UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Revista Vínculos

http://revistas.udistrital.edu.co/ojs/index.php/vinculos

Revista VÍNCULOS
Ciencia, Tecnología y Sociedad: Un Enlace hacia el Futuro

ACTUALIDAD TECNOLÓGICA

Ingeniería de software: evolución curricular

Software Engineering: Curricular Evolution

Sandra Chuquín-Badillo* Jorge E. Otálora-Luna**

Para citar este artículo: S. Chuquín-Badillo y J. Otálora-Luna (2015). Ingeniería de software: evolución curricular. *Revista Vínculos*, 12(1), 70-79.

Recibido: 12-febrero-2015 / Modificado: 26-febrero-2015 / Aprobado: 10-abril-2015

Resumen

Este artículo hace un recorrido por la evolución del currículo de la *ingeniería de software*, destacando los aspectos de mayor relevancia en cada uno de los momentos de la historia con el fin de conocer las técnicas y estrategias utilizadas en las diferentes décadas, desde que fue utilizado por primera vez el término en 1968, y así identificar los elementos claves que se deberían tener en cuenta en un currículo académico en las áreas relacionadas con esta disciplina. A través de la revisión de la literatura identificada sobre la enseñanza de la ingeniería de software se observó que se realizaron diversas propuestas curriculares, pero fue solo hasta la década del 2000 cuando se unificaron criterios, publicándose la guía SWEBOK, documento que recopila los esfuerzos realizados para definir el cuerpo de conocimiento. La guía tuvo una reciente actualización a la versión 3.0 en el año 2014.

Palabras clave: Enseñanza, currículo, ingeniería, software, SWEBOK.

Abstract

This paper takes a journey through the evolution of teaching software engineering, highlighting the most important aspects in each moment of history, in order to learn the techniques and strategies used in different decades since It was first used the term in 1968, and thus identify the key elements that should be considered in an academic curriculum in areas related to the subject. Through the review of the identified literature on teaching software engineering it was observed that various curricular proposals were made, but it was only until the 2000s when it was possible to unify criteria, publishing the SWEBOK guidance document compiles efforts made to define the body of knowledge. The guide had a recent update to version 3.0 in 2014.

Keywords: Teaching, curriculum, engineering, software, SWEBOK.

* Ingeniera de sistemas. Aspirante a título de magíster en tecnología Informática, Universidad Pedagógica y Tecnológica de Colombia, Colombia. E-mail: spchuquin@yahoo.com, sandra.chuquin@uptc.edu.co.

** Ingeniero de sistemas. Magíster en ingeniería. Docente en la Facultad de Ingeniería, Universidad Pedagógica y Tecnológica de Colombia, Colombia. E-mail: jorge.otalora@uptc.edu.co.

ISSN 1794-211X • e-ISSN 2322-939X • Vol 12, No 1 (enero-junio 2015). pp. 70-79. Universidad Distrital Francisco Jose de Caldas-Facultad Tecnologica.

1. INTRODUCCIÓN

El término software fue acuñado en 1958 por el estadístico John Turkey y el término *ingeniería de software* fue utilizado por primera vez en una conferencia de la OTAN, llevada a cabo en Alemania en 1968 [1]. En los 57 años que han transcurrido desde el nacimiento de la ingeniería de software, la enseñanza de esta disciplina se ha visto en la necesidad de evolucionar para adaptarse a los cambios tecnológicos cada vez más variados y complejos, llevando a organizaciones y entes educativos a la mejora de sus currículos académicos.

Este artículo tiene como objetivo presentar una revisión del currículo de enseñanza utilizado para impartir los conocimientos propios de la ingeniería de software a través de las diferentes décadas desde 1960. Se realizó mediante una investigación de tipo documental [2] haciendo uso del método analítico-sintético [2] con el fin de ahondar sobre la pregunta ¿cuál ha sido la evolución del currículo de la ingeniería de software desde que esta disciplina fue nombrada por primera vez con este término? Se originó de la necesidad de apoyar la investigación realizada en la Universidad Pedagógica y Tecnológica de Colombia (UPTC) denominada "Evaluación de estrategias metodológicas y propuesta de infraestructura para la enseñanza de diseño y construcción de software" [3].

En un breve recorrido a través de los momentos claves de la evolución de la tecnología se observa que en sus comienzos se crearon las grandes máquinas o *mainframes* donde la preocupación principal era crear algoritmos para ejecutar cálculos. Con la creciente demanda de software, ocurrió la denominada crisis del software, que llevó a la reflexión sobre la importancia de crear metodologías para lograr un desarrollo correcto y comenzaron a nacer los tradicionales modelos, como por ejemplo el popular modelo de desarrollo de software en cascada [4-5], seguido por el incremental, el evolutivo y el iterativo [5].

El desarrollo se desglosa a través de cuatro capítulos: introducción, metodología, resultados y conclusiones.

2. METODOLOGÍA

Se realizó una exploración bibliográfica a través de Scopus [6] con el fin de determinar la producción literaria relacionada con el tema de interés, utilizando como criterios de búsqueda las palabras clave *software engineering teaching*, con fechas entre 1968 y mediados de 2015. De los documentos identificados se seleccionaron los que se consideraron más pertinentes, extractando el contenido relevante y presentándolo por décadas en el desarrollo histórico de la sección de resultados.

3. RESULTADOS

De la consulta realizada a través de Scopus [6] se obtuvieron 8.655 documentos, ver tabla 1.

Año	Número de documentos
1968-1979	25
1980-1989	371
1990-1999	1230
2000-2009	3741
2010-2015	3288

Tabla 1. Producción literaria sobre enseñanza de la ingeniería de software. [6].

3.1. Desarrollo histórico

A continuación se describirá la evolución curricular de la enseñanza de la ingeniería de software y los aspectos relevantes en cada una de las décadas, iniciando en 1960 y finalizando en 2015.

3.1.1. Década de 1960

Las primeras publicaciones relacionadas con la ingeniería de software se enfocaban hacia el uso de algunas herramientas aplicadas a otras áreas como la economía [7-8], la estadística [9] y la automatización de procesos repetitivos. Entre 1955 y 1965 [10], después de la segunda guerra mundial, se

ISSN 1794-211X • e-ISSN 2322-939X • Vol 12, No 1 (enero-junio 2015). pp. 70-79. Universidad Distrital Francisco Jose de Caldas-Facultad Tecnologica.

incrementó la demanda de computadores y los fabricantes deseaban satisfacer a sus clientes haciéndoles creer que los desarrolladores también estarían disponibles en el momento en que fueran requeridos, generando a su vez, un aumento en la demanda de programadores. Alrededor de 1965 se produjo la llamada crisis del software que consistía en que los programas no se entregaban a tiempo o tenían numerosos errores, esto evidenció la necesidad de especializar a los desarrolladores.

En Colombia, en 1968, se crea el primer programa de Ingeniería de Sistemas y Computación, ofrecido por la Universidad de los Andes [11-12], los estudiantes tomaban clases de cálculo, programación en lenguajes Fortran y Cobol, cursos de algoritmos, entre otros. Los primeros semestres eran tomados en Colombia y los restantes en el exterior.

3.1.2 Década de 1970

Un curso avanzado, que fue dictado en la primavera de 1972, en Munich, con una intensidad de 10 días y con la participación de profesores de varias nacionalidades, contenía la estructura de lo que hoy en día conocemos como ingeniería de software y contemplaba los siguientes tópicos [10]: uso de herramientas descriptivas; técnicas en el diseño de software y procesos de producción, uso de técnicas especiales; y aspectos prácticos.

Bauer [10], plantea el problema de la educación entre 1975 y 1985. Se esperaba que la ingeniería de software contara con un conjunto de técnicas y principios, que cambiaran fundamentalmente los hábitos de la comunidad de programadores.

Se identifica a la ingeniería de software como un campo emergente [13], que apunta hacia la mejora de la calidad de los procesos usados en la producción de sistemas de información y busca trasformar su creación, de un arte en una disciplina de ingeniería. Comprendía el rango de actividades usadas para diseñar y desarrollar software, que consistían en: análisis de requisitos; diseño de software; metodología sistemática de programación; programación de pruebas; programación para la verificación; certificación del software;herramientas y entornos de programación; rendimiento del software; documentación y administración del desarrollo. El impacto de los cambios implicaba entrenamiento o reentrenamiento de todos los individuos relacionados con el desarrollo de software.

Para 1978 el mantenimiento del software, ya fuera por ampliación o mantenimiento representaba el 75% de toda la actividad de programación, lo cual era mucho más costoso que el desarrollo original; esto era provocado por documentación inadecuada, programas ilegibles y errores de software. Las instituciones universitarias debían replantear la enseñanza de la ingeniería de software y fue en 1975 cuando creció realmente el interés en las ciencias de la computación, llevando al desarrollo de nuevos cursos y currículos que tenían un enfoque teórico, con orientación matemática, incluyendo temas específicos como análisis de algoritmos, evaluación del rendimiento y verificación de programas [13].

La Universidad Nacional de Colombia [14], en 1978, mediante el Acuerdo 21 del Consejo Superior Universitario aprueba la creación del Departamento de Ingeniería de Sistemas. Las bases del programa eran las matemáticas y la teoría general de sistemas, unidas a la formación básica del ingeniero. Las áreas del currículo académico relacionadas con ingeniería de software eran: introducción a los computadores, técnicas de programación, análisis y diseño de sistemas I y II [15].

3.1.3 Década de 1980

En el año 1980 el Dr. Niklaus Wirth creó el lenguaje Pascal [16], con el fin de soportar la enseñanza y aplicación de metodologías en la ingeniería de software moderna que producía objetos de código más eficientes, reemplazando el lenguaje ensamblador.

En 1984, se crea el Instituto de Ingeniería de Software (SEI) [17], con el fin de desarrollar modelos de evaluación y mejorar en el desarrollo de software. Desde su creación se ha dedicado a investigar y construir el currículo de los diferentes módulos que deben conformar la ingeniería de software,

ISSN 1794-211X • e-ISSN 2322-939X • Vol 12, No 1 (enero-junio 2015). pp. 70-79. Universidad Distrital Francisco Jose de Caldas-Facultad Tecnologica.

para la fecha habían liberado 20, los cuales podían ser incluidos en los programas de enseñanza y estaban conformados por contenidos como [18]: especificación de requisitos; introducción al diseño; procesos de revisión técnica, administración de la configuración; protección de información; seguridad, aseguramiento de la calidad; pruebas unitarias y análisis; métricas; desarrollo de interfaces de usuario; protección de la propiedad intelectual; contratos de licenciamiento de software; entre otros.
Para el año 1987 [19] se realizó un estudio seleccionando 240 programas de aproximadamente 820, que cumplían con los requisitos mínimos para obtener un título en ciencias de la computación en Estados Unidos y Canadá. Como resultado se determinó que existían tres tendencias en los cursos ofrecidos para las áreas de ingeniería de software: la primera se enfocaba en cubrir el ciclo de vida del software e involucraba proyectos desarrollados por grupos; la segunda hacía énfasis en las primeras etapas del ciclo de vida del software; y la tercera consistía en cursos basados en temas teóricos, como métricas de software, administración de proyectos, temas legales y éticos.
Benenson [20] referencia al psicólogo Jean Piaget, quien señaló "un proceso de aprendizaje de cualquier individuo, usualmente recapitula el desarrollo histórico de un campo del conocimiento"; sin embargo, en la comunidad de programadores esta teoría no se aplica de esta forma, aprendiendo de los errores y su capacidad para superarlos y hace alusión a la responsabilidad como docentes para llevar a los estudiantes más allá del método del ensayo y error, aplicando técnicas de ingeniería de software.

3.1.4. Década de 1990

Comenzando esta década, la ingeniería de software aún no se consideraba una disciplina de la ingeniería, pero tenía el potencial para llegar a serlo según afirmaba Shaw [21], docente de la Universidad de Carnegie Mellon [22].
El interés por mejorar los currículos impulsó a las entidades educativas a buscar herramientas que ayudaran a la *consecución* de este objetivo, como es el caso del Stevens Institute of Technology [23], que propuso un programa de enseñanza en ingeniería de software integrando el computador personal para cada estudiante como herramienta de trabajo en cada uno de los cursos. Con esto lograron un cambio de actitud tanto de la facultad como de los estudiantes, demostrando que un equipo de escritorio es más alentador para el aprendizaje que un terminal en el centro de cómputo.
Se incrementa el desarrollo y uso de herramientas CASE [24], pero si se utilizaban era recomendable iniciar el currículo con los conocimientos básicos y, solo una vez adquiridos, trabajar con estas.
Otras iniciativas destacadas en esta década son las realizadas por Pierce [25], que propone hacer uso de la ingeniería inversa; Oudshoorn y Maciunas [26], que sugieren la enseñanza de la disciplina *Ingeniería de Software y Proyectos*, en donde la principal experiencia la constituye la programación del proyecto simulando los ambientes empresariales; Chang [27] señala la necesidad de construir los currículos teniendo en cuenta la programación orientada a objetos; y la Universidad de Detroit Mercy [28], que recomienda el uso de estándares para cada una de las áreas involucradas en el pénsum tales como ISO/IEC 12207, IEEE 1219, ISO/IEC 13817, UML, PMI, entre otros.

3.1.5 Década de 2000

Comenzando el nuevo milenio Gates *et al.* [29] hacen uso de una frase en su introducción: "El diseño y la programación son actividades humanas: olvide eso y todo está perdido", para hacer referencia a la necesidad de desarrollar ciertas cualidades en los estudiantes, que pueden ser necesarias en esta profesión, más que en otras. Los principales retos de la ingeniería de software están relacionados con factores humanos, como el trabajo cooperativo entre grupos, la comunicación efectiva y la creación de ambientes que fomenten la mejora continua.
Para 2002, Bourque *et al.* [30] presentan el resultado de un trabajo realizado en un workshop en Montreal,

ISSN 1794-211X • e-ISSN 2322-939X • Vol 12, No 1 (enero-junio 2015). pp. 70-79. Universidad Distrital Francisco Jose de Caldas-Facultad Tecnologica.

con el fin de dar a conocer un mapeo preliminar sobre dos iniciativas distintas acerca del cuerpo de conocimiento de la ingeniería del software, el SWEBOK [1] y el SEEK [31].

SWEBOK, o guía del cuerpo de conocimiento de la ingeniería de software, creado por el IEEE Computer Society [32], fue diseñado para caracterizar la disciplina de la ingeniería de software y proporcionar una guía de actualidad describiendo el conocimiento generalmente aceptado. Está orientado hacia una variedad de audiencias, dirigida para servir a organizaciones públicas y privadas en la necesidad de una vista consistente de la ingeniería de software; para definir la educación y requerimientos de entrenamiento; clasificación de roles; definición de políticas de evaluación de desempeño y planes de carrera. Para este momento, la guía era una versión de prueba publicada en el 2001, desarrollada por consenso en un proceso que involucró ocho mil comentarios recogidos en tres ciclos de revisión, con cerca de 500 revisores de 40 países.

SEEK, o cuerpo del conocimiento de la enseñanza de ingeniería de software [31], forma parte del Computer Curriculum Software Engineering (CCSE), una iniciativa de la Asociación de Informática (ACM) [33] y el IEEE Computer Society, para definir las recomendaciones para un currículo de ingeniería de software. El documento resultado del estudio muestra en forma detallada el mapeo de cada uno de los tópicos contenidos en cada una de las áreas del conocimiento. A nivel general, se presenta en la tabla 2.

Resultado del estudio se pueden mencionar como conclusiones: no existen marcadas diferentes entre las dos propuestas; los dos cubren el ciclo de vida del desarrollo de software e incluyen conceptos teóricos y conocimientos prácticos; en términos de amplitud, SEEK tiene mayor alcance, mientras que en términos de profundidad, SWEBOK es más completo.

Durante esta década aumenta el interés hacia los métodos de desarrollo ágil [34-35] como XP, Scrum, Dinamic Systems Development Methodology y Cristal Methods, entre otros; sin embargo, continúan vigentes marcos de trabajo más formales como Rational Unified Process-RUP® [36], que tiene como objetivo la producción de software de alta calidad, incorporando dentro de sus características las técnicas de desarrollo orientada a objetos [37] y el uso del lenguaje UML [38].

Para 2006 [39], en un estudio realizado a 101 proyectos, se determinó que la arquitectura más solicitada correspondía a la Web, con plataforma de desarrollo J2EE, sistema operativo Windows,

Guía SWEBOK	SEEK
Fuera de alcance	Fundamentos
Fuera de alcance	Práctica profesional
KA1. Requisitos de software	Requisitos de software
KA2. Diseño de software	Diseño de software
KA3. Construcción de software	Construcción de software
KA4. Pruebas de software	Verificación y validación de software
KA5. Mantenimiento de software	Evolución del software
KA6. Gestión de la configuración del software	Gestión del software
KA7. Gestión de la ingeniería de software	
KA8. Proceso de la ingeniería de software	Proceso de software
KA9. Herramientas y métodos de ingeniería de software	
KA10. Calidad del software	Calidad del software
Fuera de alcance	Sistemas y aplicación de especialidades

Tabla 2. Mapeo de las áreas de conocimiento SWEBOK vs. SEEK. [30].

ISSN 1794-211X • e-ISSN 2322-939X • Vol 12, No 1 (enero-junio 2015). pp. 70-79. Universidad Distrital Francisco Jose de Caldas-Facultad Tecnologica.

predominio de los lenguajes Java y JavaScript y base de datos Sql Sever; siendo recomendable que formaran parte del currículo académico, además, se propone incluir temas relacionados con la seguridad de software en todas las etapas [40].

La producción literaria identificada para el año 2008 da cuenta del crecimiento que se está produciendo en el mundo en diversas áreas de la tecnología y se nota un marcado interés por la arquitectura basada en servicios [41] y la tecnología móvil [42], los cuales deben formar parte de la enseñanza de las áreas de la ingeniería de software. Hislop [43] plantea la necesidad de identificar qué debe cubrir la enseñanza de ingeniería de software para la generación *Ne*. Honing [44] sugiere el uso de Team Software Process (TSP) para lograr niveles industriales de productividad y calidad razonable en entornos académicos.

Finalizando la década, Shaw [45] afirma que no se requieren más avances técnicos, sino un mejor proceso, puesto que las habilidades para la producción de software no han llegado a su madurez para una práctica comercial.

3.1.6 2010 a 2015

La literatura identificada sugiere la necesidad de incorporar en el currículo las nuevas tendencias tecnológicas en la enseñanza. Tales como entornos de aprendizaje móvil que apoyen el desarrollo de los contenidos [46]. Para el año 2011 existían países que tenían mayor número de dispositivos móviles que personas, que propiciaba la facilidad de aprovechamiento de esta tecnología. También se ha popularizado el uso de tecnologías en la nube [47] y la computación orientada a servicios [48], además del surgimiento y popularización de nuevos lenguajes de programación como Ruby y Python [49].

4. DISCUSIÓN

La ingeniería de software es una disciplina relativamente joven si se compara con otras más longevas como la medicina [50]. Por lo anterior, esta área del conocimiento es un buen referente para realizar un análisis comparativo de las características requeridas para realizar cambios curriculares.

Bland y otros [51] identificaron 31 características requeridas para el éxito de un cambio curricular en las escuelas de medicina, clasificadas en 13 categorías y organizadas en tres grupos. Las categorías por grupo se relacionan a continuación: contexto (misión y meta, historia de cambio en la organización, políticas, estructura organizacional); currículo (necesidad de cambio, alcance y complejidad de la innovación); y proceso (clima cooperativo, participación de los miembros de la organización, comunicación, desarrollo del recurso humano, evaluación, inmersión en la ejecución, liderazgo).

Se observa que la medicina cuenta con una caracterización estructurada para realizar cambios curriculares, que en algunos casos pueden equipararse con aspectos de la ingeniería de software, pero que no son tan evidentes en la documentación identificada en esta última. Se observa que la dinámica de algunas características como la necesidad de cambio, que corresponde al grupo currículo, presenta diferencias entre estas disciplinas; mientras que en medicina, afirman los autores [51], "la organización no debería embarcarse en caminos innovadores sin tener primero la absoluta certeza de la necesidad de cambio". Esto la hace más reflexiva y por ende más pausada para decidirse a realizar reajustes al currículo, por el contrario la ingeniería de software tiene una dinámica más acelerada y responde con mayor inmediatez a los cambios marcados por los avances tecnológicos.

El currículo de ingeniería de software tiene un enfoque principalmente técnico dirigido hacia el desarrollo de las áreas del conocimiento propias de la disciplina. Considerar algunos elementos de la caracterización utilizada en otras disciplinas, como la medicina, contribuiría a enriquecer la construcción curricular de la ingeniería de software con una visión holística.

ISSN 1794-211X • e-ISSN 2322-939X • Vol 12, No 1 (enero-junio 2015). pp. 70-79. Universidad Distrital Francisco Jose de Caldas-Facultad Tecnologica.

5. CONCLUSIONES

La ingeniería de software surge como respuesta a los problemas que se presentaban con los sistemas de información en la década de 1960, conocidos como la crisis del software. Esto da lugar a la necesidad de especializar a los programadores y obtener un mejor proceso de producción de software. Así, las primeras propuestas curriculares versaban sobre temas aislados, orientados básicamente a la codificación en un determinado lenguaje.

Se observa que el interés por las ciencias de la computación aumenta en la segunda mitad de la década de 1970 y que en la década de 1980 nacen nuevas instituciones preocupadas por mejorar los procesos de construcción de software como es el caso del SEI [17].

En la década del 2000, la ACM [33] y la IEEE [32] publican el SEEK [31,52] documento que define las recomendaciones para un currículo de ingeniería de software y la IEEE publica la guía SWEBOK [1,53, 54], documento que debería ser tenido en cuenta como referencia en la construcción curricular de las áreas de ingeniería de software en las instituciones de enseñanza de esta disciplina. La guía SWEBOK ha tenido actualizaciones de acuerdo con los cambios tecnológicos y la última versión fue publicada en el 2014. También se observa la necesidad de incluir el factor humano que fomente los principios y valores éticos en los futuros egresados.

Con la masificación de internet, los sistemas de software cambian su forma de operar, ahora se centran en la interacción con el usuario provocando que tengan muchos caminos alternos, mayor probabilidad de error y riesgos para la seguridad; ello conduce a la necesidad de adaptar las metodologías de desarrollo para la web incorporando los componentes que garanticen un entorno de trabajo seguro. La popularización de los dispositivos móviles, la computación en la nube, el deseo creciente de obtener nuevos productos de software, llevaron al nacimiento de las metodologías de desarrollo ágil con el fin de responder con eficiencia a las demandas del mercado, sin que esto implique el desplazamiento total de las metodologías consideradas formales [4].

La frecuencia de los cambios tecnológicos relacionados con la ingeniería del software, así como la masificación del uso de la tecnología informática, ha provocado profundos cambios y grandes avances, que implicaban que se rediseñaran los currículos académicos para adaptarse a las nuevas tendencias. Esta dinámica se sigue y seguirá presentando en la medida en que se requiera masificar el uso de los diferentes sistemas de información.

Es necesario conocer la evolución del currículo de una disciplina, con el fin de mejorar su diseño, acorde con los cambios que se dan en las necesidades de la comunidad académica y productiva, para de esta forma llegar a ser más eficientes y contribuir a la mejora continua de la enseñanza de las áreas claves que la conformen.

6. REFERENCIAS

[1] A. Abran *et al., Guide to the software engineering body of knowledge–SWEBOK*. Los Alamitos, CA: IEEE Computer Society, p. 202, 2004.

[2] E.M. Lara, "*Fundamentos de investigación"*. México D.F.: Alfaomega, pp. 49-61, 2011.

[3] S. P. Chuquín, "Evaluación de estrategias metodológicas y propuesta de infraestructura para enseñanza de diseño y construcción de software", trabajo de titulación, Maestría en Tecnología Informática, Universidad Pedagógica y Tecnológica de Colombia, Tunja, Boyacá, 2015.

[4] I. Sommerville, *Ingeniería de Software*, 7a. ed. España: Pearson Education, p 687, 2006.

[5] R. Pressman, *Ingeniería de software*, 7 ed. México: McGrawHill, p .777, 2010.

[6] SCOPUS. Noviembre 2014. [En línea]. Disponible en: http://www.scopus.com/.

[7] R. W. Berger, "Software for simulating financial decisions", *ACM SIGSIM Simulation Digest*, Vol. 4, No 3, pp. 31-40, 1973.

[8] D. K. Banner, "What a financial manager should know about Cobol and Assembly Language", in *Management and computer systems*, 1970.

ISSN 1794-211X • e-ISSN 2322-939X • Vol 12, No 1 (enero-junio 2015). pp. 70-79. Universidad Distrital Francisco Jose de Caldas-Facultad Tecnologica.

[9] M. L. Shooman, "Structural models for software reliability prediction", I*EEE Computer Society ICSE'76 Proceding of the 2nd international conference on Software engineering*, pp. 268-280, 1976.

[10] F.L. Bauer, "Software and software engineering", *SIAM Review*, Vol. 15, No 2, pp. 469-480, Abril 1973.

[11] Universidad de los Andes, "Historia Facultad de Ingeniería", Noviembre 2014. [En línea]. Disponible en: https://ingenieria.uniandes.edu.co/historia.

[12] Universidad de los Andes, "Ingeniería de Sistemas y Computación: Una utopía realizada en la Universidad de los Andes", *Revista Ingeniería*, No 32, pp. 126-130, Julio-Diciembre 2010.

[13] P. Freeman and A. I. Wasserman, "Software engineering education: Status and prospects", *IEEE*, Vol. 66, No 8, pp. 886-892, Agosto 1978.

[14] H. Castellanos, "Historiografía de la Ingeniería de Sistemas en la U.N", *Ingeniería e Investigación*, No 47, pp. 114-129, 2010.

[15] Universidad Nacional de Colombia, *Acuerdo 91 de 1978. Acta 25 de agosto 4. Por el cual se crea la carrera de Ingeniería de sistemas y se establece su plan de estudios*, Noviembre 2014. [En línea]. Disponible en: http://www.legal.unal.edu.co/sisjurun/normas/Norma1.jsp?i=69299.

[16] R. J. Cichelli, "Pascal-Programming Language for the 80's", *Small Systems World*, Vol. 7, No 12, Junio 1980.

[17] Software Engineering Institute-SEI, Noviembre 2014. [En línea]. Disponible en: http://www.sei.cmu.edu

[18] N. E. Gibbs, "The SEI Education Program: The challenge of teaching future software engineers", *Communications of the ACM*, Vol. 32, No 5, pp. 594-605, Mayo 1989.

[19] L. M. Leventhal and B. T. Mynatt, "Components of typical undergraduate software engineering courses: Result for a Survey", *IEEE Transactions on software engineering*, Vol. SE-13, No 11, pp. 1193-1198, November 1987.

[20] G. Benenson, "Using software engineering to break the programming barrier", *IEEE Frontiers in Education conference Proceedings*, Session 25B1, Santa Barbara CA, pp. 394-398, October 1988.

[21] M. Shaw, "Prospects for an engineering discipline of software", *IEEE Software*, Vol. 7, No. 6, pp. 15-24, November 1990.

[22] Carnegie Mellon University, Noviembre 2014. [En línea]. Disponible en: http://www.cmu.edu

[23] E. C. Neu, "Computer integration: Teaching innovations", *IEEE Frontiers in Education Conference,* Session 23D4, West Lafayette IN, pp. 506-509, September 1991.

[24] A. Von Mayrhauser and K. Olender, "Teaching engineering disciplines to tool developers", *IEEE Fifth International Workshop*, Montreal, Que, pp. 283-284, Julio 1992.

[25] K. R. Pierce, "The benefits of maintenance exercises in project-based course in software engineering", IEEE, Orlando, FL, pp. 324-325, November 1992.

[26] M. J. Oudshoorn and K. J. Maciunas, "Experience with a project-based approach to teaching software engineering", IEEE, Dunedin, pp. 220-225, November 1994.

[27] E. Chang, "Teaching Object Oriented Software Engineering within a Multiple Paradigm Context", IEEE, Dunedin, pp. 238-244, November 1995.

[28] V. Jovanovic *et al*., "Use of software engineering standards in teaching", *Fourth IEEE International Symposium and Forum*, Curitiba, pp. 122-130, Mayo 1999.

[29] A.Q. Gates *et al*., "A structured approach for managing a practical software engineering course", *IEEE: 30th ASEE/IEEE Frontiers in Education Conference*, Session T1C, Kansas City, MO, pp. 21- 26, October 2000.

[30] P. Bourque *et al*., "Guide to the Software Engineering Body of Knowledge (SWEBOK) and the Software Engineering Education Knowledge (SEEK) – A Preliminary Mapping", IEEE Computer Society: Proceedings of the 10th International Workshop on Software Technology an Engineering Practice (STEP'02), Montreal, pp. 8-23, October de 2002.

ISSN 1794-211X • e-ISSN 2322-939X • Vol 12, No 1 (enero-junio 2015). pp. 70-79. Universidad Distrital Francisco Jose de Caldas-Facultad Tecnologica.

[31] IEEE and ACM, "Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering", 129 p, Agosto 2004.

[32] Instituto de Ingenieros Eléctricos y Electrónicos (IEEE), Noviembre 2014. [En línea]. Disponible en: http://www.ieee.org.

[33] Association for Computing Machinery (ACM), Noviembre 2014. [En línea]. Disponible en: http://www.acm.org.

[34] R. Reichlmayr, "The agile approach in an undergraduate software engineering course project", IEEE, 33rd ASEE/IEEE Frontiers in Education Conference, Session S2C, pp.13-18, November 2003.

[35] G. Hedin *et al*., "Introducing software engineering by means of Extreme Programming", *IEEE Computer Society: Proceedings of the 25th International Conference on Software Engineering* (ICSE'03), pp. 586-593, Mayo 2003.

[36] P. Kruchten, "Tutorial: Introduction to the Rational Unified Process", IEEE, Orlando, FL, pp. 703, mayo 2002.

[37] M. B. Blake and T. Cornet, "Teaching an object oriented software development lifecycle in undergraduate software engineering education", IEEE Computer Society Proceedings of the 15th Conference on Software Engineering Education and Training, Covington Ky, pp. 234-240, Febrero 2002.

[38] C. Starrett, "Teaching UML modeling before programming at the high school level", *IEEE Computer Society: Seventh IEEE International Conference on Advanced Learning Technologies* (ICALT 2007), Niigata, pp 713-714, Julio 2007.

[39] R. J. Fornaro *et al*., "What clients want–what students do: Reflections on ten years of sponsored senior design projects", *IEEE Proceedings of the 19th Conference on Software Engineering Education & Training* (CSEET'06), Turtle Bay, HI, pp. 226-236, Abril 2006.

[40] W. A. Conklin and G. Dietrich, "Secure software engineering: A new paradigm", *IEEE Computer Society: 40th Hawaii International Conference on System Sciences*, Waikoloa, HI, pp. 272, Enero 2007.

[41] S. Tilley *et al*., "Teaching and using service-oriented arquitecture (SOA) in an academic environment", *IEEE International Systems Conference*, Montreal, pp. 1-4, Abril 2008.

[42] J. Chenard *et al*., "A laboratory setup and teaching methodology for wireless and mobile embedded systems", *IEEE Transactions on Education*, Vol. 51, n.° 3, pp. 378-384, Agosto 2008.

[43] G. W. Hislop, "Teaching programming to the net generation of software engineers", *IEEE Computer Society 21st Conference on Software Engineering Education and Training Workshop* (CSEETW'08), Charleston, pp. 5-8, Abril 2008.

[44] W. L. Honig, "Teaching successful 'Real-World' software engineering to the 'Net' generation: Process and Quality Win!", *IEEE Computer Society: 21st Conference on Software Engineering Education and Training*, Charleston, pp. 25-32, Abril 2008.

[45] M. Shaw, "Continuing Prospects for an Engineering Discipline of Software", IEEE Software, Vol. 26, No 6, pp. 64-67, Noviembre-Diciembre de 2009.

[46] R. Maceiras *et al*., "Adaptation of a virtual campus for mobile learning devices", IEEE Global Engineering Education Conference, Amman, pp. 165-167, Abril 2011.

[47] G. Yang and Z. Zhu, "The Application of SaaS-Based cloud computing in the university research and teaching platform", *IEEE International Conference on Intelligence Science and Information Engineering*, Wuhan, pp. 201-213, Agosto 2011.

[48] Y. Chen, "Service orientation in computing curriculum", *IEEE 6th International Symposium on Service Oriented System Engineering* (SOSE 2011), Irvine, CA, pp. 122-133, Diciembre 2011.

[49] E. R. Harley and Z. Harley, "A wizard for E-Learning computer programming", IEEE, Lodz, pp. 95-98, Septiembre 2012.

[50] J.A. López. "Hipócrates y los escritos hipocráticos: Origen de la medicina científica". *Revista de Filología de la UNED*, No 2, pp. 159-175, 1986.

ISSN 1794-211X • e-ISSN 2322-939X • Vol 12, No 1 (enero-junio 2015). pp. 70-79. Universidad Distrital Francisco Jose de Caldas-Facultad Tecnologica.

[51] C.J. Bland *et al.*, "Curricular change in medical schools: How to succeed". *Academic Medicine*, Vol. 75, No 6, pp. 575-594, Junio de 2000.

[52] IEEE and ACM, "Software Engineering 2014: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering", SE2004 Revision Process, 2014.

[53] P. Bourque and R. Fairley, "Guide to the Software Engineering Body of Knowledge Version 3.0. SWEBOK", IEEE Computer Society, 2014.

[54] D. Rodríguez *et al.*, "Ingeniería de software un enfoque desde la guía SWEBOK". México: Alfaomega, 2012.

ISSN 1794-211X • e-ISSN 2322-939X • Vol 12, No 1 (enero-junio 2015). pp. 70-79. Universidad Distrital Francisco Jose de Caldas-Facultad Tecnologica.