Proyecto Piloto para Conocer el Estado del Arte de la Ingenieria del Software en los Programas Curriculares de Ingeniería de Computadoras en Puerto Rico

Alcides Alvear
Universidad del Turabo, Puerto Rico, PR, aalvear@suagm.edu

Graciela E. Quintero
Universidad del Turabo, Puerto Rico, PR, gquintero2@suagm.edu

ABSTRACT

In today's society, science and technology are a powerful pillar of the cultural, social, and economic development. His influence in everyday life is such that the growing flood of products from both areas flood work environments, households and everyday life, the impact is such that today the systematic and proper handling of these products is an important indicative of modernity. From this perspective, modern society demand for flexible and innovative applications where integration of usability is an important topic for both researchers, creators and designers of software and for those responsible to use the systems, ie, for the users. Therefore, usability and agile methodologies must permeate any academic curriculum that seeks to train engineers capable computer to answer the current and future demands of the knowledge society .
This work is framed in the context of Software Engineering (IS), and aims to demonstrate the importance for future computer engineers in Puerto Rico, in their education to acquire the skills necessary to implement agile methodologies and usability Software development. Preliminarily it is recognized that in Puerto Rico training schools Computer Engineers develop curricula where usability and agile methodologies are not taken into account and if you are a tangential aspect mentioned and no core curriculum. As a result future professionals entering the labor market without the skills to develop software and keep repeating alternately the traditional schemes and often unbeknownst to the needs of the users.

Keywords: Software Engineering, Usability, Agil Methods

RESUMEN

En la sociedad actual, la ciencia y la tecnología constituyen un poderoso pilar del desarrollo cultural, social, económico. Su influencia en la vida cotidiana es tal que la creciente avalancha de productos procedentes de ambas esferas inundan los ambientes de trabajo, los hogares y la cotidianidad; es tal el impacto, que hoy en día el manejo sistemático y apropiado de estos productos constituye un importante indicativo de modernidad. Desde esta perspectiva, la sociedad moderna demanda de aplicaciones ágiles y novedosas donde la integración de la usabilidad sea un tema relevante tanto para investigadores, creadores y diseñadores de software como para las personas encargadas de utilizar los sistemas; es decir, para los propios usuarios. Por tanto, la usabilidad y las metodologías ágiles deben permear cualquier currículo académico que busque formar ingenieros de computadoras aptos para responder las demandas actuales y futuras de la sociedad del conocimiento.
Este trabajo se enmarca en el contexto de la Ingeniería del Software (IS), y pretende demostrar la importancia que tiene para los futuros ingenieros de computadoras en Puerto Rico, adquirir en su proceso formativo las destrezas necesarias para aplicar las metodologías ágiles y la usabilidad en el desarrollo de Software. Preliminarmente se reconoce que en Puerto Rico las escuelas formadoras de Ingenieros de Computadoras, desarrollan currículos donde la usabilidad y las metodologías ágiles no son tenidas en cuenta y si son mencionadas constituyen un

aspecto tangencial y no medular del currículo. Como consecuencia los futuros profesionales ingresan al mercado laboral sin los conocimientos necesarios para desarrollar Software de forma alternativa y continúan repitiendo los esquemas tradicionales y en muchos casos a espaldas de las necesidades de los propios usuarios.

Palabras claves: Ingeniería del Software, Usabilidad, Métodos ágiles

1. INTRODUCTION

La Ingenieria del Software (IS), según definición de la IEEE Std 610.12-1990, es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software. Esta rama del conocimiento provee métodos y técnicas para desarrollar software de calidad que permitan resolver problemas humanos de diferente índole. Trata áreas muy diversas de la informática y de las ciencias computacionales e integra las ciencia y las matemáticas a las propiedades de un software, con el fin de hacerlo útil a las personas y a sus necesidades (Boehm, 2006).

El concepto de IS surgió en 1968, en la conferencia en Garmisch (Alemania) que tuvo como objetivo tratar de resolver los problemas de la denominada "Crisis del software". La crisis representó una condena a la práctica de software, ya que describía un campo de acción en el que no se podía confiar; pues en la mayoría de ocasiones, los productos eran deficientes, no se terminaban a tiempo y excedian el presupuesto asignado para su creación, causando pérdidas millonarias. Además, en muchos casos el software no respondia a las necesidades del usuario, requeria experiencia para su manejo y, el mantenimiento de los productos era complejo y costoso (Yu et. al., 2009; Sommerville, 2011). La crisis tambien permitió a los ingenieros reflexionar sobre la imposibilidad de usar métodos de desarrollo de software informales en proyectos de gran tamaño y complejidad. Tambien fueron conscientes de la necesidad de formar profesionales especializados que fueran capaces de lidiar con la creciente complejidad de los nuevos sistemas (Feller y Fitzgerald, 2000).

Durante décadas, resolver la "Crisis del software" representó el principal reto de compañías e investigadores quienes creían que cada aplicación podría eventualmente resolver la crisis y convertirse en una "bala de plata" que solucionaría todas las deficiencias del software. Planteamiento que finalmente fue desechado en 1986, con la publicación del artículo "No Silver Bullet", donde se argumentó que: "ninguna tecnología o práctica por sí misma mejoraría un 10% la productividad del software en los siguientes diez años". En otras palabras, se planteó que no había nada que permitiera mejorar la calidad del software de manera radical (Brooks, 1986).

Con el transcurso de los años, la búsqueda de una única solución no funcionó, pero tambien los ingenieron fueron concientes que no existían soluciones mágicas en ninguna área del conocimiento y que solo el trabajo duro y riguroso podia hacer que la IS desarrollara productos de calidad.

2. METODELOGÍAS PARA EL PROCESO DE DESARROLLO DE SOFTWARE

En este apartado se analizan las concepciones básicas sobre la manera como se diseña un software desde el punto de vista tradicional y teniendo en cuenta las teorias modernas, centradas en los procesos ágiles.Tradicionalmente, para desarrollar un software se necesitaba un enfoque estructurado cuyo objetivo era facilitar la creación de productos software de alta calidad a un coste razonable. Los métodos abarcanban un amplio espectro de tareas donde se incluyen la planificación y estimación de proyectos, el análisis de los requerimientos del sistema y del software, el diseño de procedimientos algorítmicos, codificación, prueba y mantenimiento (Nielsen, 1993; Pfleeger, et. al., 1990; Sommerville, 2011).

Desde esta perspectiva, aprender a diseñar software significaba pasar por un proceso o "un conjunto de etapas parcialmente ordenadas con la intención de lograr un objetivo; en este caso, la obtención de un producto software de calidad". Durante el proceso las necesidades del usuario eran traducidas en requerimientos de software, estos requerimientos se transforman en diseño, el diseño implementado en código, el código era probado, documentado y certificado para su uso operativo" (Jacobsen, 1998). A este proceso también se le llama el ciclo de vida del software, que comprende las etapas por las que pasa un proyecto software desde que es concebido, hasta que está listo para usarse (Ferré, 2004).

De igual manera, bajo esta concepción se especifican cuatro actividades fundamentales comunes a todo proceso software: *i*. Especificación: usuarios e ingenieros definen el software a producir y las restricciones en su funcionalidad. *ii.* Desarrollo: fase en la cual el software se diseña y se programa. *iii*. Validación: el software debe ser probado para asegurar que cumple con las necesidades del cliente. *iv*. Evolución: el software debe poder ser modificado para adaptarse a cambios en el mercado y en las necesidades de los usuarios.

Cada producto software necesita un proceso diferente. Por tanto, estas etapas genéricas deben organizarse de diferente manera y en diferentes niveles según el tipo de software para el que se aplique el proceso. Un uso inapropiado del proceso software puede reducir la calidad o la usabilidad del producto a ser desarrollado, e incluso incrementar los costes de desarrollo.

A través del tiempo han surgido diferentes modelos para el desarrollo del software los más usados se resumen a continuación:

Modelo en cascada: La primera descripción formal de este modelo fue efectuado por Royce, en 1970 (Royce, 1970), ordena rigurosamente las etapas del ciclo de vida del software, de tal forma que el inicio de cada etapa debe esperar la finalización de la inmediatamente anterior. Algunos de los inconvenientes del modelo fueron presentados por (McCraken y Jackson, 1981) y señalan que el modelo impone una estructura de gestión de proyecto al desarrollo del sistema. Pero a la fecha se han realizado multiples mejoras (Boehm, 1981; Sommerville, 2011; Sigwart et. al., 1990).

Especificación operacional: Fue propuesto por Zave en1984, propone un modelo de proceso que permite a los desarrolladores y los clientes examinar rápidamente los requerimientos y sus implicaciones lo más temprano posible en el proceso de desarrollo. Para este modelo, los requerimientos del sistema, son evaluados o ejecutados en una forma que demuestra el comportamiento del sistema en el tiempo (Pfleeger y Atlee, 2010).

Incremental e iterativo: El sistema es particionado en subsistemas de acuerdo a su funcionalidad. Las versiones se definen comenzando con un subsistema funcional pequeño y agragando funcionalidad con cada nueva versión. En cada incremento, una parte de la funcionalidad es desarrollada, desde el análisis hasta las pruebas (Pfleeger y Atlee, 2010).

Espiral: combina los procesos en cascada y prototipado. Fue definido por Barry Boehm en 1988. El proceso inserta un paso para evaluar riesgos y construir prototipos de las alternativas, antes de escribir un documento sobre el más alto nivel en el que se espera pueda trabajar el sistema (Pfleeger y Atlee, 2010).

RAD (Rapid Application Development): emplea técnicas iterativas y de prototipado. Fue introducido por Martin en 1991. Según este modelo se puede hacer un desarrollo de software en poco tiempo, normalmente, en periodos menores de o iguales a 60 días. Se trabaja en grupos pequeños que incluyen en algunos casos a los propios clientes, con su participación el grupo modela las aplicaciones y las va haciendo funcionales de acuerdo a la orientación directa del cliente (Martin, 1991).

RUP (Rational Unified Process): es un proceso de desarrollo de software iterativo y junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización. Captura varias de las mejores prácticas en el desarrollo moderno de software que es aplicable para un amplio rango de proyectos y organizaciones.

Metodo orientado a objetos (MOO): desarrollado en 1987 por Jacobsen. Este método permitió el desarrollo de un lenguaje estándar el UML (Unified Modeling Language). La MOO se basa en tres principios básicos: todo son objetos, encapsulamiento / ocultación y herencia / polimorfismo. El primer principio indica la unidad básica de trabajo. El segundo permite englobar en un mismo concepto a los datos y a las operaciones. El tercero permite agrupar y tratar de igual forma a objetos similares.

A lo largo de los años se han ido desarrollando distintas metodologías para desarrollar software, a menudo vinculadas a algún tipo de organización, que además desarrolla, apoya su uso y la promueve. La metodología es documentada formalmente y ofrecida al público implicando costos por su uso.

Recientemente, la IS ha evolucionado hacia maneras más ágiles y dinámicas en los procesos de construcción del software; centrándose en dimensiones relacionadas con el factor humano. Esta es la filosofía de las metodologías ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Este enfoque está mostrando su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad. Las metodologías ágiles están revolucionando la manera de producir software, y a la vez generando un amplio debate entre sus seguidores y quienes por escepticismo o convencimiento no las ven como alternativa para las metodologías tradicionales (Hazzan y Dubinsky, 2010).

Desde esta óptica, se consideran metodologías como:

Metodologías ágiles: En febrero de 2001, tras una reunión celebrada en Utah-EEUU, nace el término ágil aplicado al desarrollo de software. En esta reunión participan un grupo de 17 expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto (Abrahamsson et. al., 2002)

Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas. Metodologías como "Scrum" (Schwaber, 1995), DSDM (Stapleton, 1997) o "Extreme Programming" (Beck, 1999) hacen parte de los procesos que permiten construir software eficientes y en poco tiempo.

Scrum: desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración (Schwaber et. al., 2001).

DSDM: define el marco para desarrollar un proceso de producción de software. Nace en 1997 con el objetivo de crear una metodología RAD unificada. Sus principales características son: es un proceso iterativo e incremental y el equipo de desarrollo y el usuario trabajan juntos. Propone cinco fases: estudio viabilidad, estudio del negocio, modelado funcional, diseño y construcción, y finalmente implementación. Las tres últimas son iterativas, además de existir retrolimentación a todas las fases (Stapleton, 1997)

Extreme Programming (XP) es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico (Jeffries et al., 2001)

3. DISTRIBUCIÓN DE OFRECIMIENTOS DE INGENIERÍA DE COMPUTADORAS EN PUERTO RICO

En este apartado se analiza la manera como se abordan la IS en algunas universidades de Puerto Rico y pretende determinar si el grado de capacitación que reciben los estudiantes es el necesario y suficiente para abordar los retos a los que se enfrentan en el campo laboral. En Puerto Rico, el primer programa de Ingeniería de Computadoras inició en el año 1980 en la Universidad de Puerto Rico Recinto de Mayagüez y la primera clase se graduó en mayo de 1983. En fechas posteriores, las universidades privadas abrieron ofrecimientos en ingeniería de computadoras, siendo la segunda en creación la Universidad Interamericana de Bayamón (1995), la Universidad Politécnica (2002) y la Universidad del Turabo (2007). Todas estas universidades se han dado a la tarea de proveer a la Isla con los recursos humanos en Ingeniería de Computadoras necesarios para su desarrollo.

En lo que respecta a la IS, los currículos de estas cuatro universidades son bastantes similares, en todas se ofrece un solo curso llamado "Software Engineering" o Diseño y Construcción de Software, cuyas descripciones de ofrecen a continuación: en la Universidad de Puerto Rico (UPRM**):** el curso figura dentro de la oferta del programa como un curso electivo, es ofrecido en el campus de Mayagüez en el cuarto año, tiene 3 créditos, 3 horas contacto y es un curso tipo lectura; es decir, no incluye una práctica de laboratorio y tiene como prerrequisito el curso de estructuras de datos. La descripción que hace la universidad de este curso es la siguiente: "Técnicas utilizadas durante el ciclo de desarrollo de software; especificación, diseño, pruebas, documentación y mantenimiento. Uso de un procedimiento orientado a un lenguaje en el diseño e implementación de un proyecto de software" (2014, uprm.edu).

La Universidad Interamericana de Bayamón (UIB): No ofrece una descripción muy detallada de su curso Diseño y Construcción de Software, no obstante; se indica que tiene 3 créditos, 3 horas contacto y es un curso tipo lectura con soporte práctico o laboratorio (2014, bc.inter.edu).

En la Universidad Politécnica de Puerto Rico (UPPR): El curso es ofrecido en el cuarto año, tiene 3 créditos y 3 horas contacto, es un curso tipo lectura; la descripción que hace la universidad del mismo es la siguiente: "El estudiante que toma este curso requiere estar familiarizado con las aplicaciones de Windows o Unix, y el conocimiento de la programación orientada a objetos avanzados. Un ciclo de desarrollo de software completo se ejecuta en un proyecto de pequeña escala. Se discuten en detalle técnicas de análisis orientado a objetos, diseño, codificación y pruebas utilizando UML. Herramientas de apoyo a métodos de ingeniería de software para planificación de proyectos, gestión de la configuración del software y desarrollo orientado a objetos son demostradas y utilizadas por los estudiantes para crear productos de trabajo de ingeniería de software (pupr.edu, 2014).

Finalmente, en la Universidad del Turabo (UT)**:** el curso es del cuarto año, tiene 3 créditos, 3 horas contacto, es un curso tipo lectura, pero incluye el desarrollo de un proyecto mediante el uso y aplicación de una metodología ágil como eXtremme Programming, Scrum o Kamban. La descripción que hace la universidad del curso es la siguiente: "Técnicas utilizadas durante el ciclo de desarrollo de software; especificación, diseño, pruebas, documentación y mantenimiento. Diseño, implementación y gestión de un proyecto software". Los tópicos que cubre este curso son: Introducción a Sistemas e Ingeniería del Software, Procesos y modelo del ciclo de vida, requerimientos y manejo de proyectos en IS, Modelamiento de sistemas y diseño arquitectural, diseño orientado a objetos, Desarrollo de software y pruebas, Sistema de pruebas, Entrega y mantenimiento del sistema, Evaluación de productos, procesos y recursos, Metodologías ágiles de desarrollo de software (suagm.edu/Turabo, 2014).

De acuerdo a la información anterior, se puede observar que existe un hilo conductor que subvalora la IS como herramienta indispensable para la formación de Ingenieros de Computadoras en Puerto Rico, aspecto que probablemente se debe a que la mayoría de instituciones universitarias toman como ejemplo los currículos que se desarrollan en la universidad pública y a partir de ellos diseñan los que ofrecerá la universidad privada; es decir, si Mayagüez por alguna razón determina que un curso es suficiente para abordar este tópico, quizá las demás universidades, con menos experiencia, pero con deseos de ofrecer una alternativa académica a los estudiantes que por diversas razones no pueden acceder a la universidad pública, la tomen como primera opción frente a posibilidades diferentes.

Por otra parte, se reconoce que aunque los programas de Ingeniería de Computadoras en Puerto Rico, no están diseñados para formar Ingenieros Software, resulta bastante improbable que, con un solo curso (en ocasiones electivo), los estudiantes puedan tener una aproximación real al tema y a la importancia que el mismo reviste para su formación profesional. Como se deduce, la carencia de formación en IS, hace que los futuros ingenieros de computadoras se enfrentan a situaciones en las que quizá solo estén capacitados para cierto tipo de empleo, donde ese componente no sea de peso, o recurran a proceso de educación no formal, que en términos generales son realizados por compañías que se lucran capacitando en procesos que bien pudieron ser abordados durante su proceso formativo.

Las metodologías ágiles son sin duda uno de los temas recientes en IS que están acaparando gran interés. Prueba de ello es que se están haciendo un espacio destacado en la mayoría de conferencias y workshops celebrados en los últimos años. Es tal su impacto que actualmente existen 4 conferencias internacionales de alto nivel y específicas sobre el tema. Además ya es un área con cabida en prestigiosas revistas internacionales.
En adición, en momentos en los que la tecnología representa el principal paradigma de modernidad; el tiempo, el espacio y el dinero son recursos a los que se debe sacar el mayor provecho posible. Por esa razón, currículos en donde las metodologías ágiles son tangencialmente o no abordadas, resultan replicando perfiles profesionales obsoletos, poco ajustados a realidad y a las necesidades de los tiempos modernos.

4. EFECTO DEL DEFICIT DE FORMACION EN IS EN LOS PROFESIONALES DE INGENIERIA DE COMPUTADORAS

En las escuelas de ingeniería de Puerto Rico, se ha experimentado un déficit en la cantidad de estudiantes que aspiran a convertirse en ingenieros de computadoras; es decir, esta profesión no es la más solicitada por los estudiantes de nuevo ingreso, en las distintas universidades que ofrecen el programa. En adición, como se observa en la Tabla 1, son relativamente muchos los estudiantes que inician el proceso como se muestra en la columna uno, pero pocos los que se mantienen y logran adquirir el título universitario. En la misma tabla, se observa que por ejemplo la UT enroló 768 estudiantes en ingeniería, en el mismo período se graduaron 80 y de estos solo 7 pertenecían a ingeniería de computadoras, dato que corresponde al 8.75% del total de graduados en la universidad. Es importante resaltar en esta tabla, que la UIB no tiene datos reportados a ASEE.

Tabla 1. Estudiantes enrolados/graduados y graduados en ingeniería de computadoras

Universidad	Estudiantes enrolados en ingeniería	Grados otorgados en ingeniería	Grados otorgados en ingeniería computadoras	Relación porcentual ingeniería computadoras y el total
UPRM	4380	346	63	18.20%
UPPR	3120	609	34	5.58%
UT	768	80	7	8.75%
UIB	-	-	-	-

Fuente: (ASEE, 2013; VPEEI, 2014)

La situación que se plantea, se agrava en Puerto Rico, donde la aspiración de muchos profesionales no es la de quedarse en la Isla, sino la de emigrar, típicamente a Estados Unidos, en busca de mejores trabajos y mejores ofertas salariales. Si bien es cierto, que los esquemas sociales son difíciles de transformar, también es cierto que las entidades educativas no deben estar a espaldas de la realidad socioeconómica de un país. Quizá por esta razón hoy en día, se cuenta con una gran oferta formativa y diversificada para profesiones, que si bien es cierto no se enfoca 100% en IS, tienen alguna afinidad con ese campo del conocimiento.

Paradójicamente, en 2006 "Money Magazine and Salary" publicaron en su revista que IS era una de las carreras de mayor auge en América en términos de crecimiento, remuneración, nivel de estrés, flexibilidad horaria, creatividad, entorno de trabajo y capacidad de ascenso; ocho años después en febrero de 2014, el "US NEWS" continua afirmando que "Software Developer y Computer Systems Analyst" resultan siendo las habilidades mejor remuneradas en Estados Unidos, ambas directamente relacionados con la IS.

Todo apunta a señalar que la información anterior aunque es de dominio público, no es difundida ni promocionada de la mejor manera, pues ni los estudiantes que se encuentran actualmente cursando ingeniería de

computadoras ni mucho menos los que decidieran por esta profesión en el futuro, son conscientes de las ventajas y los retos que plantea esta profesión en la actualidad y en el futuro mediato.

Por otra parte, la situación puertorriqueña no hace diferencia con la que se plantea en otras latitudes, donde se plantea una crisis mundial para esta área del conocimiento. Al respecto, Serna y Serna en el 2013 determinaron que la crisis se debía principalmente a: falta de profesionales capacitados; a los estudiantes de nuevo ingreso que no quieren tomar estas carreras; a las universidades que carecen de dinamismo para actualizar los programas; a los profesores que no cuentan con la experiencia profesional apropiada y la falta de actualización de los docentes en áreas de desarrollo más novedosas, ágiles y contemporáneas (Serna y Serna, 2013).

Hasta ahora, se ha tratado de señalar el efecto de la carencia de programas curriculares que enfoquen su contenido hacia la IS y cómo esta carencia, aunque no se perciba, tiene un impacto nocivo, ya que se continúa formando profesionales con perfiles que quizá no correspondan a las demandas sociales, culturales y hasta económicas de la sociedad.

El siguiente paso de este proyecto piloto, consistió en indagar qué tipo de procedimientos se realizan en el proceso de diseño y desarrollo de software en algunas de las compañías establecidas en Puerto Rico, con este fin se realizó una encuesta a seis empresas identificadas con las letras "N, H, P, I, B, M" como se muestra en la tabla 2 (no se dan los nombres reales de las empresas por razones de la ley HIPPA); algunas de estas son totalmente Puertorriqueñas y otras son americanas, con sede en Puerto Rico desde hace varios años.

La empresa N, es una empresa que se dedica al diseño de "web pages" y los dominios de internet .pr utilizados en páginas como www.google.com.pr y www.pr.gov. La empresa H, es una empresa dedicada a la creación de modelos para aplicaciones de procesamiento digital de señales (DSP – Digital Signal Processing). La empresa P, se dedica al desarrollo de software de tipo didáctico y ofrece sus servicios al Departamento de Educación de Puerto Rico. La empresa I, se dedica al diseño y suministro de soluciones para la prevención de infecciones en los centros médicos, teniendo en cuenta la asistencia sanitaria y la seguridad de las personas que los frecuentan. La empresa B, se dedica al diseño y desarrollo de un software de gestión de proyectos que permite a los usuarios disponer de una herramienta para clasificar, programar y manejar sus tareas de forma eficiente. La empresa M, no se dedica exclusivamente a la creación de software, pero tiene un departamento de diseño y desarrollo de software relacionados con la construcción de dispositivos médicos para tratamiento de enfermedades vasculares, diabetes, trastornos neurológicos y músculo-esqueléticos. Como información adicional, en la tabla 2 se especifica el número de empleados que trabajan en cada empresa entrevistada, con el fin de determinar si el criterio de extensión puede o no ser significativo a la hora de optar por un proceso particular en el diseño de software.

Tabla 2. Cantidad de empleados de cada empresa encuestada

Empresa	Cantidad Empleados
N	80
H	10,000 +
P	5
I	500 +
B	100 +
M	46,000

La encuesta aplicada tuvo como objetivo determinar qué tipos de metodología de desarrollo utilizan las empresas, se entrevistaron cuatro personas de cada empresa, la encuesta fue replicada de estudios preliminares realizados por Ferré en la Universidad Politécnica de Madrid (Ferré, 2003).

Una de las preguntas formuladas en la encuesta permitía al entrevistado seleccionar diferentes técnicas de usabilidad y los momentos durante el desarrollo del software en las que ellas eran de mayor utilidad. En la tabla 3, se resumen los resultados obtenidos en la pregunta 1 de la entrevista. Para especificar mejor el momento, A correspondió a la definición de requerimientos funcionales; el B a elicitación y definición de especificaciones técnicas; C al diseño y D al análisis de comentarios y sugerencias. Como se observa, los resultados variaron poco en las 6 empresas entrevistadas, por ejemplo: todas determinaron que las técnicas de usabilidad *observacional, lluvia de ideas, escenarios, diagramas de afinidad, análisis de competencias y prototipos* son de suma utilidad sobre todo durante las fases de elicitación y diseño de software. Por el contrario, las técnicas de *personas, elaboración de tutoriales, rutas de navegación, evaluación heurística, pruebas, medidas y cuestionarios de usabilidad* no son seleccionadas como importantes y solo algunas empresas las usan en algunas fases del desarrollo del software.

Tabla 3. Técnicas de usabilidad más utilizadas para el desarrollo del software, en las 6 empresas entrevistadas. Los números indican la cantidad de empresas cuyos entrevistados respondieron que la técnica era de utilidad durante alguna fase del desarrollo del software (A: requerimientos funcionales, B elicitación, C diseño, D análisis de comentarios y sugerencias)

Técnicas de usabilidad	Actividades			
	A	**B**	**C**	**D**
Técnica Observacional	3	3	6	2
Lluvia de ideas	3	3	5	3
Escenarios	3	3	4	2
Diagramas de afinidad	2	4	3	1
Análisis competitivo	2	4	5	5
Personas	2	2	0	1
Análisis de caso	3	1	3	0
Prototipos de papel	2	3	6	0
Especificaciones de usabilidad	3	0	2	1
Elaboración de tutoriales	0	0	1	1
Ordenamiento de tarjetas	1	0	2	0
Arboles de sitios Web	1	0	0	1
Rutas de navegación	0	0	1	0
Guía de productos	3	2	1	2
Evaluación heurística	0	0	1	2
Pensando en voz alta	2	0	0	0
Pruebas de usabilidad con medidas	1	0	2	1
Cuestionarios de usabilidad	1	0	0	0

Los resultados tienen mucho sentido si se tiene en cuenta que las técnicas más utilizadas por los profesionales coinciden con las que aprenden en los cursos de ingeniería del software; sin embargo, no ocurre lo mismo con técnicas de usabilidad ya que estas sólo se mencionan en la clase, pero no se hacen prácticas o ejemplos que refuercen el aprendizaje de estos temas.

5. CONCLUSIONES

No existe una metodología universal para hacer frente con éxito a cualquier proyecto de desarrollo de software. Toda metodología debe ser adaptada al contexto del proyecto (recursos técnicos y humanos, tiempo de desarrollo, tipo de sistema, etc.). Históricamente, las metodologías tradicionales han intentado abordar la mayor cantidad de situaciones de contexto del proyecto, exigiendo un esfuerzo considerable para ser adaptadas, sobre todo en proyectos pequeños y con requisitos muy cambiantes. Las metodologías ágiles ofrecen una solución casi a la

medida para una gran cantidad de proyectos que tienen estas características. Una de las cualidades más destacables en una metodología ágil es su sencillez, tanto en su aprendizaje como en su aplicación, reduciéndose así los costos de implantación en un equipo de desarrollo.

En Puerto Rico, la Ingeniería de Computadoras es una disciplina joven en constante evolución, pese a que no se ha enfocado en los procesos de IS, sus egresados de alguna manera desarrollan productos de alta calidad a nivel mundial, tal es el caso de una de las empresas entrevistadas que se dedica a la fabricación de dispositivos biomédicos; en esta empresa, el manejo y desarrollo del Software es indispensable para asegurar productos de alta calidad. Como se espera, en la medida que los egresados tengan un perfil profesional que mejor se adapte a las necesidades de la industria, mejor será también su desempeño.

De ahí la necesidad de proponer una modificación curricular a los programas de Ingeniería de Computadoras, para que tengan en cuenta la pertinencia de formar egresados con mejores conocimientos y destrezas en el área de IS, estas destrezas deben estar acorde a las necesidades del país, de tal manera que los futuros egresados y posibles empleados tengan la formación suficiente para no solo generar productos software de alta calidad sino responder a las demandas de la sociedad tecnológica, ágil y cambiante.

Con los métodos tradicionales pasa un gran lapso de tiempo antes que el cliente vea resultados, esto conlleva a que las metodologías ágiles sean vistas por las compañías de desarrollo como una alternativa que se puede utilizar en gran cantidad de tipos de proyectos que permiten entregas con mayor frecuencia y por consiguiente un cliente más satisfecho.

REFERENCIAS

American Society for Engineering Education – ASEE. (2013). "Profiles of Engineering & Engineering Tchnology Colleges. ISBN 978-0-87823-240-6.

Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J. "Agile software development methods Review and analysis". VTT Publications. 2002.

Beck, K. (1999). "Extreme Programming". Addison-Wesley.

Boehm B.. (1981). "Software Engineering Economics". Englewood Clifs, Nueva Jersey.

Boehm, B. W. (1988). "A spiral model for software development and enhancement". IEEE Computer, 21(5), 61-72.

Boehm, B. (2006). "A View of 20th and 21st Century Software Engineering". Keynote speech at ICSE 2006.

Brooks, F. P. Jr (1986). No Silver Bullet Essence and Accident in Software Engineering, Proceedings of the IFIP Tenth World Computing Conference, H. J. Kugler, ed., Elsevier Science B.V., Amsterdam, NL(1986) pp. 1069-76.

Feller, J., Fitzgerald, B. (2000). "A Framework Analysis of the Open Source Software Development Paradigm". 21st Annual International Conference on Information Systems. Brisbane, Australia.

Ferré, X., Moreno A. M. (2004). "Integración de la IPO en el Proceso de Desarrollo de la Ingeniería del Software: Propuestas Existentes y Temas a Resolver". Interacción-2004, Lleida, España.

Ferré, X. (2003). "Integration of Usability Techniques into the Software Development Process". Universidad Politécnica de Madrid. 2003.

Hazzan, O., and Dubinsky, Y. (2010). "A HOT --- Human, Organizational and Technological --- framework for a software engineering course". ACM/IEEE. Volume 1 (ICSE '10).

IEEE. (1990). Std 610.12-1990. "IEEE Standard Glossary of Software Engineering Terminology".

Jacobsen, I. (1998). "Object Oriented Software Engineering. Addison-Wesley.

Jeffries, R., Anderson, A., Hendrickson, C. "Extreme Programming Installed". Addison-Wesley. 2001

Martin, J. (1991). "Rapid application development". Macmillan Publishing Co., Inc. Indianapolis, IN, USA.

McCraken, D.D., Jackson M.A. (1981). "A minority disseting option". W.W. Cotterman *et al*. (comps.). *System Analysis and Design: A foundation for the* 1980s, 551-553. New York: Elsevier.

Money Magazine. (2006). http://money.usnews.com/money/careers. (02/26/2014).

Nielsen J. (1993). "Usability Engineering. AP Professional". Boston (MA), USA, 1993.

Pfleeger, S. L., McGowan C. L. (1990). "Software metrics in the process maturity framework". Journal of systems and software. 12(1), 255-261.

Pfleeger, S. L., Atlee, J. M. (2010). "Software Engineering". Prentice-Hall. Fourth Edition.

Royce W. E. (1970). "Managing the development of large software systems: concepts and Techniques". Proceedings, Wescon.

Schwaber K., Beedle M., Martin R.C. (2001). "Agile Software Development with SCRUM". Prentice Hall.

Schwaber, K. (1995). "Scrum Development Process". OOPSLA'95 Workshop on Business Object Design and Implementation. Springer-Verlag.

SEI. (2014). http://www.sei.cmu.edu/index.cfm. (02/28/2014).

Serna, E. M., Serna A. A. (2013). "Está en crisis la Ingeniería en el mundo? Revisión a la literatura". Fac. Ing. Univ. Antioquia N.° 66 pp. 199-208.

Sigwart C. y col. (1990). "Software Engineering: a project oriented approach". Franklin, Beedle y Assocites, Inc., Irvine, California.

Sommerville, I. (2011). "Ingeniería de Software". Addison Wesley. Novena Edición.

Stapleton, J. (1997). "Dynamic systems development method - The method in practice. Addison-Wesley.

Universidad del Turabo. (2014). "Oficina de estadísticas institucionales". http://www.suagm.edu/turabo, (02/28/2014).

Universidad del Turabo UT. (2013). "Universidad del Turabo, School of Engineering, Data Book".

Universidad de Puerto Rico. (2014). http://www.ece.uprm.edu, (02/28/2014).

Universidad Interamericana de Bayamón. (2014). http://www.bc.inter.edu, (02/28/2014).

Universidad Politécnica de PR. (2014). http://www.pupr.edu, (02/28/2014.

US New. 2014). http://money.usnews.com/careers/best-jobs/rankings/the-100-best-jobs, (03/01/2014).

VPEEI. (2014). "Oficina de estadísticas institucionales". ©Suagm.

Yu, X. L., Li, J., Zhong, H. (2009). "Extending the Boundary of Spreadsheet Programming: Lessons Learned from Chinese Governmental Projects". SEEUP 2009.

Zave, P. (1984). "The operational versus the conventional approach to software development". ACM, 27(2), 104-118.

Authorization and Disclaimer

Authors authorize LACCEI to publish the paper in the conference proceedings. Neither LACCEI nor the editors are responsible either for the content or for the implications of what is expressed in the paper.