EL APRENDIZAJE ACTIVO COMO FUNDAMENTO PARA LA ENSEÑABILIDAD DE CURSOS DE DISEÑO DE SOFTWARE, UNA REFLEXIÓN DESDE LA DIDÁCTICA DEL DISEÑO DE SOFTWARE DDS

Robinson Andrés Jiménez Toledo[1]
Álvaro Alexander Martínez Navarro[2]

Artículo resultado de la investigación "Didáctica del Software", realizado para optar por el título de Magister en Docencia Universitaria de la Universidad de Nariño y, de la experiencia y sistematización de resultados en el Programa de Ingeniería de Sistemas de la Universidad Mariana. Línea de investigación Pedagogía y Didácticas.

Fecha de recepción: 1° – Nov.- 013 **Aceptado: 7**– Dic. - 013

RESUMEN

El objetivo del presente artículo es aportar en el proceso de enseñabilidad para cursos de Diseño de Software basado en experiencias exitosas de profesores en contextos internacionales. El estudio se enmarca principalmente desde el *paradigma cualitativo*, puesto que propende por presentar una alternativa práctica de solución, como fruto de la descripción, interpretación, auto reflexión y auto entendimiento. Basado en un *enfoque crítico-social*, encaminado al logro de una conciencia autorreflexiva y crítica para transformar la realidad, bajo un contexto cultural en donde el diálogo, el debate y la praxis (relación teoría-práctica), sean los ejes del quehacer docente investigativo. Y, fundamentado en el tipo de investigación colaborativa, caracterizado porque un grupo de profesionales define, armoniza y realiza un estudio de alguna problemática de interés común y que contribuye a la construcción de conocimiento didáctico y pedagógico, que facilite la integración de estos nuevos elementos a las labores educativas. De donde se llegó a la conclusión que, la formación pedagógica en docentes de cursos de Diseño de Software es fundamental para tratar de asegurar un buen proceso de enseñanza aprendizaje y evaluación, puesto que, conocer el constructo teórico de estos principios, posibilita al educador su labor de diseñar, construir y vivir escenarios activos que permitan al estudiante alcanzar las competencias necesarios.

PALABRAS CLAVE: didáctica específica, diseño de software, aprendizaje activo.

---

[1] Colombiano. Ingeniero de Sistemas – Universidad Cooperativa de Colombia. Magister en Docencia Universitaria - Universidad de Nariño. Docente Tiempo Completo de la Universidad Mariana. Director del Grupo de Investigación GISMAR de la Universidad Mariana. E-mail: razo_net@hotmail.com.

[2] Colombiano. Ingeniero de Sistemas – Universidad Mariana. Especialista en Docencia Universitaria – Universidad de Nariño. Magister en Docencia Universitaria - Universidad de Nariño. Docente Tiempo Completo de la Universidad Mariana. E-mail: aamartinezn@hotmail.com.

AS THE BASIS FOR ACTIVE LEARNING COURSE TEACHABILITY SOFTWARE DESIGN, A REFLECTION FROM THE EDUCATIONAL SOFTWARE DESIGN DDS

ABSTRACT

The aim of this paper is to contribute to the process of teachability for Software Design courses based on successful experiences of teachers in international contexts. The study was conducted primarily from the qualitative paradigm, since it tends to present an alternative settlement practice, as a result of the description, interpretation, self-reflection and self-understanding. Based on a critical-social approach, aimed at achieving a self-reflexive critical awareness and to transform reality in a cultural context where dialogue, debate and praxis (theory-practice relationship) are the axes of investigative teaching work. And, based on the type of collaborative research, wherein a group of professionals defined, harmonizes and makes a study of some problems of common interest and contributes to the construction of educational and pedagogical knowledge, to facilitate the integration of these new elements educational work. From this it was concluded that teacher training in educational courses Design Software is essential to try to ensure a smooth process of teaching learning and assessment, since, knowing the theoretical construct of these principles, it enables the teacher his work to design, build and live active scenarios that allow the student to achieve the necessary skills.

KEY WORDS: specific teaching, software design, active learning.

Año 2 Vol. 2 Nº 2 – 2013 (Págs. 104-121)
ISSN - 2322-9292

INTRODUCCIÓN

A nivel internacional se han logrado avances significativos en áreas relacionadas con el Diseño de Software, pero, existen pocos referentes, aportes y reflexiones en relación con la forma como se enseñan estos temas.

Este estudio se orienta en primera instancia, desde el aprendizaje activo basado en casos, donde el estudiante es el promotor de su propio aprendizaje, en segunda instancia, desde la temática relacionada con la didáctica, que a lo largo de la historia en la educación, se trata de definir, tanto a partir de su perspectiva etimológica, desde sus inicios con autores como Wolfgang Ratke (1571-1635) y Jan Amos Komenský (1592-1670), como desde autores contemporáneos como Mallart; de donde a partir de una exhaustiva recopilación, análisis y síntesis de los conceptos de diversos autores, apunta a conceptualizar la didáctica como la ciencia de la educación que estudia e interviene en el proceso de enseñanza, aprendizaje y evaluación, con el fin de conseguir la formación intelectual del educando (Mallart, 2000).Y en tercerea instancia, se aborda desde la temática de la enseñanza del diseño de software para las disciplinas internacionales formalizadas en la Associationfor Computing Machinery (ACM, 2009), reconocida como la más grande e importante sociedad informática, científica y académica del mundo en su área, junto con la IEEE – ComputerSociety (IEEE, 2009) otra asociación de gran relevancia en el mundo profesional de las ciencias de la computación, dedicada al avance de la innovación tecnológica y excelencia en beneficio de la humanidad.

Teniendo en cuenta la trascendencia del software en la vida del ser humano descrita por IanSommerville (Sommerville, 2008), quien afirma que casi todos los países dependen de complejos sistemas informáticos y la mayor parte de estos sistemas incluyen una computadora y software de control, la construcción de software de alta calidad se convierte en una actividad indispensable para ámbitos, tales como el educativo, económico, científico y social, entre otros. Con la evolución tecnológica y sus implicaciones en la sociedad, el diseño de sistemas computacionales se vuelve cada vez más complejo y en respuesta a ello la enseñanza del diseño de software debe sufrir cambios importantes para afrontar estos retos.

El proceso educativo de enseñar esta área de la ingeniería de software, está presente desde hace ya cuatro décadas aproximadamente, con una evolución sorprendente en el campo del conocimiento específico per sé. Sin embargo, las reflexiones sobre la práctica docente en dicho campo son muy escazas, aspecto que produce un desconocimiento general sobre la labor docente en cuanto a diseño de software. En este sentido, la enseñanza del diseño de software no

Año 2 Vol. 2 Nº 2 – 2013 (Págs. 104-121)
ISSN - 2322-9292

cuenta con la misma producción de conocimiento, como sí es manifiesta en la evolución de la didáctica de las matemáticas según (Gascón, 2007).

Los aportes al campo de estudio, se enmarcan dentro de una perspectiva internacional, construidos a través de mecanismos de participación de docentes con reconocida experiencia en el campo del diseño y el desarrollo de software; para ello, se identificaron las 80 mejores universidades del mundo en el campo de ingeniería, tecnología y ciencias de la computación según el ranking mundial dado por el Instituto de Educación Superior de la Universidad ShanghaiJiaoTong en China (Academic Ranking of WorldUniversities, 2009); a nivel nacional, se identificaron las 25 mejores universidades según el ranking dado por el observatorio de la Universidad Colombiana, con los mejores resultados en los exámenes de calidad de la educación superior (Saber Pro), para el programa de Ingeniería de Sistemas. Tanto a nivel internacional como nacional, fueron aplicadas encuestas online, cuyos resultados permitieron identificar bases teórico-prácticas de la labor docente.

Las más importantes asociaciones de computación e informática en el mundo han manifestado su preocupación por los currículos de las disciplinas relacionadas a estos campos. Según el informe denominado Computing Curricula 2005: (ACM, IEEE Computer Society and AIS, 2005) creado participativamente por tres instituciones a nivel mundial (ACM – The Associationfor Computing Machinery, AIS – The AssociationforInformationSystems y la IEEE-CS – The ComputerSociety), sugiere la presencia de cinco disciplinas específicas en el campo, a saber: a.) ingeniería en computación, b.) ciencia en computación, c.) sistemas de información, d.) tecnología de información, e.) ingeniería de software. Este estándar guía a las carreras profesionales y tecnológicas en el mundo que estén relacionadas con el campo de la computación y la informática. Lo interesante de este asunto es que, según este informe, el componente de construcción de software está presente en todas las disciplinas tomando parte importante en ellas.

La caracterización de las experiencias de enseñanza de diseño de software brinda una gran utilidad puesto que permite evidenciar la situación actual de cómo es la labor docente en el contexto internacional.La estrategia formulada tanto para abordar el problema como para darle solución a través de la didáctica, se convierte en una metodología atractiva y valiosa para la comunidad académica-científica que investiga en este campo, puesto que a nivel internacional no existen suficientes estudios que orienten la integración de aspectos multidisciplinarios de didáctica y diseño de software.Adoptar la didáctica como el conjunto de estrategias activas para potenciar al estudiante su capacidad de

Año 2 Vol. 2 Nº 2 – 2013 (Págs. 104-121)
ISSN - 2322-9292

diseño de software, les permitirá a los docentes contar con otro referente teórico para su quehacer; sobre todo, teniendo en cuenta la importancia del diseño de software en las carreras con enfoque disciplinar ACM. En consecuencia, la práctica docente orientada a través de la didáctica, pretende mejorar las habilidades de diseño del estudiante con el fin de garantizar aportes en avances tecnológicos y computacionales que respondan a las exigencias del nuevo milenio.

De acuerdo con lo anterior se obtuvo los siguientes resultados:

Tabla 1. Detalle de resultados para población y muestra.

Categorías	POBLACIÓN		MUESTRA
	No. universidades convocadas	N° de profesores titulares convocados	N° profesores titulares participantes
Internacional	80	80	15
Nacional	9	9	5

REFERENTES PARA EL CURSO DE DISEÑO DE SOFTWARE

Para construir la matriz de referentes del diseño de software, fue necesario realizar una revisión documental a los documentos *Computing Curricula 2005* (ACM, IEEE-CS & AS, 2005) y *SWEBOK 2004* (IEEE-CS, 2004) con el fin de obtener los perfiles, los objetivos de aprendizaje y los contenidos específicos sobre diseño de software.

Además, se realizó un vaciado de información acerca de los contenidos específicos que son cubiertos por los 15 docentes extranjeros y los 5 docentes nacionales. Con estos insumos se construyó la matriz de referentes de diseño de software tal como lo muestra la Figura 1.

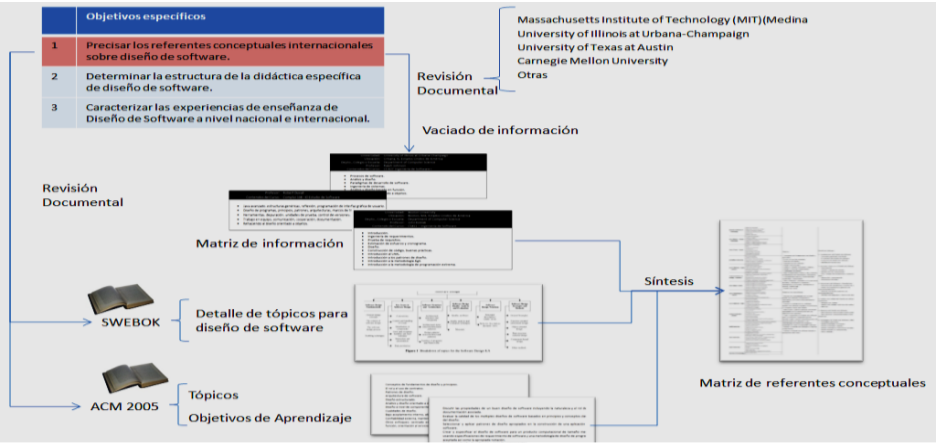

Figura 1. Metodología para precisar los referentes conceptuales internacionales sobre diseño de software.

Año 2 Vol. 2 Nº 2 – 2013 (Págs. 104-121)
ISSN - 2322-9292

CONTENIDOS DEL CURSO DE DISEÑO DE SOFTWARE

Dentro de la construcción de la didáctica (cómo enseñar) es importante estructurar los contenidos mínimos para el diseño de software (qué enseñar). Teniendo en cuenta el carácter internacional de la propuesta, este estudio sigue las recomendaciones hechas por ACM a través de su documento Computing Curricula. Sin embargo, el contenido del curso al cual se pretende armar una didáctica es muy específico teniendo en cuenta el amplio rango de acción de las ciencias de la computación. Es por eso que, para asumir los retos del diseño de software de estos tiempos, se hace necesario indagar sobre los contenidos apropiados para impartir un curso como tal. Como insumo principal del presente estudio se encuentra el análisis de los syllabus de cada curso ofrecido por los expertos encuestados a nivel internacional y nacional.

Los contenidos de los syllabus coinciden en ciertos contenidos y se diferencian en otros. Haciendo un estudio sobre aquello que se debería enseñar en un curso de diseño de software, se ha tomado como segunda referencia el texto titulado "SWEBOK: Guide to Software EngineeringBody of Knowledge (2004)". Esta guía ha sido producida por una gran cantidad de expertos en el mundo en el tema de la educación en ingeniería de software y, es respaldada por IEEE ComputerSociety.

Entonces, se tienen tres fuentes que orientan la estructura de contenidos para el curso al cual se le diseñará la didáctica, como se indica en la figura 2.

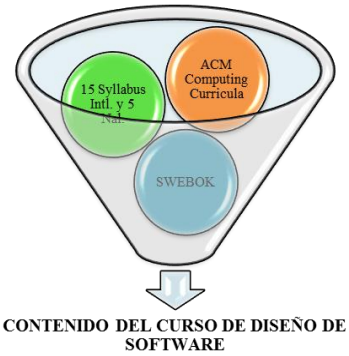

Figura 2. Estructura orientadora de contenidos para la Didáctica del Diseño de Software.

IDENTIFICACIÓN DE LOS COMPONENTES DE LA DIDÁCTICA DE DISEÑO DE SOFTWARE

Para precisar cada uno de los elementos que conforman la Didáctica del Diseño de Software, se procedió como se muestra en la figura 3, indicándose los insumos, procesos y resultados frutos de estudio del segundo objetivo específico.

Año 2 Vol. 2 Nº 2 – 2013 (Págs. 104-121)
ISSN - 2322-9292

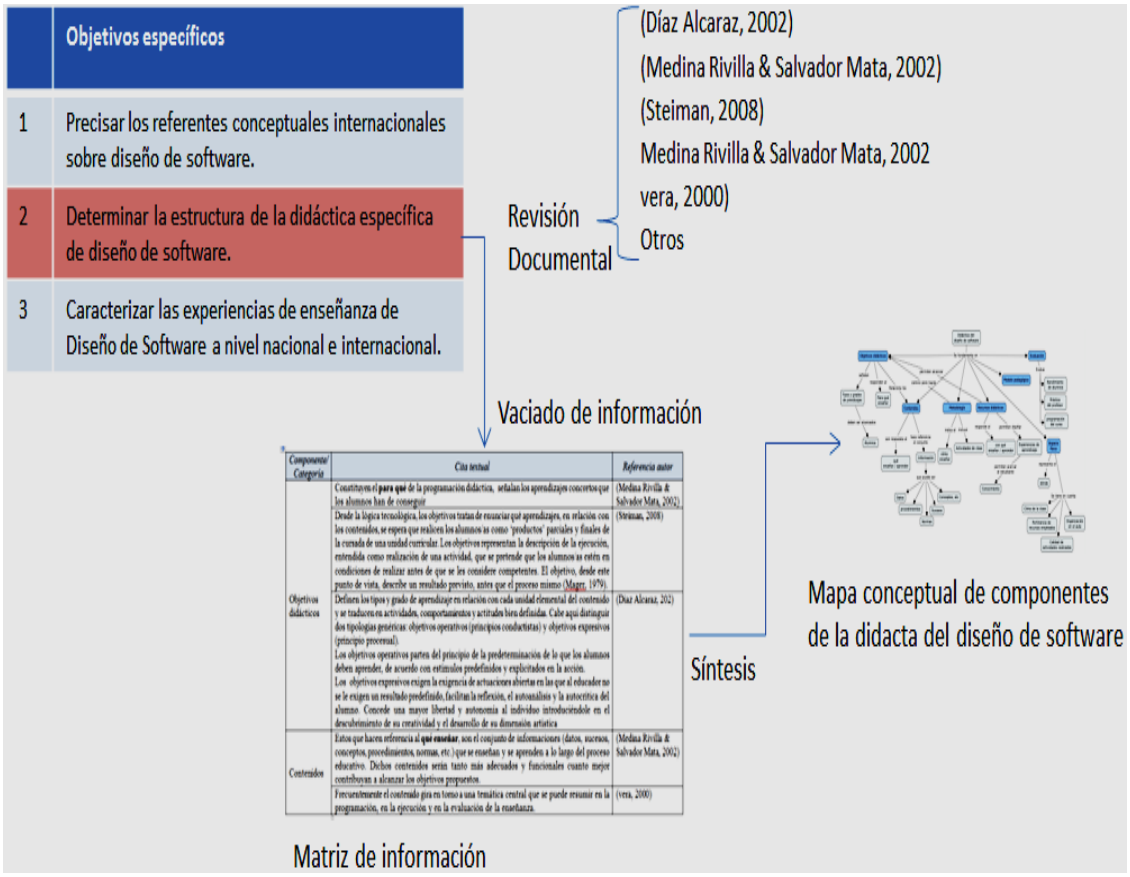

Figura 3. La metodología seguida para identificar los elementos constitutivos de la Didáctica de Diseño de Software.

En síntesis, se construyó un mapa conceptual como se indica en la figura 4, como resultado del análisis de la revisión documental, identificándose los elementos comunes reflexionados por diferentes autores y contextos, proceso que permitió crear categorías para el diseño de los instrumentos usados en la recolección de información de los expertos en la enseñanza de cursos de diseño de software y enriquecer la estructura de la Didáctica del Diseño de Software.

Año 2 Vol. 2 Nº 2 – 2013 (Págs. 104-121)
ISSN - 2322-9292

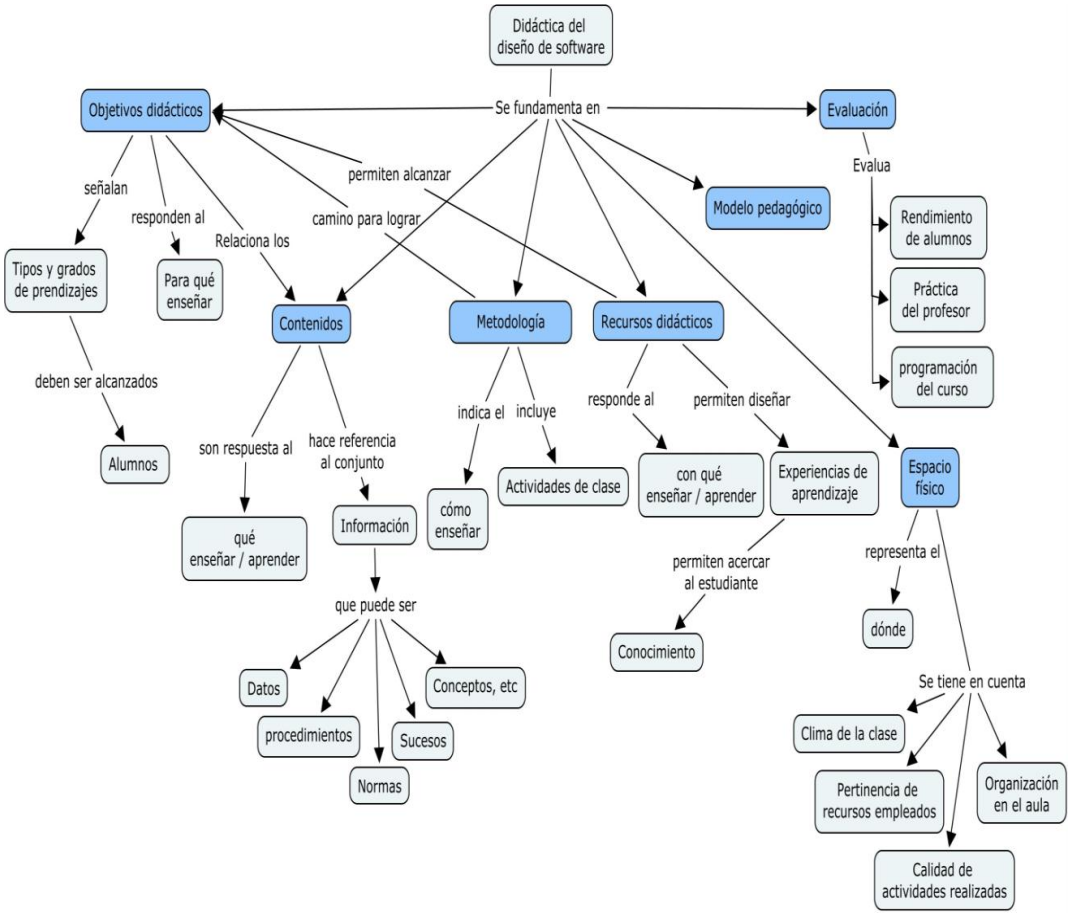

Figura 4. Mapa conceptual síntesis de los componentes de la Didáctica del Diseño de Software.

CARACTERIZACIÓN DE LAS EXPERIENCIAS DE ENSEÑANZA DE DISEÑO DE SOFTWARE

Para la caracterización de las experiencias de la práctica docente de los docentes colaboradores de la investigación se aplicó una encuesta de opinión a través de correo electrónico (ver Anexo E). El proceso de análisis, interpretación y resultados, llevado a cabo se puede apreciar en la figura 5.

Año 2 Vol. 2 Nº 2 – 2013 (Págs. 104-121)
ISSN - 2322-9292

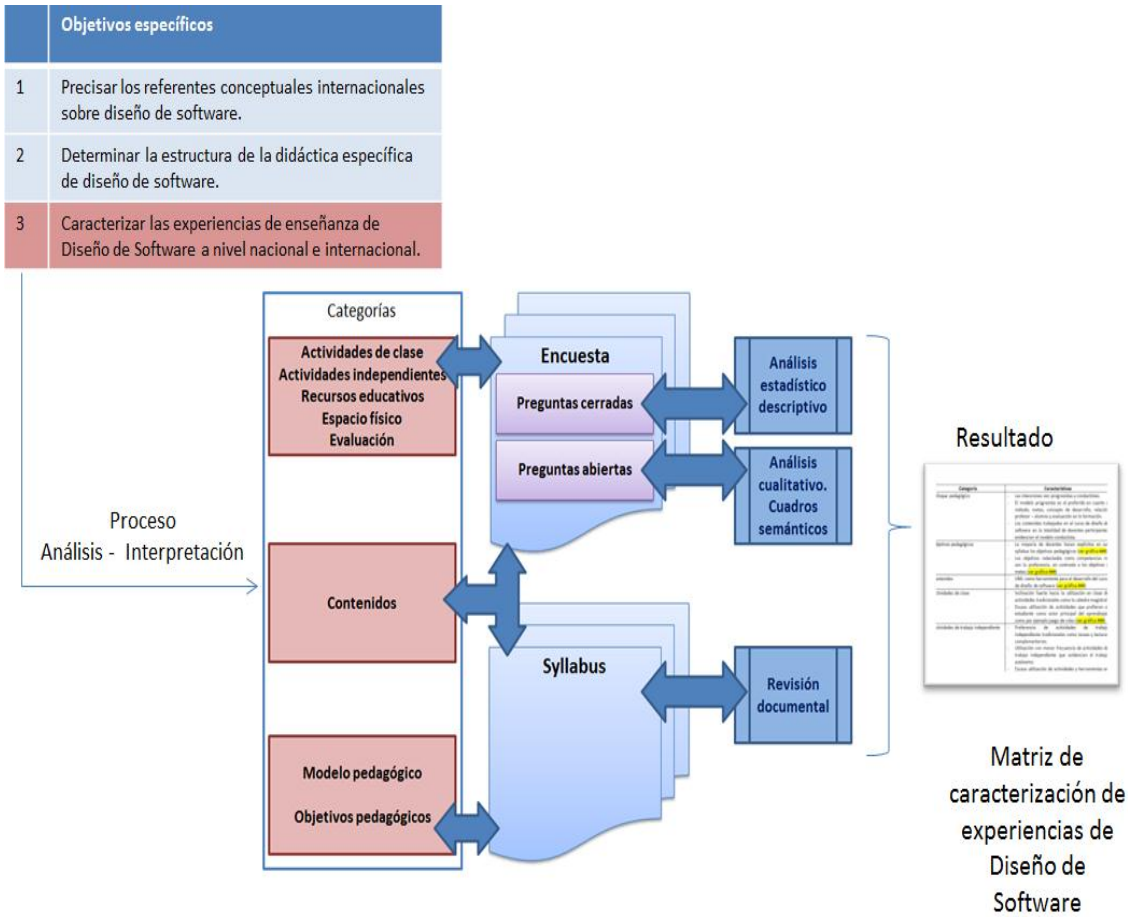

Figura 5. Metodología para caracterizar la enseñanza del diseño de software.

ANÁLISIS ESTADÍSTICO DESCRIPTIVO.

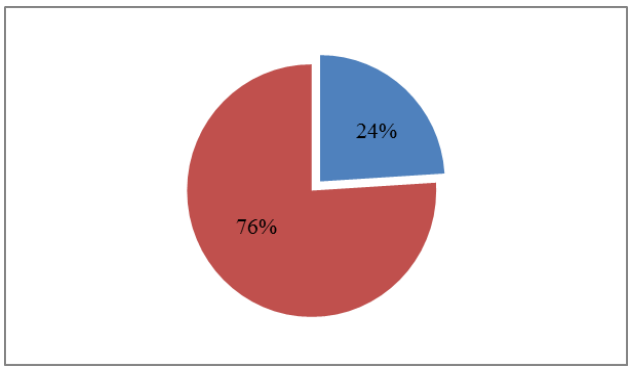

Figura 6**.** Uso de UML en la enseñanza del curso de diseño de software.

Año 2 Vol. 2 Nº 2 – 2013 (Págs. 104-121)
ISSN - 2322-9292

En la figura 6 se muestra como la gran mayoría (76%) de los encuestados utiliza en la enseñanza del curso de diseño de software UML, hecho que corrobora que a nivel mundial el estándar de modelado es masivamente trabajado, en la didáctica se tendrá muy en cuenta este aspecto.

Quienes no hacen uso de la norma para modelar (24%) manifestaron, al momento de contestar la encuesta, no estar trabajando directamente como docentes sino que hacen parte del profesorado, bien sea en áreas diferentes al diseño de software o en la parte administrativa.

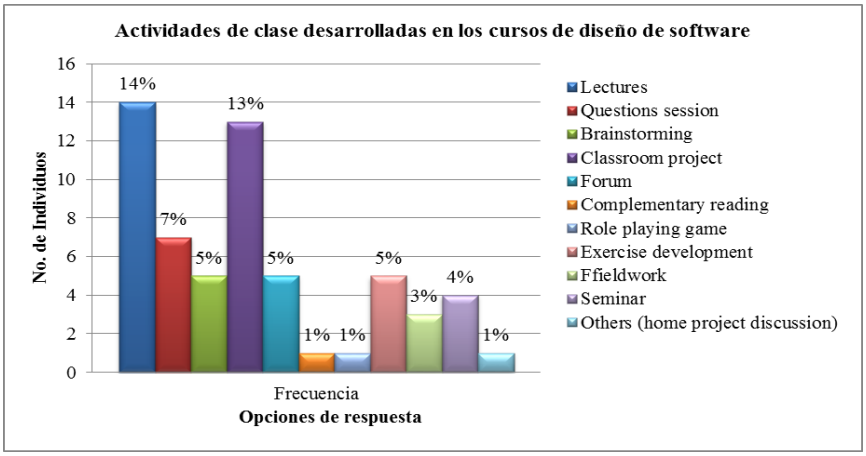

Figura 7. Actividades de clase desarrolladas en los cursos de diseño de software.

La figura 7 evidencia la cátedra magistral y los proyectos de clase como las dos principales actividades desarrolladas en el aula en el curso de diseño de software, obteniendo el mayor puntaje la primera de ellas, es claro entonces, la inclinación por formas tradicionales de enseñanza.

Las sesiones de preguntas y la lluvia de ideas ocupan un tercer lugar con una mínima diferencia entre ellas. Y, se refleja que actividades como lecturas complementarias, juego de roles, discusiones, trabajo de campo, seminarios y desarrollo de ejercicios, que se caracterizan por preferir al estudiante como principal protagonista de su aprendizaje, ocupan un lugar secundario.

Año 2 Vol. 2 Nº 2 – 2013 (Págs. 104-121)
ISSN - 2322-9292

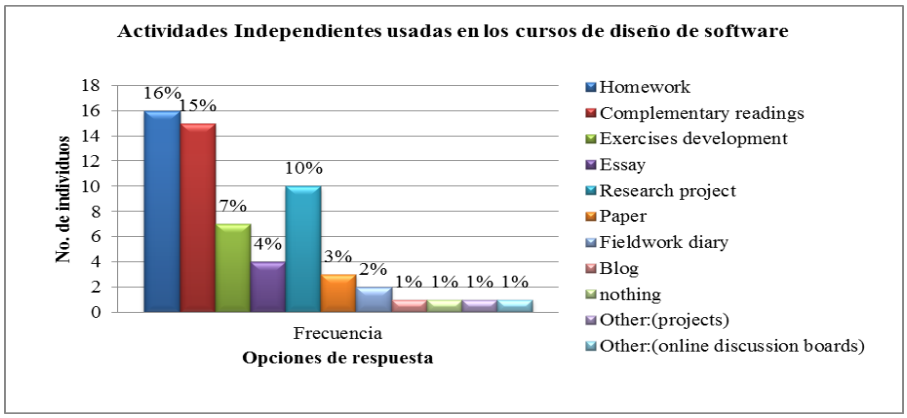

Figura 8. Actividades independientes usadas en los cursos de diseño de software.

Las tareas, lecturas complementarias y proyectos de investigación son las principales actividades independientes utilizadas en el curso de diseño de software por los docentes encuestados. Un segundo, lugar lo ocupan en su orden: el desarrollo de ejercicios, ensayos y artículos. Con un mínimo puntaje se encuentran las actividades en línea como panel de discusión y blog, proyectos y diario de campo. Un solo docente encuestado no utiliza actividades independientes en el desarrollo de su curso. Se evidencia según la figura 8 que, a nivel nacional e internacional aún se prefiere enfoques tradicionales cuando de actividades independientes se trata y trabajos como el artículo y el ensayo, que obligan al estudiante a ser autónomo en su quehacer muy poco son utilizadas.

Como se puede apreciar en la anterior descripción, el perfil del trabajo independiente preparado por los docentes encuestados para sus estudiantes, aún tiene características tradicionales y aprovecha en un mínimo porcentaje las herramientas en línea disponibles en la red.

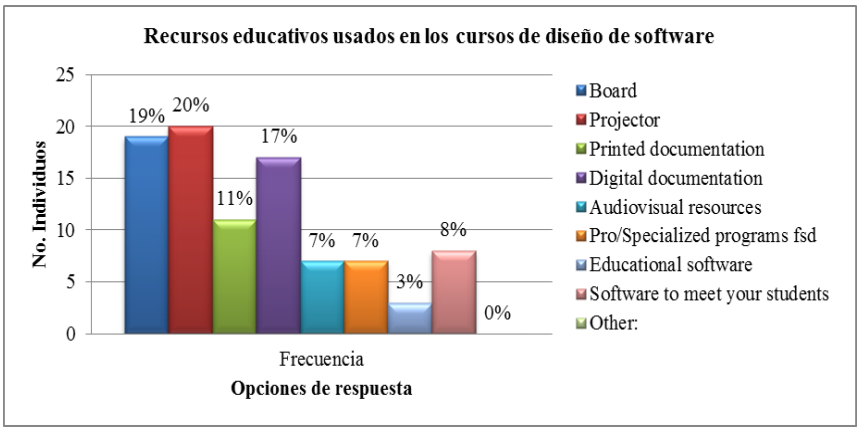

Figura 9. Recursos educativos usados en los cursos de diseño de software.

Año 2 Vol. 2 Nº 2 – 2013 (Págs. 104-121)
ISSN - 2322-9292

En cuanto a los recursos educativos usados en el desarrollo del curso de diseño de software, la figura 9 muestra como el proyector y el tablero son los preferidos por un 97.5% de los docentes, esto refleja una inclinación hacia las formas tradicionales de trabajo en clase. La documentación tanto digital como impresa es utilizada también como recurso educativo en un alto porcentaje (70%), los recursos software especializados y educativos son utilizados con menor frecuencia (50%). Con un porcentaje menor pero importante aparece la utilización de software para trabajar en equipo y para mantener comunicación con los estudiantes (40%). La inclinación de la utilización de los recursos en clase, se caracteriza de manera clara, si se trae como referencia al credo de aprendizaje activo de Silberman, porque al final del proceso de enseñanza – aprendizaje la mayoría los estudiantes olvidarán en un alto grado ó comprenderán muy poco lo visto en cada sesión.

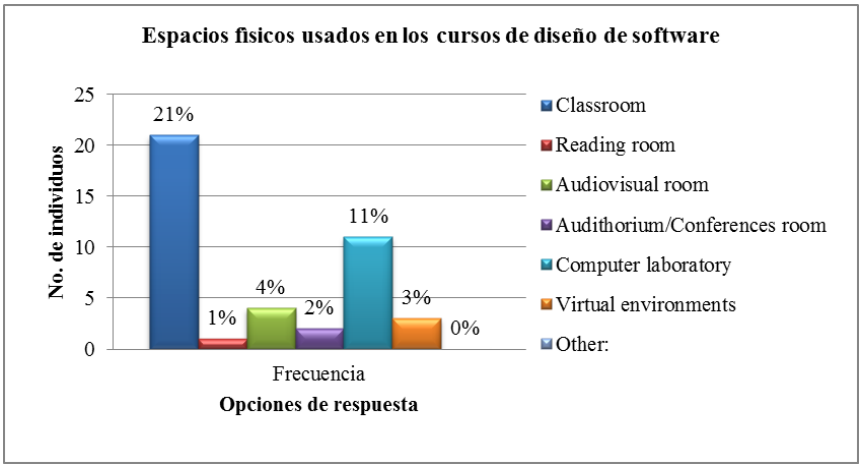

Figura 10. Espacios físicos usados en los cursos de diseño de software.

Como se ilustra en la figura 10, el aula de clase en un 100% es el lugar de trabajo preferido de los docentes encuestados, en un segundo lugar aparece el laboratorio de computadores y compartiendo un tercer puesto con un porcentaje de 12.5% están la sala de audiovisuales, entornos virtuales, salón de conferencias y auditorios y, la sala de lectura. Ninguno de los encuestados utiliza para el desarrollo del curso de diseño de software espacios físicos diferentes a los planteados en la encuesta. Es clara la tendencia de utilizar el salón de clase como principal espacio físico en el desarrollo del curso de diseño de software, por tanto, es muy pertinente hacer de este lugar un espacio activo de aprendizaje, la Didáctica del Diseño de Software aporta elementos concretos en este sentido.

Año 2 Vol. 2 Nº 2 – 2013 (Págs. 104-121)
ISSN - 2322-9292

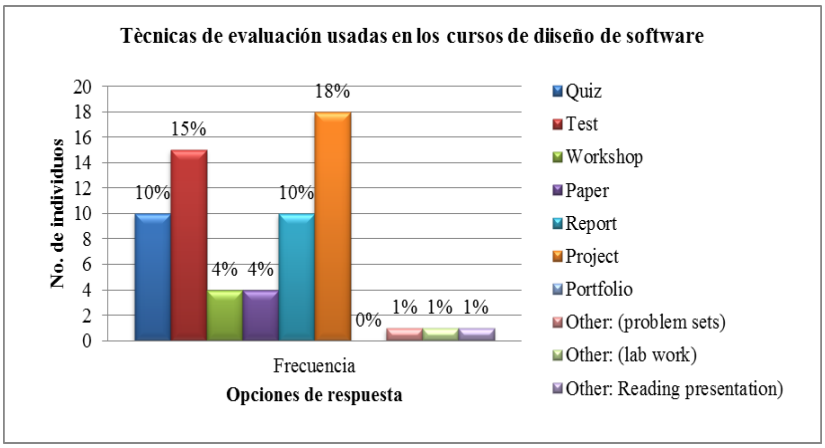

Figura 11. Técnica de evaluación usada en los cursos de diseño de software.

El proyecto, con un 90% es la técnica de evaluación más utilizada por los docenes encuestados, le siguen en su orden el examen con un 75%, la prueba corta y reporte con un 50%, el artículo y taller ocupan el cuarto puesto con un 20% y en último lugar están los problemas, trabajo de laboratorio y exposiciones con un 5%, información visualizada en la figura 11. Ninguno de los encuestados utiliza el portafolio como técnica de evaluación. Estos resultados indican que, si bien el proyecto es utilizado en su gran mayoría y por sus características permite al estudiante ser autónomo y auto regularse; aún se sigue con la evaluación tradicional a través de exámenes, pruebas cortas y reportes cuyas características enmarcan al estudiante en posibilidades limitadas de respuesta.

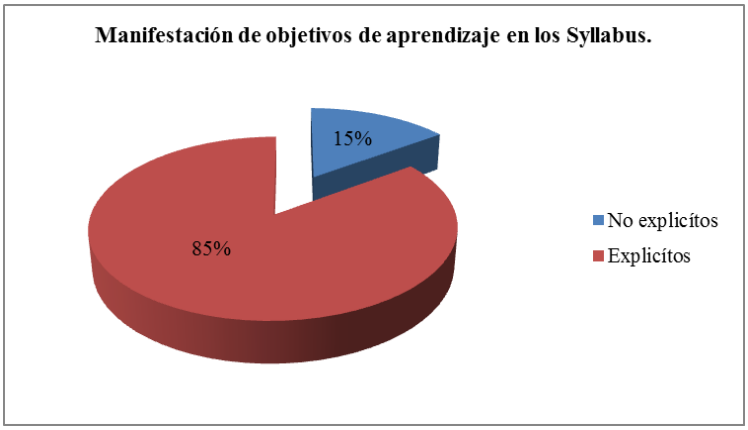

Figura 12. Manifestación de objetivos de aprendizaje en los Syllabus.

En cuanto a la manifestación de los objetivos de aprendizaje en los Syllabus de los cursos de diseño de software, en la figura 12, se alcanza a apreciar que la

Año 2 Vol. 2 Nº 2 – 2013 (Págs. 104-121)
ISSN - 2322-9292

gran mayoría (85%) los hacen explícitos, el resto (15%) no lo hacen, esto evidencia el hecho de orientar y otorgar intención a la enseñanza. Es clara la tendencia de orientar la labor de enseñanza por medio de propósitos.

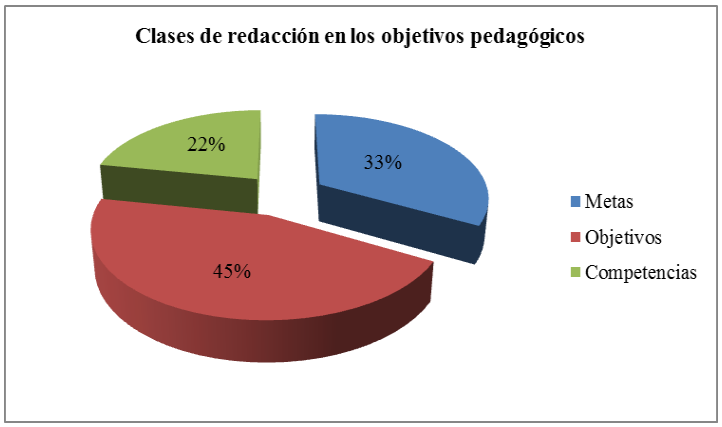

Figura 13. Clase de redacción en los objetivos pedagógicos.

Para clasificar el tipo de redacción que se evidencia en los objetivos pedagógicos de los Syllabus se tomaron las definiciones de meta, objetivo y competencia; la primera hace referencia a un fin que se persigue y se alcanza en corto plazo a través de herramientas técnicas disciplinares; la segunda persigue formar, además de utilizar herramientas técnicas; y las competencias persiguen que el estudiante logre con el saber y el ser, hacer en un contexto dado.

En la figura 13 se aprecia, en un 45% de ellos, la inclinación por la redacción en forma de objetivos, un 33% lo hacen a manera de metas y apenas el 22% redactan los propósitos como competencias. Estos resultados revelan que el tránsito hacia el enfoque por competencias aún está en curso y que la intención de formación utilitarista basada en herramientas técnicas cada vez más, se transforma en objetivos y de esta forma se dan pasos importantes hacia competencias.

ANÁLISIS CUALITATIVO DE LA INFORMACIÓN.

En la encuesta aplicada a los docentes de las universidades extranjeras, se preguntó de manera abierta por los aspectos positivos y negativos de su práctica docente en los cursos de diseño de software, la forma de analizar sus respuestas dadas se puede apreciar en la figura 14, que comienza con la fase de vaciado de información donde literalmente se escriben los datos proporcionados, luego se identifican las palabras recurrentes y se agrupan las

Año 2 Vol. 2 Nº 2 – 2013 (Págs. 104-121)
ISSN - 2322-9292

respuestas por recurrencias; a partir de ello se construyen predicados y descriptores, que se constituyen en insumos para determinar las mezclas que fueron integradas en las redes semánticas (ver figura 15 y 16).

Figura 14. Metodología de análisis cualitativo.

Redes semánticas

Figura 15. Red semántica aspectos positivos.

Año 2 Vol. 2 Nº 2 – 2013 (Págs. 104-121)
ISSN - 2322-9292

Figura 16. Red semántica aspectos negativo.

CONCLUSIONES

El modelo conductista es el preponderante en los contenidos, tanto en las experiencias de práctica docente como en los syllabus de los profesores colaboradores del estudio. El modelo estructuralista o progresista fue encontrado en los demás elementos pedagógicos.

La mayoría de docentes hacen explícitos sus objetivos pedagógicos y prefieren

Año 2 Vol. 2 Nº 2 – 2013 (Págs. 104-121)
ISSN - 2322-9292

redactarlos como objetivos y metas, en lugar de darles sentido de competencias.

Para las actividades dentro y fuera de clase, existe una preferencia marcada por el trabajo tradicional (cátedra magistral y lecturas complementarias) y, muy poco utilizan actividades como juegos de roles que destacan el trabajo del estudiante tanto individual como colectivo.
Los docentes aún no cambian el aula, bien sean equipadas con tecnología o no, como espacio físico para sus clases, ni tampoco utilizan en ella recursos diferentes a los tradicionales (documentación digital e impresa). Los entornos virtuales, software educativo y especializado son muy poco manejados en clase, ni tampoco existe la posibilidad de trabajo en equipo a través de herramientas en línea.

Los proyectos y exámenes escritos u orales, son una señal marcada de que los docentes combinan los modelos pedagógicos conductista y progresista en su concepción de evaluación. El portafolio como oportunidad de aprendizaje y al mismo tiempo de evaluación no es utilizada por los docentes colaboradores del estudio.

El acercamiento a problemas reales que permitan discusión, trabajo tanto individual como en pares y equipos y, brinden la oportunidad de adquirir experiencia práctica en el hacer, es el punto clave dentro de la enseñanza de los cursos de diseño de software.

Las mayores dificultades en la enseñanza del curso de diseño de software son: el escaso tiempo para desarrollar las temáticas, grupos con niveles académicos heterogéneos, aulas sin el software especializado necesario, grupos numerosos, evolución vertiginosa de la profesión, fuerte cultura del código y prueba apoyada por el sector productivo, textos con desarrollo de temas inútiles y uso de estrategias pedagógicas que no benefician el desarrollo de competencias y aprendizaje autónomo del estudiante.

Incluir gradualmente aprendizaje activo en las clases universitarias contribuye a superar los obstáculos presentes en el proceso enseñanza – aprendizaje tradicional.

Tener claro los componentes que estructuran una didáctica específica permite construir un quehacer pedagógico íntegro que exalta la labor docente.

Tener en cuenta en la elaboración de una didáctica específica en el área de diseño de software los contenidos de aprendizaje establecidos a nivel internacional por organizaciones académicas, permite a los docentes ser coherentes con las tendencias y necesidades de conocimiento a nivel mundial y no únicamente local.

Año 2 Vol. 2 Nº 2 – 2013 (Págs. 104-121)
ISSN - 2322-9292

REFERENCIAS BIBLIOGRÁFICAS

Academic Ranking of World Universities ARWU. (2009). *Academic Ranking of World Universities in Engineering/Technology and Computer Sciences - 2009* . Recuperado el 23 de Julio de 2009, de Academic Ranking of World Universities in Engineering/Technology and Computer Sciences - 2009 : http://www.arwu.org/FieldENG2009.jsp

ACM, IEEE Computer Society and AIS. (2005). *Computing Curricula 2005. The Overview Report.* Recuperado el 15 de 02 de 2010, de Computing Curricula 2005. The Overview Report: http://www.acm.org/education/education/curric_vols/CC2005-March06Final.pdf

IEEE-CS. (2004). *SWEBOK: Guide to the Software Engineering - Body of Knowledge.* USA: IEEE-CS.

Gascon, J. (2007). *Evolución de la didáctica de las matemáticas como disciplina científica.* Barcelona: Editorial Universidad Autónoma.

Mallart, J. N. (2000). *Capítulo 1. Didáctica: Concepto, objeto y finalidades.* Recuperado el 7 de Abril de 2010, de http://www.xtec.es/~tperulle/act0696/notesUned/tema1.pdf

SWEBOK. (2004). *Software Engineering body of knowledge.* PierreBourque. Ecole de Technoligies superienre.

Sommerville, I. (2008). *Ingeniería de Software.* Madrid España: Pearson Education.

Año 2 Vol. 2 Nº 2 – 2013 (Págs. 104-121)
ISSN - 2322-9292