

La incertidumbre y la ingeniería de software

María Irma Díaz

Una respuesta metodológica al desafío de modificar el pensamiento para enfrentar las condiciones del presente y el futuro.

A comienzos del siglo XX la comunidad científica asistió a la revolución del modelo clásico –de un mundo determinista– al modelo moderno –de un mundo incierto. Sin embargo, el retraso general en la enseñanza ha venido encerrando a la opinión pública dentro del anticuado sistema clásico que no corresponde a la realidad actual [1].

Casi un siglo después, la Unesco, en el marco del proyecto transdisciplinario Educación para un futuro sostenible [2], afirma que “uno de los desafíos más difíciles será modificar nuestro pensamiento de manera que enfrente la complejidad creciente, la rapidez de los cambios y lo imprevisible que caracterizan nuestro mundo”.

Este desafío no puede ser asumido únicamente por el sistema educativo, requiere de la participación activa de todas las disciplinas y profesiones. En este documento se presenta un resumen de cómo la ingeniería de *software* ha ido construyendo una respuesta metodológica a este desafío.

La incertidumbre

Edgar Morin¹, invitado por la Unesco a participar en el proyecto mencionado, formalizó su propuesta en el documento “Los siete saberes necesarios para la educación del futuro” [3], en el que presenta en forma explícita cómo aprender a enfrentar las incertidumbres. Al respecto afirma: “Lo inesperado nos sorprende porque nos hemos instalado con gran seguridad en nuestras teorías, en nuestras ideas y estas no tienen

ninguna estructura para acoger lo nuevo. Lo nuevo brota sin cesar; nunca podemos predecir cómo se presentará, pero debemos contar con su llegada. Y una vez sobrevenga, habrá que ser capaces de revisar nuestras teorías e ideas en vez de dejar entrar por la fuerza el hecho nuevo en una teoría que es incapaz de acogerlo verdaderamente”.

La incertidumbre se presenta en contextos que exhiben las siguientes características en el tiempo: su pasado es inconsistente, su presente ambiguo y su futuro impredecible.

La agilidad

La palabra que generalmente se usa para cualificar una entidad con la habilidad no sólo de enfrentar sino de prosperar en un contexto incierto es la agilidad. Esta habilidad ha sido estudiada de manera amplia por diversas áreas del conocimiento. Veamos como ejemplo el enfoque de dos de ellas.

En la teoría de la complejidad, se estudian los sistemas ágiles definiéndolos como los que tienen la habilidad de modificar su estructura interna y su comportamiento para ser más eficientes y efectivos, considerando la retroalimentación de su ambiente cambiante.

En administración, se consideran negocios ágiles aquellos que ofrecen soluciones de valor agregado para sus clientes configurando sus productos y servicios, mientras se adaptan de manera reactiva a los cambios de su ambiente, innovan en forma proactiva para causar cambios en el ambiente y cooperan de forma oportunista interna y externamente con sus pares para aumentar su competitividad. [5]

Pensamiento ágil

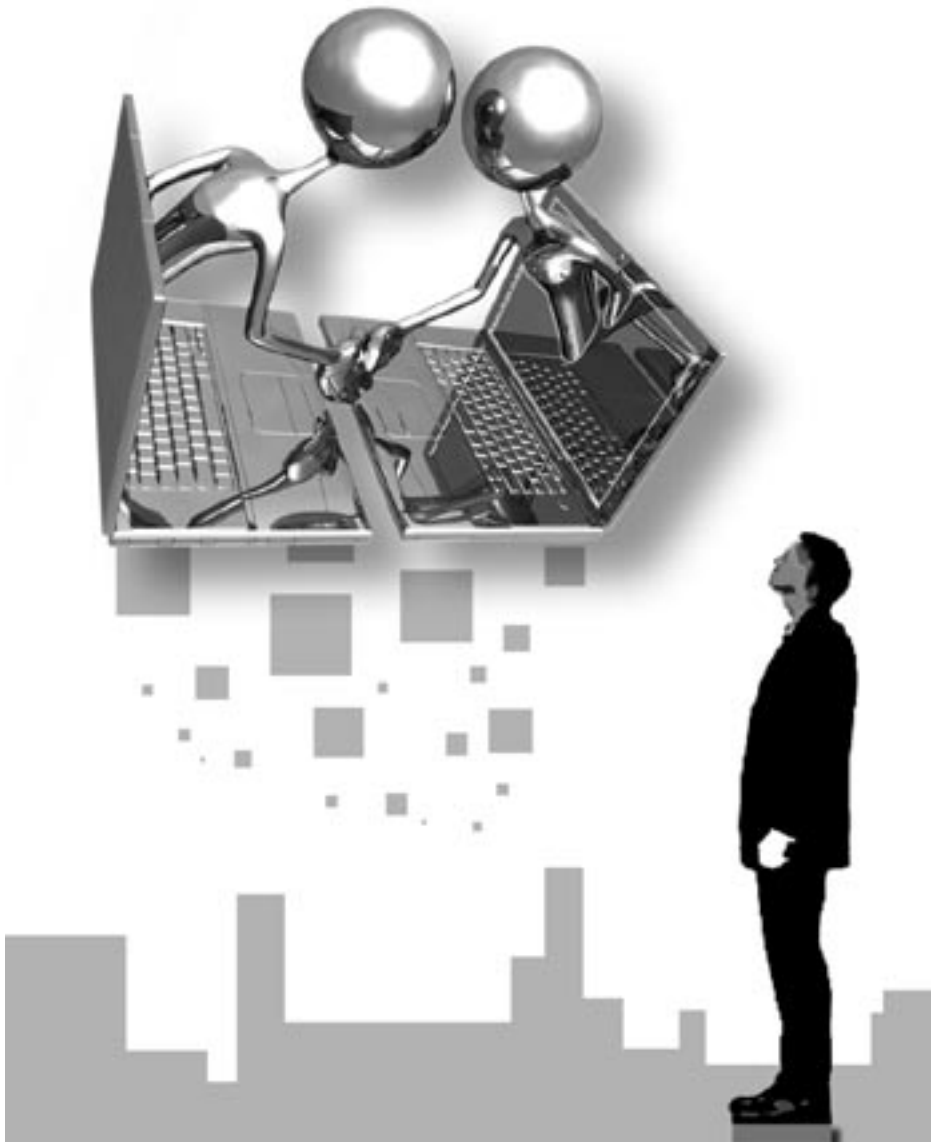
Considerando lo anterior y retomando el desafío planteado por la Unesco, ¿qué condiciones debe cumplir un individuo para tener la capacidad real de enfrentarse a lo incierto? Básicamente tres. En primer lugar, debe ser consciente de que no es posible erradicar la incertidumbre del mundo: el futuro es abierto e impredecible. En segunda instancia, debe comprender la naturaleza básica de lo incierto como resultado de las fuerzas de la complejidad y cambio del universo. Y finalmente, como consecuencia de las dos anteriores, debe poder preparar los medios que le permitan detectar con facilidad la llegada de lo desconocido y adquirir las estrategias para tratarlo, es decir ser ágil. Un individuo así podría no sólo enfrentarse a la incertidumbre, sino aprovecharla para obtener ventajas competitivas y construir

artefactos que exhiban esta misma cualidad.

Ingeniería de software

¿Qué pasa en la ingeniería de *software*? La responsabilidad de esta

profesión es cada vez mayor, toda vez que debe responder a una demanda creciente de sistemas cada vez más complejos y críticos, además de continuar con el mantenimiento de los sistemas existentes. Por otra parte, debe reaccionar a los cambios



que proceden en forma simultánea de múltiples fuentes: requisitos, miembros del equipo de trabajo, metodologías, tecnologías, entre otras. El proceso de desarrollo de *software* se da en contextos discontinuos, ambiguos e impredecibles; es decir, inciertos.

Historia

Hacia 1950 la programación era una actividad caótica resumida en “codifique y arregle”. El *software* era desarrollado sin un plan de soporte y el diseño surgía de decisiones puntuales del proceso de codificación. Esta manera de trabajar funcionó relativamente bien en sistemas simples y pequeños, pero a medida que los sistemas se hicieron más complejos y creció el número de errores encontrados en los productos fue evidente que esa era una aproximación inadecuada.

Como solución a los problemas anteriores, alrededor de 1960 surgieron las metodologías que trataron de imponer un proceso disciplinado en el desarrollo de *software* buscando que esta tarea fuese cada vez más predecible y eficiente. Sin embargo, los resultados no fueron los esperados; muchos proyectos debían ser cancelados y la mayoría de los que se terminaban excedían el presupuesto y no cumplían con el cronograma. El fracaso de esta

iniciativa se explicó por la falta de precisión de las metodologías disponibles o de rigor en su aplicación. Entonces, las metodologías se fueron haciendo cada vez más precisas, pero menos efectivas en la mayoría de los casos.

La mayor crítica que han recibido esas metodologías es que son burocráticas; es decir, se debe invertir una cantidad significativa del tiempo del proyecto en perseguir objetivos paralelos impuestos, lo cual hace que se reduzca el tiempo efectivo para cumplir con el objetivo real: construir *software*. No obstante, hay quienes afirman que el fracaso del llamado “edificio metodológico de la ingeniería de *software* temprana” se debe a que se apoya en una premisa alejada de la realidad del universo, especialmente del universo *software*, en donde se supuso que la incertidumbre era erradicable, siendo inevitable [6]. Es decir, la mayoría de esas metodologías estaban impregnadas del pensamiento del mundo clásico y pretendían ser útiles en un mundo moderno.

Las llamadas “nuevas metodologías” surgieron hacia 1990 en reacción al fracaso de las primeras. Muchos ingenieros de *software* pensaron que simplemente se estaba tratando de responder a una de las críticas evidentes acerca de las metodologías an-

teriores y que buscaban un equilibrio entre los extremos de sin y con mucho proceso que permitiera obtener un beneficio razonable; por ello, se conocieron inicialmente como “metodologías livianas” y las anteriores empezaron a llamarse, por contraste, “metodologías pesadas”. Sin embargo, en estas nuevas metodologías se estaban incorporando características que las habilitaban para responder a las necesidades del mundo moderno entre las que se destacan ser más adaptativas que predictivas y más orientadas a las personas que a los procesos [5].

Nuevas metodologías

De estas nuevas metodologías las más conocidas son las de Programación Extrema (XP), Scrum y Cristal; las cuales comparten características pero se distinguen explícitamente en sus fundamentos. La metodología XP se estructura alrededor de cinco valores: comunicación, realimentación, simplicidad, coraje y respeto. En Scrum, aceptando que los problemas no pueden ser totalmente definidos, el énfasis está en maximizar la capacidad del equipo para responder con entregas frecuentes a los requerimientos que vayan surgiendo en el proceso. Cristal tiene como prioridades la eficacia del producto, la eficiencia del trabajo y la comodidad del ambiente de trabajo; y como principios, la entrega frecuente, el

mejoramiento reflexivo y la comunicación cercana. Vale la pena aclarar que Cristal es realmente una familia de metodologías que permite seleccionar la más apropiada a las características de un proyecto específico.

Adicionalmente se tiene el Proceso Unificado (UP), que aunque es considerado por algunos como una aproximación “pesada”, no puede ser clasificado ya que no es un proceso concreto prescriptivo, sino un marco de proceso adaptable, definido para ser personalizado por los equipos de proyectos de *software*, seleccionando los elementos que se adapten a sus necesidades. Así, se pueden tener versiones UP pesadas, livianas, etc. UP está basado en tres principios: es dirigido por casos de uso, centrado en la arquitectura e iterativo incremental.

Es interesante notar que los fundamentos de UP se refieren a la tecnología y no a las personas a diferencia de lo que ocurre con las tres metodologías anteriores.

Manifiesto ágil

En enero del 2001 se reunió un grupo de diecisiete ingenieros de *software* interesados en compartir sus experiencias para descubrir mejores maneras de desarrollar *software*; los creadores de XP,

Scrum y Cristal participaron en ese encuentro.

El equipo de trabajo consideró interesante sintetizar los acuerdos producto de esta reunión y dar un nombre a esa tendencia; así aparecieron la palabra **ágil**, calificando el desarrollo de *software*, y el conocido Manifiesto para el desarrollo ágil de *software* [7]. En esta declaración se especifica la prioridad que a juicio del equipo debe darse a los diferentes elementos presentes en el desarrollo de *software*: privilegiar individuos e interacciones sobre procesos y herramientas, *software* que funciona sobre documentación exhaustiva, colaboración con el cliente sobre negociación de contratos y respuesta al cambio sobre seguimiento de un plan.

A partir de esta reunión, por obvias razones, XP, Scrum y Cristal son llamadas metodologías ágiles de desarrollo al igual que las nuevas propuestas que acojan el Manifiesto.

Si estas metodologías ágiles se asumieran como una respuesta en construcción de la ingeniería de *software* a la concepción moderna del mundo, que debe ser previamente comprendida y aceptada, podrían ser un buen punto de inflexión en la construcción de propuestas más efectivas para desarrollar *software* que no exijan certezas inalcanzables. En muchos

casos, se han interpretado con la mirada antigua y han sido asumidas simplemente como otra forma de hacer las cosas; o peor aún, como sucede en algunos casos, como una disculpa para no utilizar ingeniería en los procesos de desarrollo de *software*.

Evaluación

Actualmente no existen modelos reconocidos para evaluar la agilidad en el desarrollo de *software*. Sin embargo, cuando se realiza este tipo de evaluación el énfasis no está en los procesos, como ocurre en los modelos tradicionales de valoración, sino en los equipos específicos en acción. Por ejemplo, S. Amber, un consultor en temas ágiles, realiza la evaluación en cuatro etapas.

En la primera de ellas solicita una reunión con sus clientes y examina



el conjunto de pruebas de regresión en ejecución; si el equipo no puede cumplir fácilmente con esos requerimientos, hay una alta probabilidad de que no esté siguiendo una aproximación ágil. Luego solicita ver el *software* en ejecución y el código fuente, que debe ser de alta calidad y estar bajo control de configuración, y el procedimiento para manejar el cambio de requerimientos; si el equipo responde adecuadamente, es casi seguro que está tomando una aproximación ágil.

En la última etapa busca obtener evidencias del estado de liderazgo, empoderamiento y cooperación del equipo de trabajo y el nivel de automatización de su ambiente de desarrollo. [8]

Para evidenciar la relación entre la evaluación propuesta por Amber con el Manifiesto, en la siguiente tabla se presenta la correspondencia entre los valores y cinco de los seis elementos incorporados a la revisión.

Valor	Etapas	Elemento de revisión
Individuos e interacciones	1	Reunión con los clientes
	3	Equipo de trabajo
Software que funciona	1	Pruebas de regresión
	2	Software en ejecución
Colaboración con el cliente	1	Reunión con los clientes
	2	Manejo de cambios de requerimientos
Responder al cambio	1	Reunión con los clientes
	2	Control de configuración Código fuente

Conclusiones

Es indispensable aprender a vivir en un mundo incierto. De ahí la necesidad de que los profesionales conozcan las propuestas que ha desarrollado su disciplina para afrontar la incertidumbre.

Específicamente, se debe ofrecer a los estudiantes y profesionales de Ingeniería de *Software* recursos que les permitan conocer el contexto general de la incertidumbre, la visión

de la misma desde su profesión y los medios que se han desarrollado para manejarla.

Así mismo, se requiere que los ingenieros de *software* estén más dispuestos a explorar las soluciones propuestas por otras disciplinas y ofrecer aquellos elementos de las suyas que consideren importantes y que puedan extrapolarse para trabajos en otras áreas.

Notas de pie de página

¹Filósofo francés. Presidente de la Agencia Europea para la Cultura. Presidente de la Asociación para el Pensamiento Complejo.

Referencias

[1] Sorman, G. (1992), *Los verdaderos pensadores de nuestro tiempo*, Colombia, Seix-Barral, cap. II, pp. 35-54

[2] Organización de las Naciones Unidas para la Educación, la Ciencia y la Cultura (UNESCO), (1997) “Educación para un futuro sostenible” [en línea], UNESCO, disponible en <http://unesdoc.unesco.org/images/0011/001106/110686S.pdf>

[3] Morin, E. (2001) “Los siete saberes necesarios para la educación del futuro”, [en línea], UNESCO, disponible en <http://unesdoc.unesco.org/images/0011/001177/117740so.pdf>

[4] Si Ahir, S. (2005) “The Agil Unified Process” [en línea], disponible en home.comcast.net/~salhir/TheAgileUnifiedProcess.PDF

[5] Flower, M. (2005) “The New Methodology” [en línea], disponible en <http://martinfowler.com/articles/newMethodology.html>

[6] Mendinilla, N. (2005), *En busca de respuestas para la ingeniería de software* [en línea], disponible en <http://is.ls.fi.upm.es/udis/docencia/proyecto/docs/FilosofiaIS.pdf>

[7] Kent, B. et al. (2001), “Manifiesto for Agile Software Development” [en línea], disponible en <http://www.agilemanifesto.org/>

[8] Ambler, S. (2005), “The Agile Edge: How Agile Are You?” [en línea], disponible en <http://www.ddj.com/dept/architect/184415445?cid=Ambysoft>



María Irma Díaz. Máster en Ingeniería de Software de la Universidad Politécnica de Madrid e Ingeniero de Sistemas y Computación de la Universidad de los Andes. Profesora adscrita al Centro de Estudios de Ingeniería de Software de la Decanatura de Ingeniería de Sistemas de la Escuela Colombiana de Ingeniería Julio Garavito. Experiencia en las áreas de Ingeniería de Software e Ingeniería de Conocimiento.