

Tecnología

Luis Fernández Sanz

Universidad Europea de Madrid, Socio de ATI

<lufern@dpris.esi.uem.es>

1. Introducción

Actualmente hablamos de ingeniería del software, pero ¿la manera en la que desarrollamos software actualmente se puede denominar, con propiedad, ingeniería? La palabra ingeniería tiene una connotación de prestigio que provoca que muchas disciplinas traten de autocalificarse como tal. Si bien existen estudios detallados sobre el concepto de ingeniería (véase, por ejemplo, [Shaw,1994]), en general se concibe la ingeniería como una aplicación práctica y eficiente de los conocimientos científicos. Desde este punto de vista, el sector del software se podría encontrar en una fase de producción comercial en algunas organizaciones [Shaw,1994] pero seguramente resulta demasiado optimista hablar de la aplicación generalizada de una auténtica ingeniería. En este artículo, se repasan algunos aspectos de la situación actual de la ingeniería del software para terminar por sugerir al lector una visión de futuro deseable para los próximos ¿cinco o diez o veinticinco? años.

2. ¿Se aplica actualmente la ingeniería del software?

La investigación y el desarrollo de técnicas y métodos de ingeniería del software son constantes y suelen suponer interesantes avances en la resolución de problemas de desarrollo de software. Sin embargo, es habitual que en la práctica diaria profesional no se incluya prácticamente ninguna de las recomendaciones más elementales de la ingeniería del software. En efecto, es habitual que el desarrollo de software se parezca más al descontrol del cuento de «si los programadores fueran albañiles...» ([Novatica, 1996]) que a una idílica y bien organizada factoría de software (concepto de gran vigencia a finales de los ochenta [Nunamaker *et al.*, 1988]). De hecho, las evaluaciones de los procesos productivos de software realizadas a raíz de los modelos de procesos de software (por ejemplo, con CMM [Paulk *et al.*, 1993]) confirman que el desarrollo de software suele estar básicamente en estado caótico. Y no sólo en, como uno podría pensar, pequeñas organizaciones de un país mediterráneo como el nuestro sino en también en las empresas adjudicatorias de interesantes contratos de defensa de países «avanzados» como EE.UU. o Japón. Si bien estos modelos de evaluación aún acumulan distintas acusaciones de rigidez o falta de contraste empírico, los resultados obtenidos en sus evaluaciones resultan consistentes con la sensación de constante «apagafuegos» que suelen sufrir los profesionales del software. También suelen revelar la práctica despreocupación de los responsables de las organizaciones de software por la mejora de la calidad de sus procesos de trabajo o de sus productos: bien sea por contar con una situación interna de empresa poco propicia, porque el ambiente del mercado no fomenta la preocupación por estos temas o bien por su poco interés real en la búsqueda de la calidad.

El futuro de la ingeniería del software o ¿cuándo será el software un producto de ingeniería?

Parece difícil saber exactamente las causas de esta situación. No obstante, cabe la posibilidad de reflexionar sobre algunas ideas que, quizás, estén aportando su grano de arena a mantener este caos. Por una parte, las relaciones entre suministrador y cliente en el mercado actual de desarrollo de software. La dinámica del mercado y la tradición imperante en la gestión de las ofertas de software no fomentan el uso de planificaciones bien fundadas en un análisis apropiado de los requisitos. Lamentablemente, la gestión de los requisitos y de la planificación no se realiza en las mejores condiciones para los analistas y jefes de proyecto. Por otra parte, actualmente las empresas se quejan de la carencia de profesionales cualificados para cubrir sus necesidades de personal (que, en muchos casos, se ha disparado con la explosión de Internet). Así, el informe de IDC para *Microsoft* [Milroy y Rajah, 2000] indica la falta de hasta 1,200.000 profesionales relacionados con las TIC (tecnologías de la información y de las comunicaciones)¹. Otras informes manejados por Cisco también confirman esta tendencia. Es de esperar que las necesidades de personal no fomenten precisamente que todos los nuevos profesionales lleguen a la empresa con los conocimientos apropiados de ingeniería del software para solventar los problemas de desarrollo de aplicaciones. De hecho, habrá que capturar personas que prácticamente no cuenten con casi ningún tipo de formación informática y a quienes será difícil transmitir ciertas técnicas y recomendaciones para un buen desarrollo de software.

Las posibilidades reales de aplicación de las técnicas de ingeniería del software quedan, pues, limitadas también por la ya preocupante carencia de profesionales específicamente formados en ellas. Estos problemas entroncan con el debate sobre la ingeniería del software como profesión (un tema también que se abordará en un artículo de *Novática* en breve). Lo cierto es que la profesión de ingeniero de software (o especialista en software) difiere muchísimo de la concepción tradicional de informático relacionada con la ciencia de la computación (*computer science*). Es necesario que el ingeniero de software sea, ante todo, ingeniero y que sea capaz de trasladar con sentido práctico los conocimientos científicos de la informática al desarrollo y mantenimiento de software. Esto obliga a que el ingeniero de software aporte soluciones reales a los problemas diarios de las organizaciones de software, lo que puede suponer agregar a los conocimientos estrictamente técnicos, habilidades y formación en aspectos de gestión, economía, legislación, etc. Evidentemente el proyecto **SWEBOK** (*Software Engineering Body of Knowledge*: <http://www.swebok.org>) actualmente en marcha con el apoyo del **IEEE** y de la **ACM** puede ayudar a determinar mejor el conjunto de conocimientos de la ingeniería del software y constituir una base de trabajo para la formación de profesionales.

Evidentemente, la formación en técnicas y métodos de ingeniería del software podría paliar de alguna forma el

déficit de aplicación de la ingeniería del software. Lamentablemente, este tipo de formación parece oscilar entre la influencia de la corriente académica de ciencia de la computación² (normalmente tan alejada de la realidad profesional y de la filosofía de ingeniería) y el excesivo practicismo cortoplacista de algunos responsables de empresa (que consideran poco útil aquello que no tiene relación inmediata con el código y el producto final). Así, el voluntarismo de escasos entusiastas de la ingeniería del software, muchas veces incomprendidos desde ambos extremos, debe contener con la incompreensión del mundo profesional y de empresa en el que reina muchas veces la idea de que «todo esto no sirve para nada» y «nadie lo aplica». Hay que ser capaces de transmitir la idea de que ya hay bastantes casos de aplicación en los que las mejoras han compensado con creces el esfuerzo y la inversión en la mejora de los métodos de trabajo. También hay que evitar, por ejemplo, la idea de que un curso de una semana convierte, sin otro tipo de acciones o apoyos, a los empleados en muy productivos ingenieros de software con procesos de trabajo muy refinados, aunque es fundamental contar con formación cualificada en las técnicas (pero siempre inmersa en todo un plan de mejora).

3. ¿Renovarse o morir? o ¿renovarse para morir?

Una de las sensaciones más habituales de los profesionales de la informática en general, y del desarrollo de software en particular, consiste en la impresión de desbordamiento ante la avalancha constante de nuevas tecnologías, entornos, siglas, etc. Parece como si los seres humanos fueran incapaces de gestionar con sentido el mundo tecnológico que han generado. La renovación tecnológica parece dejar exhaustos a muchos profesionales independientemente de las ventajas y oportunidades que supone o de la aceptación de su inevitabilidad.

Especialmente el último año se ha caracterizado por una locura (que ha llegado con retraso a España) sobre el mundo de Internet. Las nuevas modas arrasan y, al igual que en un río revuelto, ésta parece que ha supuesto un paso atrás en la ingeniería del software. En efecto, el desarrollo de las aplicaciones de gestión comenzaba a contar con un entorno relativamente estable y las organizaciones habían empezado a asimilar ciertas prácticas de ingeniería del software. Pero, ahora, la acuciante necesidad de aplicaciones para Internet no ha supuesto un lógico paso más en el esquema de desarrollo sino un terremoto que ha revolucionado las estructuras profesionales, de mercado y de organización. Los medios de comunicación generales y los especializados no se cansan de proclamar, con ejemplos, cómo cualquiera (da lo mismo si alguna vez estudió o practicó la disciplina de desarrollar software con cabeza o no), sin más que consultar los recursos gratuitos existentes en Internet, puede crear magníficos sitios y aplicaciones para web. Y es totalmente cierto. Muchos sitios web suelen ser atractivos pero han descuidado aspectos básicos de estructura y diseño que generan un infierno cuando hay que abordar el mantenimiento (otra vez el mantenimiento de software tan decisivo en la problemática de la crisis del software durante los años setenta y ochenta). Precisamente ahora que la tremenda confusión en el desarrollo de software orientado a objetos va resolviéndose, para bien o para mal, con la aceptación global de UML [Rumbaugh et al., 1999] como notación de trabajo. Parece que la ingeniería del software va avanzando y retrocediendo continuamente, movida continuamente por las variaciones en el mercado de tecnologías de la informa-

ción. En cualquier caso, la ingeniería del software tiene, habitualmente, que responder a posteriori a los cambios en la tecnología de software, es decir, deberá aportar métodos y técnicas para su desarrollo y mantenimiento una vez que se conozcan adecuadamente las características de dichas novedades técnicas en el mundo del software.

4. Presencia de la ingeniería del software

A pesar de que el sueño de la «factoría de software» sigue vigente entre todas las organizaciones de desarrollo (e incluso existan centros con esta denominación en España), lo cierto es que sigue existiendo un alto porcentaje de artesanía y trabajo a medida. Los datos de SEDISI (<http://www.sedisi.es>) pueden ser reveladores de la gran cifra de negocio que supone el desarrollo a medida en España, a pesar del gran incremento de soluciones de software que están agilizando la gestión empresarial (bien a través de paquetes cerrados o bien mediante los omnipresentes y archiconocidos sistemas de ERP). De hecho, el desarrollo de software a medida y el mantenimiento de aplicaciones supusieron en 1998 una cifra de negocio de unos 170.000 millones de pesetas (un 12,4% aproximadamente del total del mercado), cifra prácticamente igual al total de ventas de productos de software. Ciertamente en el desarrollo de software a medida es donde tradicionalmente se ha buscado un mercado natural para la ingeniería del software pero cualquier organización de desarrollo tiene que establecer un marco de trabajo moderno para sus operaciones³.

Quizás si preguntamos a alguien qué es lo primero que le sugiere la ingeniería del software, es muy probable que nos comente algo sobre metodologías, métodos de diseño, etc. Es verdad que el sueño de los desarrolladores ha estado depositado en encontrar una metodología o proceso de trabajo que guiara de forma determinista, sin ningún tipo de fallos, la producción de software. A ser posible, también se busca la herramienta que permita ejecutar todas las operaciones de desarrollo con un mínimo de aportación humana. Lo cierto es que se difícil recoger toda la complejidad y facetas del desarrollo de software (recogido como creación intelectual en el R.D.L. 1/1996 sobre propiedad intelectual) en un método más o menos simple.

De hecho, la tendencia actual se orienta a la formulación de guías generales que requieren de la experiencia y la capacidad de análisis de los ingenieros de software para su adaptación a cada proyecto o situación concreta. En el caso del proceso de desarrollo, las metodologías se entienden más como indicaciones genéricas que como rígidos esquemas como los propuestos en los años ochenta. Así, hablamos sobre todo de procesos (por ejemplo, el *Unified Process* [Jacobson et al., 1999]) y ciclos de vida genéricos y flexibles (por ejemplo, ciclos iterativos) y de la organización de los procesos mediante grandes líneas de evaluación y mejora (por ejemplo, SPICE [ISO, 1998]). En el caso de las técnicas de desarrollo, la tendencia apunta a heurísticas y guías formuladas más como patrones [Grand, 1998] y *frameworks* que como métodos con pasos muy detallados. En definitiva, se sigue más bien el esquema de identificar buenas prácticas (*best practices*) genéricas cuya aplicación inteligente depende de los ingenieros de software y los jefes de proyecto.

Sin embargo, el verdadero carácter de ingeniería parece apreciarse más en la gestión del software a través de la planificación de proyectos, estimaciones y gestión de recursos, aseguramiento de calidad, gestión de configuración,

gestión de documentación, etc. En este aspecto de la ingeniería del software lo que se echa en falta en la práctica es, simplemente, que se ponga en práctica mediante el empleo de alguna técnica mínimamente rigurosa. Al fin y al cabo, el desarrollo hay que hacerlo de todas formas pero la gestión de todos los aspectos mencionados aparece en muchas ocasiones como algo secundario que se resuelve como se puede, sobre la marcha.

No obstante, la verdadera tendencia que, definitivamente, parece que el software en general ha asumido (no sólo desde el punto de vista de ingeniería de software) es la necesidad de contrastar empíricamente las propuestas de programación, de técnicas, de métodos, etc. De hecho la situación empezaba a ser insostenible incluso en los ámbitos que contaban con fama de mayor rigor⁴ [Zelkowitz y Wallace, 1998]. La gestión mediante datos y medidas validadas y bien aplicadas o la experimentación bien diseñada permite comprobar si las técnicas propuestas realmente son mejores que sus predecesoras. Esta tendencia ha empezado afortunadamente a calar en la comunidad investigadora y profesional internacional, si bien, lamentablemente, todavía no ha llegado con suficiente fuerza a España. A pesar de ello, ya ha aparecido la primera publicación al respecto [Dolado y Fernández, 2000]. Un ejemplo de esta preocupación internacional es la creación de publicaciones específicamente orientadas a la experimentación y la medición (por ejemplo, *Empirical Software Engineering*).

Por último, hay que señalar que la ingeniería del software tiene respuestas (aunque hay que seguir trabajando en refinarlas y ponerlas en práctica) para el desarrollo en entornos de cliente/servidor e Internet. En el mundo de las aplicaciones ERP integradas y de su implantación personalizada, no obstante, hace falta que se realice un esfuerzo en aplicar técnicas de aseguramiento de calidad y de adaptación de las técnicas de análisis y diseño. Son dos campos donde la ingeniería del software debería avanzar para solventar muchos de los problemas diarios de los profesionales del software.

5. El futuro

Quizás hubiera podido comenzar con este apartado el presente artículo. La idea consiste en tratar de presentar un futuro de lo que me gustaría fuera el mundo de la ingeniería del software en el futuro. Para ello, me gustaría poder describir la idea del futuro ideal para los desarrolladores de software, un futuro posible en el que les gustaría trabajar, un futuro en el que:

- La planificación de los proyectos se hace con rigor y está bien ajustada a la complejidad y tamaño de las aplicaciones y de sus requisitos.
- Las especificaciones son el producto de un exhaustivo y sólido trabajo de interacción con el cliente/usuario y cuentan con su aprobación, incluido el acuerdo sobre los criterios de aceptación de la aplicación.
- Se cuenta con colecciones de componentes, funciones, etc. realmente controlados, especificados y gestionados que solventan buena parte de la generación del código, pasando los ingenieros de software a efectuar tareas de diseño más relacionadas con la arquitectura del software.
- En todo el desarrollo, el jefe de proyecto cuenta con una cuidada selección de medidas e indicadores que le permiten conocer y controlar en todo momento el estado real del trabajo.

- Y tantas ideas más que identificamos con «hacer bien las cosas» y que a todos nos gustaría disfrutar ya porque son factibles.

Por último, y aún a riesgo de equivocarme, no pensemos que facilitar el desarrollo de software supone necesariamente un riesgo para el empleo de programadores y analistas. Según los estudios de prospectiva, la carencia de profesionales cualificados en España (que no podrán ser cubiertos con las promociones previstas de titulados) para los próximos años es grave (un 13% sobre el total de profesionales en el año 2003 [Milroy y Rajah, 2000]). Además de esto, se pueden recordar (una especie de historia que no he podido contrastar) algunas opiniones realizadas a raíz de la aparición del lenguaje ensamblador, en las que se proclamaba la desaparición del oficio de programador porque «programar con ensamblador ya sería tan fácil [comparado con la situación anterior] que no haría falta ningún profesional cualificado para ello». En fin, un poco de sentido del humor no es malo para este pequeño artículo de reflexión.

6. Referencias

- [Dolado y Fernández, 2000] J.Dolado y L.Fernández, *Medición para la gestión en la ingeniería del software*, Ra-Ma, 2000.
- [Grand, 1998] M.Grand, *Patterns in Java* (vol. 1 y 2), John Wiley and Sons, 1998.
- [ISO, 1998] ISO, *ISO/IEC TR 15504 Information Technology: software process assessment, Part 1-9*, ISO, 1998.
- [Jacobson et al., 1999] I. Jacobson, J.Rumbaugh y Grady Booch, *The Unified Software Development Process*, Addison-Wesley, 1999.
- [Milroy y Rajah, 2000] A. Milroy y P. Rajah, *Europe's Growing IT skills shortage, Special Report compiled for Microsoft*, IDC, 2000.
- [Novática, 1996] Anónimo, «Si los programadores fueran albañiles...», *Novática*, nº 124, noviembre-diciembre, 1996, p. 77 (<http://www.ati.es/novatica/1996/124/124.html>)
- [Nunamaker et al., 1988] J.F. Nunamaker, M. Chen, E.E. Rudolph y D. Sutherland, «Task force on software productivity and the automated software factory», *Proceedings of the Twenty-First Annual Hawaii International Conference on System Sciences, Vol.IV. Applications Track*, 1988, pp. 271 - 272.
- [Paulk et al., 1993a] Paulk, M.C., García, G.M., Chrissis, M.B. y Bush, M., *Capability maturity model for software, versión 1.1 CMU/SEI-93-TR-24*, Software Engineering Institute y Universidad Carnegie Mellon, febrero, 1993.
- [Rumbaugh et al., 1999] J.Rumbaugh, I. Jacobson y Grady Booch, *The unified modeling language reference manual*, Addison-Wesley, 1999.
- [Shaw, 1994] M. Shaw, «Prospects for an engineering discipline of software» en J.Marciniak (ed.), *Software Engineering Encyclopedia*, IEEE, 1994, pp. 930-940.
- [Zelkowitz y Wallace, 1998] M.V.Zelkowitz y D. Wallace, «Experimental models for validating technology», *IEEE Computer*, vol. 31, nº 5, mayo, 1998, pp. 23-31.

7. Notas

¹ La demanda se ha estimado mediante 12.000 entrevistas a responsables de TIC en empresas europeas. La oferta de profesionales se ha obtenido de las cifras de titulados que sacan las universidades y centros de formación europeos.

² No hay que identificar necesariamente esta denominación con el nombre del área de conocimiento oficialmente reconocida a efectos de organización universitaria.

³ El hecho de que sean pocas las grandes compañías multinacionales de software cuya sede de desarrollo esté en España puede contribuir a la idea de que la ingeniería del software influye más en el desarrollo a medida.

⁴ De hecho, se comprobó que un tercio de las propuestas publicadas en las revistas del IEEE no incluían ningún tipo de validación empírica y otro tercio del total ofrecían validaciones informales de nulo valor.