Capítulo 3: ¿Qué es la Ingeniería de Software?: Una Visión Crítica

La frase 'ingeniería de software' [en la Conferencia de Garmisch] fue elegida deliberadamente provocativa, implicando la necesidad de que la manufactura del software debía estar basada en los tipos de bases teóricas y disciplinas prácticas, que son tradicionales en las ramas establecidas de la ingeniería.

(Naur y Randell, 1969, p. 13)

La ingeniería de software es: "(1) la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software; eso es, la aplicación de la ingeniería al software; (2) el estudio de los enfoques en (1)" (SWEBoK)

(Abran et al., 2004, p. 1-1)

... "el mismo término 'software engineering' en sí mismo es una metáfora que se ha convertido, a lo más, [...] en un accidente histórico emergente"

(Bryant, 2000, p. 80)

3.1. Introducción

En ese capítulo no me concentraré en hacer una revisión de "lo que debería ser" técnicamente el desarrollo de software, de acuerdo a la definición oficial porque lo abordé en Zavala (2000) como un intento de describir esa concepción ideal. En aquel entonces, yo estaba plenamente convencido de que el camino que debería seguirse era implantar la ingeniería de software a nivel personal y a nivel organizacional y me convertí en su promotor. Así, acorde a la definición oficial, hice mi interpretación en ese ensayo. Desde entonces, he recibido retroalimentación de los lectores de ese ensayo y en general, afirman que les es muy útil su lectura, a pesar de que luego caen en la cuenta de que, en la práctica, eso que se pregona, no se no se lleva a cabo. Este capítulo, en cierto sentido, es el complemento de aquel. La fuente oficial de lo que debería ser la ingeniería de software es la "Guide to Software Engineering Body of Knowledge" (SWEBoK®)

(Abran et al., 2004) a la cual remito al lector. Para mí era plausible, como lo es para cualquier ingeniero, que la ingeniería da orden y disciplina, sin embargo, en el desarrollo de software esto no se cumple cabalmente en gran parte por la dinámica de interacción social con quienes juegan el papel de usuario o cliente, que define los requerimientos y califica los resultados. Así que la llamada ingeniería de requerimientos, en la ingeniería de software, es un engaño, similar a la ingeniería social ("social engineering") que de ingeniería sólo lleva el nombre.

Este capítulo primero aborda el surgimiento de la industria del software con el primer sistema de software de negocios en el mundo en Gran Bretaña, hasta su establecimiento como negocio a finales de los años 1960s. Se aborda el origen de los proyectos de *outsourcing* y de la consultoría en software. En la segunda parte se aborda la "crisis del software" como un mito y las consecuencias del impulso de la ingeniería de software como su solución y como una disciplina que pretende reemplazar a las ciencias de la computación, con muchas consecuencias. Luego, se expone el proceso de institucionalización de la ingeniería de software, primero en la industria, luego como profesión y en la academia, donde a pesar de todo, carece de sus fundamentos. En la sexta parte, se aborda la ingeniería de software del Free/Libre Open Source Software (FLOSS) como alteridad a la ingeniería de software tradicional. En la séptima parte se analiza la controversia sobre la identidad de la disciplina. En la octava parte, se analiza el estado actual de la teoría y se expone un esquema para establecerla a partir de la práctica. Finalmente se cierra con las conclusiones.

3.2. La industria del software y su importancia

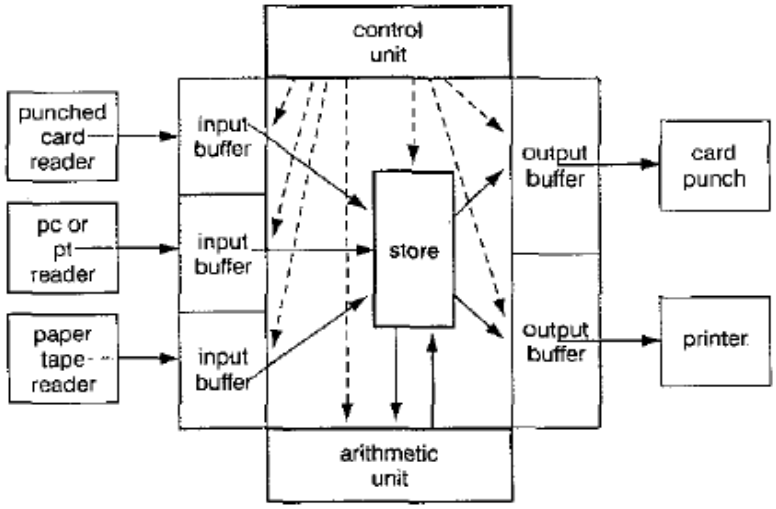
La "Electronic Numerical Integrator and Computer" (ENIAC) fue la primera computadora electrónica construida entre 1943 y 1946 por John Mauchly y John Presper Eckert bajo un contrato del Departament of Defense (DoD) de los Estados Unidos. John von Newmann creó el modelo lógico de la ENIAC, con la

inspiración de Alan Turing, Howard Aiken y Norbert Wiener, entre otros científicos de la época (cf. Aspray, 1990). Sin embargo, a Turing se le considera el inventor de la "teoría de la computación" (*computation theory*) o "computabilidad" (*computability*), que es estudio de las habilidades y limitaciones de las computadoras digitales, por el invento de la Máquina de Turing, una máquina de cálculo universal propuesta teóricamente en 1936-1937. Al respecto se recomienda ampliamente leer la disección del trabajo de Turing en Petzold (2008). En honor a Turing, la ACM instituyó en 1966, la presea *Alan Turing Award*, considerado como el Premio Nobel en computación. En 2012 se celebrará el año del centenario del nacimiento de Turing con celebraciones y eventos.

Aunque el software, desde un principio, tuvo aplicaciones científicas y militares, la revolución llegó cuando el software se comenzó a utilizar en los negocios y luego la administración pública. Esta es la breve historia del primer sistema "Enterprise Resource Planning" (ERP) (software de negocios) en el mundo, sus lecciones y sus protagonistas y de aquellos que fundaron esta industria, es la historia de la industria del software antes de que existiera. Es la historia paralela al desarrollo de software científico y militar que precedió a los grandes gigantes de la industria de la computación.

J. Lyons era una gran empresa de alimentos y abarrotes, tés, helados, panes, pasteles y pays. Era una de los llamados "teashops" en Inglaterra. J. Lyons fue la primera empresa en contar con un "departamento de sistemas", llamado "Systems Research" luego renombrado a "Organisation and Methods" (Caminer, 2002, p. 5). Ese departamento funcionó como el departamento de "Investigación y Desarrollo" orientada a la mejora de los proceso de negocio. Fue dirigido por David T. Caminer. En 1947, J. Lyons era una compañía líder en el campo de los sistemas de oficina. En ese año, dos de sus ejecutivos visitaron los Estados Unidos y vieron la posibilidad de usar el "cerebro electrónico", publicitado en la prensa, en sus negocios (Caminer, 1997, pp. 585-586). En

1949, después de que la computadora EDSAC[8] funcionó en Cambridge, en J. Lyons se decidió construir la LEO I, la primera computadora de negocios del mundo, la equivalente de la UNIVAC ("Universal Automatic Computer"), la primera computadora de negocios en los Estados Unidos, disponible para su venta masiva en 1951 (cf. Ceruzzi, 2003).



Fuente: Caminer (2002, p. 276).

Figura 3.1. Esquema lógico de LEO I

El Proyecto LEO fue dirigido por David T. Caminer, el primer "systems and programming manager" del mundo. El 17 de noviembre de 1951 corrió por primera vez el programa de software en LEO I, la primera de tres generaciones de computadoras; y con ello, nació la moderna tecnología de información (Caminer, 2002, p. 273). El 30 de noviembre de 1951 entró en operación la

[8] La Electronic Delay Storage Automatic Calculator (EDSAC) fue de las primeras computadoras creadas en Gran Btretaña. Fue construida en la Universidad de Cambridge. Estuvo inspirada en la Electronic Numerical Integrator And Computer (ENIAC) construida en los Estados Unidos de 1943 a 1946.

primera aplicación de software de negocios[9] en el mundo (Aris, 2000, p.4; Caminer et al., 1998, pp. 1-42), que convirtió J. Lyons en la *primera empresa de software de negocios en el mundo* y unos meses después, también fue *la primera empresa de servicios de software* (Caminer, 1997, p. 951). LEO fue el primer proyecto de software de negocios:

> La estrategia con la que LEO especificó y diseñó los sistemas de los clientes aventureros (podría ser considerada como *una temprana manifestación del outsourcing*) fue acertada, pero no podía ser sostenida indefinidamente. *Los clientes fueron animados a alquilar su propio staff profesional*, a menudo de LEO, y gradualmente asumieron estas tareas para ellos mismos. Sin embargo, *antes de fines de los años 1950s la mayoría de las aplicaciones de computadora comerciales sofisticadas del mundo eran todavía diseñadas por LEO, o diseñadas por ingenieros en sistemas entrenados por LEO*, de acuerdo con los principios delineados por Caminer y su staff, aunque para entonces la competencia estaba comenzando a ponerse al corriente. (Aris, 2000, p. 14, traducción libre, énfasis agregado).

LEO fue, sin duda el primer sistema empresarial ERP en el mundo (Figura 3.1.):

> Hacia fines de 1956, cinco años después de la ceremonia con motivo de la primera corrida del trabajo de valuaciones de la panadería, LEO estaba procesando una carga representativa de *aplicaciones de oficina — nómina, distribución, facturación de ventas, contabilidad y control de inventarios*— y, al mismo tiempo, el aceleramiento de las operaciones físicas de J Lyons y suministrando información oportuna para la acción táctica de la administración (management). (Caminer, 1997, p. 596, traducción libre, énfasis agregado).

Por el recuento que hicieron Caminer et al., (1998), la experiencia de LEO hizo factible el negocio de software de negocios entre 1951 y 1963, es decir, por poco más de 10 años, interviniendo en los departamento de contabilidad y

[9] Software empresarial o *Enterprise Resource Planning* (ERP) Systems, como se les conoce hoy.

finanzas, lo que actualmente siguen siendo las aplicaciones de negocios. Entre 1960 y 1963, LEO creó la tercera generación de su hardware y software. Luego, en un intento de incursionar en el negocio del hardware, *LEO Computers* para entonces, se unió con la *English Electric* en 1963 para buscar su expansión al negocio del hardware. Sin embargo, después de un intento de consolidación y de una crisis en 1965 que se prolongó hasta 1968, LEO terminó definitivamente (Caminer et al., 1998, pp. 1-154, Caminer 1997, p. 597). Aunque LEO terminó, la LEO 326, la última instalación de LEO, estuvo en funcionamiento en el servicio postal inglés desde 1964 hasta 1981 (Caminer, 2002, p. 279).

LEO fue un éxito por varias razones, pero quizás lo fue más porque, en mi opinión, la solución que ofreció fue "adaptar la tecnología a la organización y a su modelo ideal de negocio" (Baskerville, 2003, p. 258), un principio que sigue siendo válido ya que ahí está la ventaja competitiva de los negocios y no en adaptar la organización a un software diseñado para un mercado masivo como frecuentemente ocurre. Además, David Caminer y su equipo fueron exitosos "por una combinación de experiencia previa, intuición, y aprendizaje concienzudo e imaginativo en el trabajo, … [que] conjuntó un *estándar de mejor práctica* que fue única para su tiempo" según John Aris, uno de los miembros de ese staff (Aris, 2000, p. 15, énfasis agregado).

LEO dejó una gran lección sobre la no-sustentabilidad de su modelo operativo y de negocios del software, basado en individuos altamente capaces:

> LEO pudo haber perdido una oportunidad de reutilizar los sistemas y el código (moviéndose hacia los paquetes de software), pero las restricciones de hardware, de espacio y tiempo lo hicieron [… factible] hasta fines de 1950s. LEO […] perdió la importancia de la compatibilidad hacia adelante ("*forward compatibility*"), algo que IBM explotó muy bien en los 1960s. *[Además, LEO fue] muy dependiente de individuos de alta calidad que trabajan muy duro, lo que no fue sostenible en el largo plazo*. (Aris, 2000, p. 15, traducción libre, énfasis agregado).

Lo anterior, sigue siendo un serio problema hoy día. Además indica que no se logró la amplia socialización del conocimiento en el equipo.