Mario Piattini

Mario.Piattini@uclm.es

PHd in Computer Science
Catedrático de la Universidad
de Castilla La Mancha
Director Científico del Alarcos
Quality Center UCLM

Evolución de la Ingeniería del Software y la formación de profesionales

La industria del software ya tiene casi setenta años y en este período ha realizado grandes avances, ya que disponemos de lenguajes de programación más sofisticados, procesos de desarrollo más maduros, y las aplicaciones que se construyen en la actualidad son más complejas. De hecho, el software forma parte de nuestras vidas, está en todos los aparatos que manejamos, medios de transporte, sistemas de telecomunicaciones, equipos médicos, sistemas de administración pública y financieros, en el arte, en el ocio y en el entretenimiento. En definitiva, como decía Bjarne Stroustrup: “Our civilization runs on software”.

Ahora bien, hay que tener en cuenta como señala Grady Booch que: “el desarrollo de software ha sido, es y probablemente será fundamentalmente difícil”. En efecto, cada vez se construyen sistemas más complejos desde el punto de vista tecnológico. A lo que hay que añadir también, muchas veces, problemas en el gobierno de las tecnologías y sistemas informáticos, defectos producidos por primar la puesta en marcha de los sistemas de forma oportuna -sacrificando su calidad-, y la falta de formación de los responsables del desarrollo de software.

A continuación, resumiremos la evolución de la Ingeniería del Software desde su nacimiento hasta la actualidad; que según Barry Boehm, ha seguido un proceso de tesis, antítesis y síntesis que explicaría las diferentes propuestas y contrapropuestas que se han sucedido a lo largo de estas décadas; señalando las nuevas demandas que se han producido en la formación de los profesionales.

• **Décadas de los 40 y 50,** en estas décadas el coste del hardware era tremendamente superior al del software, que tenía por lo tanto una importancia relativa mucho menor. Se consideraba además que el software se podía desarrollar de la misma forma que se desarrolla el hardware; y, de hecho, los primeros ingenieros que se ocupaban del software eran los mismos que desarrollaban el hardware.

• **Década de los 60,** a pesar de importantes éxitos como las misiones de la NASA, se empieza a hacer evidente que el software se diferencia demasiado del hardware para poder ser tratado de la misma manera. Es la época de los famosos "códigos espagueti" (muy difíciles de entender incluso por quien lo escribía) y la aparición de "héroes" que después de varias noches sin dormir conseguían arreglar a último minuto el software para cumplir los plazos marcados. En el NASA/IEEE Software Engineering Workshop de 1966; y las conferencias de la OTAN en 1968 y 1969, se analizó la "crisis del software", y se plantearon ideas fundamentales como "reutilización" o "arquitectura software". En 1968 aparece también el artículo de Dijkstra "Go To Statement Considered Harmful" que impulsó la programación estructurada y en el congreso IFIP se cita por primera vez el concepto de "factoría o fábrica de software". Sin embargo, la formación de los profesionales sigue siendo ad-hoc y más centrada en los sistemas y en la programación, que en una verdadera Ingeniería del Software.

• **Década de los 70**, en esta década las organizaciones empezaron a comprobar que los costes del software superaban a los del hardware. Parnas propone la descomposición modular y el concepto de ocultamiento de información (information hiding), Chen el modelo E/R y Royce el modelo de ciclo de vida en cascada. La formación de los profesionales de la Ingeniería del Software se centra entonces en las metodologías estructuradas (Warnier, Jackson, Myers, Yourdon y Constantine, Gane y Sarson, Demarco, SSADM, MERISE, etc.) que supusieron un avance importante en el análisis y diseño de software.

• **Década de los 80.** Leo Osterweil impartió una charla invitada en la International Conference on Software Engineering (ICSE) cuyo título fue "Software processes are software too" que supuso el inicio de una nueva forma de abordar los procesos software. Los problemas de no conformidad de proceso se intentaron resolver con estándares como el DoD-STD-2167 o el MILSTD- 1521B por parte del Departamento de Defensa de EEUU que, con el fin de mejorar la calidad de sus sistemas y evaluar a sus proveedores, encargan al entonces recientemente creado Software Engineering Institute (SEI) de la Universidad Carnegie Mellon, un modelo de madurez de la capacidad software (SW-CMM) que desarrollaría Watts Humphrey. En cuanto a la tecnología, se automatiza parte del ciclo de vida del software, apareciendo la conocida como primera generación de herramientas CASE, y los lenguajes de programación orientados a objetos que, si bien empezaron a finales de la década de los sesenta con el lenguaje Simula y en los setenta con Smalltalk, se difundieron sobre todo en la década de los ochenta con la aparición de C++, Objective-C y Eiffel. La formación de los profesionales del software requiere entonces el manejo de las herramientas CASE, comprender el gran cambio de paradigma que supone la orientación a objetos, y adquirir conocimientos sobre los procesos software y los modelos de madurez.

• **Década de los 90,** durante la cual se desarrollan los modelos relacionados con la mejora de procesos software, como Ideal, TSP o PSP, y las normas y estándares de calidad como la ISO 9126, ISO 12207, ISO 9000-3, etc. También durante esta década se consolida la orientación a objetos (OO) como aproximación para el desarrollo de sistemas informáticos, apareciendo más de cien metodologías, que terminan dando lugar a la aparición del Lenguaje de Modelado Unificado (UML) y el Proceso Unificado (UP). También surgen en los noventa y la década siguiente multitud de técnicas y conocimientos sobre la construcción de sistemas orientados a objetos: patrones, heurísticas, refactorizaciones, etc. Lo que supone una profundización en la formación de los profesionales que deben adquirir todas estas "buenas prácticas" para la correcta construcción del software. Por otro lado, los problemas del año 2000 y del Euro, que agudizaron aún más los clásicos problemas del mantenimiento de software, hicieron plantearse a muchas organizaciones la conveniencia de externalizar (outsourcing) sus procesos de mantenimiento, impulsando la creación por parte de muchas empresas de centros y unidades dedicadas específicamente a la externalización. La gestión y el desarrollo de software externalizado demanda conocimientos y habilidades especializados a los Ingenieros de Software.

• **Década de los 2000.** Se firma el "Manifiesto Ágil" como intento de simplificar la complejidad de las metodologías existentes y en respuesta a los modelos "pesados" tipo CMM, y surgen, los métodos híbridos, que buscan un equilibrio, combinando la adaptabilidad de los ágiles con la formalidad y documentación de los métodos rigurosos. Actualmente vivimos el auge de este tipo de métodos, especialmente de Scrum, y ha sido necesario reciclar a los Ingenieros de Software en la "cultura" y técnicas ágiles.
Cabe destacar también que en esta década se difunden el Desarrollo Software Dirigido por Modelos (DSDM) y las líneas o familias de productos software, que suponen un esfuerzo al Ingeniero del Software al trabajar con modelos de alto nivel como elemento principal del desarrollo y mantenimiento de software. Otro tema relevante es el Desarrollo Distribuido de Software (especialmente cuando los equipos se distribuyen más allá de las fronteras de una nación, recibiendo el nombre de Desarrollo Global de Software (GSD)), que requiere una formación mucho más amplia del Ingeniero de Software, para resolver problemas como: comunicación inadecuada, diversidad cultural, gestión del conocimiento o diferencia horaria, entre otros.
Por último, en esta década queremos resaltar la Ingeniería del Software Empírica (ESE) y la Ingeniería del Software Basada en Evidencias (EBSE), que sentaron las bases para la experimentación y rigurosidad en Ingeniería del Software.

• **Década de los 2010**
En esta década, además de afianzarse las líneas descritas en las décadas anteriores, estamos asistiendo a una mayor integración entre la Ingeniería del Software y la Ingeniería de Sistemas -destacando el papel de los requisitos no funcionales y, sobre todo, de la seguridad-; la importancia de la "Ciencia, Gestión e Ingeniería de los Servicios" que requiere un enfoque interdisciplinar (informática, marketing, gestión empresarial, ciencias cognitivas, derecho, etc.) a la hora de abordar el diseño de los servicios; la necesidad de adaptar los métodos de desarrollo de software para trabajar en un "mundo abierto" -crucial cuando nos enfrentamos a dominios tales como la inteligencia ambiental, las aplicaciones conscientes del contexto, y la computación pervasiva-; los "Sistemas de Sistemas Intensivos en Software" (SISOS) con decenas de millones de líneas de código, decenas de interfaces externas, proveedores "competitivos", jerarquías complejas, etc.
También estamos viendo ya la implantación de la "Ingeniería del Software Continua", y su correspondiente tecnología y "filosofía" "DevOps", que logran reducir el tiempo entre que se compromete un cambio en el sistema y que se ponga en producción normal; lo que requiere un cambio cultural para aceptar la responsabilidad compartida (entre desarrollo y operación) de entregar software de alta calidad al usuario final. Además de necesitar aprender nuevos conceptos (p.ej. infraestructura como código o microsistemas) es necesario que el desarrollador amplíe su visión con elementos de operación.

Conclusiones

Hemos resumido muy brevemente la historia de la Ingeniería del Software y sus principales innovaciones hasta la fecha; sin embargo, la Ingeniería del Software deberá evolucionar aún mucho más para adaptarse a la computación molecular, la computación cuántica y la computación biológica, etc.

Hay que reconocer que, como toda disciplina joven, la Ingeniería del Software ha recorrido algunas veces caminos poco claros, siguiendo determinadas "modas" sin saber muy bien a dónde conducían; es la denominada por Alan Davis "Lemmingeniería del Software", que ha causado confusión y decepción en muchos usuarios y profesionales del software. Por eso es muy importante la formación del Ingeniero Informático, al que se le desafía para que proporcione soluciones a los problemas dentro de un cierto coste y tiempo, siendo responsable de construir productos de calidad (usables, seguros, mantenibles, etc.).

Quería finalizar diciendo que -parafraseando a nuestro ilustre pintor Raúl Soldi-, "las tecnologías de desarrollo de software progresan, mientras que la Ingeniería del Software evoluciona", de ahí a veces la dificultad para obtener productos software de calidad ●