informática y los directivos de la organización, lo cual no

[4] Roger Pressman, *Ingeniería del Software, un enfoque práctico* (Madrid: McGraw-Hill, 1997), 3-4.

permitía que la gente de informática tuviera acceso a los objetivos de la empresa.

La segunda era[5]

Abarca desde mediados de la década del sesenta hasta finales de la década del setenta.

Comienzan a aparecer la multiprogramación y los sistemas multiusuarios. Las nuevas técnicas acompañadas por *hardware* más potente pasaron a realizar procesamiento de información en forma mucho más rápida. En esta época aparecieron los sistemas de gestión de bases de datos.

El *software* comenzó a generarse de manera tal que pudiese comercializarse, cubriendo las necesidades de las organizaciones, surgieron empresas que creaban *software* para su comercialización masiva.

De esta forma aparecen los sistemas de *software* de grandes cantidades de líneas de código para cubrir todas las posibles necesidades de las empresas. Esto trajo consigo un gran problema, cuando comenzaron a fallar o a no cubrir las necesidades cambiantes de las organizaciones, o adaptarse a otro *hardware*, fue necesario realizar cambios en estas miles de líneas de código, entonces se hizo necesario mantener estos sistemas. Este trabajo de mantenimiento comenzó a consumir muchos recursos. Por lo general, estos sistemas no tenían documentación y habían sido realizados sin metodología alguna, lo cual los hacía imposibles de mantener. Esto da comienzo a la denominada **crisis del *software*** (o aflicción crónica).

1965

Programación a pequeña escala (1965-1975): se comenzó a comprender que los sistemas informáticos no eran solamente procesadores de números, sino también de información en forma simbólica. Se pasó de los programas monolíticos al énfasis en el estudio de los algoritmos y estructuras de datos.

1968

La Comisión de Ciencias de la OTAN, convocada en Garnisch (Alemania), reconoce por primera vez la crisis del *software*. También se

[5] Pressman, *Op. cit.*, 4-5.

detectó la necesidad de llevar adelante la ingeniería del *software*, la que se definió de la siguiente forma: Establecer y usar principios de ingeniería orientados a obtener *software* de manera económica, que sea fiable y funcione eficientemente sobre máquinas reales.

1970

- ***Smalltalk***: primer lenguaje de programación orientado a objetos.[6]
- **Ciclo de vida en cascada** (Royce): la creación de *software* es un proceso de resolución de problemas, con el objetivo de satisfacer una necesidad mediante una solución tratable por computadora. Para ello se cuenta con varios pasos lógicos por realizar. Estos pasos o etapas son los que se aplican en el ciclo de vida de un producto *software*. Este ciclo fue cambiando con el tiempo, mejorándose y adaptándose a las distintas necesidades del mercado, pero para esta época el ciclo de vida en cascada fue de gran ayuda para organizar el trabajo de los desarrolladores y fue usado por mucho tiempo y se usa aún en la actualidad.

  En sus comienzos contaba con cinco etapas que fueron las siguientes: requisitos, diseño, codificación, prueba y operación. En la actualidad se lo usa con nueve etapas que son: planificación y definición de requisitos, diseño del producto, diseño detallado, codificación y pruebas unitarias, integración y pruebas, implantación, explotación y mantenimiento, verificación y validación y, por último, gestión de configuración.

  Entre las ventajas de este ciclo de vida podemos mencionar que las etapas están ordenadas de una forma lógica, incluyendo procesos de revisión y es iterativo.

  Las desventajas son, en primer lugar: se asume que antes de entrar al diseño los requisitos deben ser congelados, cosa que no siempre sucede, de esta forma estamos atados a mantener el *hardware* seleccionado hasta el final del proyecto. Sin embargo, la peor desventaja es que cuando el producto llega al usuario se han invertido casi la totalidad de los recursos, y es imposible realizar cambios.
- **Estimación del *software***: Es un conjunto estimado de valores para algo que debe ser hecho, que por lo general trata de estimar el costo y

[6] Martin J. Odell, *Análisis y diseño orientado a objetos* (México DF: Prentice Hall, 1992), 70.