DIDÁCTICA DEL DISEÑO DE SOFTWARE

JESÚS INSUASTY PORTILLA
RÓBINSON ANDRÉS JIMÉNEZ TOLEDO
ÁLVARO ALEXANDER MARTÍNEZ NAVARRO

UNIVERSIDAD DE NARIÑO
VICERRECTORÍA DE INVESTIGACIONES, POSTGRADOS Y RELACIONES INTERNACIONALES
FACULTAD DE EDUCACIÓN
MAESTRÍA EN DOCENCIA UNIVERSITARIA
SAN JUAN DE PASTO
2010

DIDÁCTICA DEL DISEÑO DE SOFTWARE

JESÚS INSUASTY PORTILLA
RÓBINSON ANDRÉS JIMÉNEZ TOLEDO
ÁLVARO ALEXANDER MARTÍNEZ NAVARRO

Trabajo de Grado presentado como requisito para optar al Título de Magister en Docencia Universitaria.

Asesor:
Magister DELIO GÓMEZ

UNIVERSIDAD DE NARIÑO
VICERRECTORÍA DE INVESTIGACIONES, POSTGRADOS Y RELACIONES INTERNACIONALES
FACULTAD DE EDUCACIÓN
MAESTRÍA EN DOCENCIA UNIVERSITARIA
SAN JUAN DE PASTO
2010

“Las ideas y conclusiones aportadas en el trabajo de grado, son responsabilidad exclusivas de sus autores”

Artículo 1° del acuerdo No. 324 de Octubre 11 de 1966 emanado del Honorable Consejo Directivo de la Universidad de Nariño

Nota de aceptación

_____

_____

_____

_____

_____

_____

_____

Presidente del Jurado

_____

Jurado

_____

Jurado

_____

Jurado

San Juan de Pasto, Noviembre 22 de 2010

A David Alejandro mi hijo y a Carolina D. mi esposa por ser las luces que guían mi camino.
RÓBINSON

A mis padres por darme la vida y a ti por ser mi sueño eterno
ÁLVARO

A Lorena, el poder supremo que mueve el motor de mi vida
JESÚS

AGRADECIMIENTOS

Por sus conocimientos y valiosos aportes, agradecemos a nuestros profesores de maestría, a nuestro asesor Mg. Delio Gómez y a nuestro director de línea de investigación Dr. Roberto Ramírez.

Por su decisiva y valiosa información, agradecemos de forma especial a los profesores participantes en la investigación: Daniel Jackson (Massachusetts Institute of Technology - USA), Ralph Johnson (University of Illinois @ Urbana Champaign – USA), Dewayne E. Perry (University of Texas @ Austin – USA), Jonathan Aldrich (Carnegie Mellon University – USA), Tevfik Bultan (University of California @ Santa Barbara – USA), George Candea (EPFL – Swiss Federal Institute of Technology), Alexander L. Wolf (Imperial College London – UK), David Notkin (University of Washington – USA), Premkumar T. Devanbu (University of California @ Davis – USA), Kenneth M. Anderson (University of Colorado @ Boulden – USA), Joseph Vybihal (McGill University – USA), Robert Duvall (Duke University – USA), John Keklak (Boston University – USA), P.G. Kluit (Delft University of Technology – Holland), Paolo Bottoni (University of Rome "La Sapienza"), Diego Hernán Montoya Bedoya (Escuela de Ingeniería de Antioquia – COLOMBIA), Alexander Barón (Universidad de Nariño - COLOMBIA), Carlos Cobos (Universidad del Cauca - COLOMBIA), Armando Rafael Acuña Martínez (Fundación Universitaria San Martín - COLOMBIA).

Y de forma muy especial, agradecemos a la coordinadora de la maestría en docencia universitaria Martha Alicia López Lasso, por su dedicación en pro del enriquecimiento de la calidad de la maestría, su compromiso con el desarrollo académico de la región y su motivación constante que fue el motor que impulsó el desarrollo de nuestro proyecto.

Y finalmente, agradecemos a la Universidad de Nariño por propiciar los escenarios para nuestra formación avanzada en la labor que es el eje de nuestra vida: la docencia.

Los autores.

TABLA DE CONTENIDO

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE ANEXOS

RESUMEN ANALÍTICO DEL ESTUDIO
R.A.E.

CÓDIGO: 98.392.695.
98.398.072.
98.387.169.
PROGRAMA ACADÉMICO: MAESTRÍA EN DOCENCIA UNIVERSITARIA.
AUTORES: JESÚS HOMERO INSUASTY PORTILLA.
RÓBINSON ANDRÉS JIMÉNEZ TOLEDO.
ÁLVARO ALEXANDER MARTÍNEZ NAVARRO.
ASESOR: Mg. DELIO GÓMEZ.
TÍTULO: DIDÁCTICA DEL DISEÑO DE SOFTWARE.
ÁREA DE INVESTIGACIÓN: MEJORAMIENTO CUALITATIVO DE LA EDUCACIÓN SUPERIOR.
LÍNEA DE INVESTIGACIÓN: PEDAGOGÍA Y DIDÁCTICAS.
PALABRAS CLAVE: Didáctica, diseño de software, aprendizaje activo.

DESCRIPCIÓN

La Didáctica del Diseño de Software es una tesis de maestría en docencia universitaria la cual presenta la metodología de construcción de una estrategia de enseñanza así como la propuesta didáctica como tal, esta propuesta fue el resultado de la investigación de contexto internacional.

La construcción de la Didáctica del Diseño de Software es el objetivo general de la investigación; consecuentemente, la metodología para la construcción de dicha didáctica se llevó a cabo por fases en el sentido de dar cumplimiento a los objetivos específicos de la investigación.

Los referentes conceptuales en el campo del diseño de software fueron tomados en la fase de revisión documental. En ese punto, los documentos internacionales llamados *Computing Curricula 2005: The Overview Report* (ACM, IEEE-CS & AIS, 2005) y *SWEBOK: Guide to the Software Engineering Body of Knowledge* (IEEE-CS, 2004) ayudaron a soportar la estructura conceptual de la disciplina. Además, la investigación incluyó referencias acerca de los fundamentos de didáctica y su estructura, así como las bases teóricas del aprendizaje activo.

Las características de las experiencias en enseñanza de los profesores tanto extranjeros como nacionales permitieron la elaboración de una síntesis de casos exitosos. Es importante resaltar que los profesores del contexto nacional e internacional fueron seleccionados de acuerdo con la clasificación académica de las universidades mundiales

del 2009 en el área de ingeniería/tecnología y ciencias de la computación y los resultados de las universidades nacionales según las pruebas ECAES 2009.

Todos los elementos anteriores permitieron la construcción de la Didáctica del Diseño de Software con un enfoque de aprendizaje activo, la propuesta no pretende ser un estándar pero si una guía para el cuerpo profesoral de la disciplina a nivel mundial.

CONTENIDO

Este documento se encuentra organizado en tres capítulos, en los que el lector encontrará información detallada sobre el proceso investigativo llevado a cabo, los resultados encontrados tras el cumplimiento de cada uno de los objetivos específicos y la elaboración de la propuesta didáctica. El capítulo 1 presenta los antecedentes que incluyen 25 Investigaciones, que ayudaron a tejer el estado del arte de la enseñanza del diseño de software en el contexto internacional y nacional, dando como resultado la identificación de elementos comunes asociados al campo educativo y de la enseñanza. Así mismo presenta el compilado de teorías que fundamentan la investigación, entre las que se encuentran el diseño de software, didáctica y aprendizaje activo. El capítulo 2 describe los resultados obtenidos por el cumplimiento de objetivos específicos de la investigación, indicándose las metodologías para: a.) Precisar los referentes conceptuales internacionales sobre diseño de software b.) Identificar los elementos constitutivos de la Didáctica de Diseño de Software c.) Caracterizar la enseñanza del diseño de software. Finalmente el capítulo 3 hace la presentacion formal de la Didáctica de Diseño de software, que es el resultado del análisis y síntesis de tres fuentes de información: Profesores nacionales y extranjeros en el campo del diseño de software, dos documentos disciplinares *The Joint Task Force for Computing Curricula 2005* y *SWEBOK Guide 2004*, y la fundamentación teórica sobre aprendizaje activo. Gracias al experto grupo de profesores, La Didáctica del Diseño de Software establece un e*stado del arte* relacionado con la enseñanza del diseño de software, obteniendo las mejores características tales como contenidos, ejercicios, clases, trabajo en equipo, proyectos, evaluaciones, etc. Se pretende servir como guía en la enseñanza de un curso introductorio de diseño de software.

METODOLOGÍA

Paradigma, enfoque y tipo de investigación.

El conjunto de normas, presupuestos, reglas, procedimientos y creencias básicas que sirven de guía a la presente investigación se encuentra descrito a continuación:

Paradigma. Esta investigación se enmarca principalmente en el paradigma *cualitativo*, debido a que se propende por presentar una alternativa práctica de solución, como fruto de la descripción, interpretación, auto reflexión y auto entendimiento. Sin embargo no se descarta el uso de algunos elementos de la metodología cuantitativa.

Enfoque. El estudio se fundamenta bajo el enfoque *critico-social*, que pretende que la investigación se encamine al logro de una conciencia auto-reflexiva 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.

Tipo. El tipo de investigación es 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.

CONCLUSIONES

El modelo conductista es el que se encuentra en los contenidos, tanto en las experiencias de práctica docente como en los syllabus de los profesores colaboradores de la investigación. El modelo estructuralista o progresista fue encontrado en los demás elementos pedagógicos. La mayoría de docentes hacen explícitos sus objetivos pedagógicos y prefieren 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 colectivamente.

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 de la investigación.

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. Además, tener claro los componentes que estructuran la Didáctica del Diseño de Software, permite

construir un quehacer pedagógico íntegro que exalta la labor docente; y 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.

RECOMENDACIONES

Resultaría interesante se avance en los contenidos del curso diseño de software dando pasos hacia modelos pedagógicos como el socialista, que tienen como prioridad la naturaleza y vida social.

En el mundo globalizado en que vivimos, los futuros profesionales deberían ser capaces de desenvolverse con su saber en cualquier contexto y de forma colaborativa, por tanto debería pensarse en la posibilidad de perfilar la didáctica aquí propuesta hacia el desarrollo de competencias y trabajo en equipo en todos sus elementos.

Contrastar los excelentes resultados de las prácticas docentes internacionales con sus metodologías ortodoxas de enseñanza, aportaría de manera significativa al campo de la didáctica y a futuro, implementar en la Didáctica del Diseño de Software la evaluación siguiendo el modelo pedagógico socialista, dará luces en el proceso de avance en la didáctica aquí propuesta.

Aplicar la Didáctica del Diseño de Software propuesta en la universidad y estudiar los resultados obtenidos puede convertirse en un trabajo interesante de investigación y contribuiría a mejorarla.

BIBLIOGRAFÍA

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

Díaz Barriga , A. (1997). Didáctica y currículum. Paidós.

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

Silberman, M. (1998). Aprendizaje activo, 101 estrategias para enseñar cualquier tema. Argentina: Troquel S.A.

Villalobos, J., & Vela, M. (2007). CUPI2-An Active Learning and Problem Based Learning Approach to Teaching Programming. 8th ALE International Workshop, 8.

ANALITICAL ABSTRACT OF THE STUDY
A.A.S

CODE: 98.392.695.
98.398.072.
98.387.169.
ACADEMIC PROGRAM: MAESTRÍA EN DOCENCIA UNIVERSITARIA.
AUTHORS: JESÚS HOMERO INSUASTY PORTILLA.
RÓBINSON ANDRÉS JIMÉNEZ TOLEDO.
ÁLVARO ALEXANDER MARTÍNEZ NAVARRO.
ADVISOR: Mg. DELIO GÓMEZ.
TITLE: DIDÁCTICA DEL DISEÑO DE SOFTWARE.
RESEARCH AREA: QUALITATIVE IMPROVEMENT OF HIGHER EDUCATION
RESEARCH LINE: PEDAGOGY AND DIDACTICS.
KEYWORDS: Didactic, software design, active learning.

DESCRIPTION

Didactic of Software Design is a master thesis in high education teaching, which shows a teaching strategy's construction methodology and a didactic proposal itself, this proposal was the result of a worldwide context research.

The making of Didactic of Software Design is the main objective of this research; consequently, this methodology was done by phases in order to accomplish the research's specific objectives.

Concept references in the software design field were taken in the phase of documentary review. In that point, international documents called *Computing Curricula 2005: The Overview Report* (ACM, IEEE-CS & AIS, 2005) and *SWEBOK: Guide to the Software Engineering Body of Knowledge* (IEEE-CS, 2004) helped to support the concept structure of the discipline. Furthermore, the research included references about Didactic foundations and its structure, and Active Learning's theoretical bases.

Features of the professors' teaching experiences permitted to construct a synthesis of succeed cases. It is important to note that foreign and national professors were selected according to the Academic Ranking of World Universities in 2009 in Engineering/Technology and Computer Science, and the results of National Universities according to ECAES 2009.

All those items above permitted the construction of the Didactic of Software Design with an Active Learning approach, the proposal does not pretend to be a standard but a guide for the worldwide faculty staff of the discipline.

CONTENT

This document is organized into three chapters, the reader will find detailed information about the research process undertaken, the findings upon completion of each of the specific objectives and the development of the didactic. Chapter 1 provides background including 25 Research, who helped weave the state of the art of teaching software design in the international and national context, resulting in the identification of common elements associated with the field of education and teaching. It also presents the compilation of theories underlying the research, among which are the software design, teaching and active learning. Chapter 2 describes the results obtained by the achievement of specific objectives of the investigation, indicating the methodologies for: a.) Specify the conceptual referents international software design b.) Identify the elements of the Teaching Software Design c.) Characterize the teaching of software design. Finally, chapter 3 is the formal presentation of the Didactic Design software, which is the result of analysis and synthesis of three sources: domestic and foreign professors in the field of software design, two disciplinary documents for the Joint Task Force Computing Curricula 2005 and SWEBOK Guide 2004, and active learning theoretical framework. Due to the expert group of teachers, The Teaching Software Design provides a state of art related to teaching software design, obtaining the best features such as content, exercises, classes, team work, projects, assessments, etc.. It is intended to serve as a guide in teaching an introductory course in software design.

METHODOLOGY

Paradigm, approach and research type.

The set of rules, concepts, procedures and basic beliefs to guide this research is described below:

Paradigm. This research is working in the qualitative Paradigm because it try to show an alternative practices in order to face the main problem; common activities in this paradigm are: description, interpretation, self-reflection and self-understanding. However, usage of some elements of Quantitative methodologies is not discarded.

Approach. The study is based on *critic-social* approach, which pretends to reach a self-reflexive conscience and to critic in order to transform reality according to the context, where dialog, debate and praxis (theory-practice relationship) be the teaching labor's axis.

Research Type. The type of the research is collaborative. This research type handled by a group of professionals which defines, harmonizes, and conceives a study of common interest in order to contribute to the construction of Pedagogical and Didactic knowledge. This research type facilitates the integration of these new elements to the education labors.

CONCLUSIONS

Behaviorist Model is present in worldwide course contents, teaching experiences and syllabus. Progresist Model (Structuralism) was found in the rest of pedagogical items.

Most professors make explicit their educational objectives and they prefer to write them as targets and goals, instead of giving them a sense of competence.

For activities inside and outside the classroom, there is a strong preference for the traditional work (lectures and complementary readings), and very little use as role-playing activities that highlight student work both individually and collectively.

Professors still do not change the classroom, whether or not fitted with technology, as physical space for their classes, nor use it other than the traditional resources (digital and printed documentation), virtual environments, and specialized educational software are poorly managed in the classroom, nor is there the possibility of teamwork through online tools.

Projects and written tests or oral, they are a strong signal that professors combine behavioral models of education and progressive in their understanding of assessment. The portfolio, as an opportunity for learning and assessment at the same time, is not used by the educational research collaborators.

The approach to real problems allows discussion, working individually and in pairs and teams, and provides the opportunity to gain practical experience in the making. It is the key point in the teaching of software design courses.

The greatest difficulties in teaching software design course are: 1) Limited time to develop the topic groups with academic standards heterogeneous classrooms without the necessary specialized software. 2) Large groups 3) Rapid evolution of the discipline. 4) Strong code culture and testing supported by the productive sector development. 5) Textbooks with useless items. 6) Use of teaching strategies that do not benefit the development of independent learning skills.

Introduce gradually active learning in college classes to help overcome barriers in the teaching - traditional learning.

Be clear about the components that make up a specific teaching can build a full pedagogical work that enhances the teaching profession.

Note the learning content established internationally by academic organizations in the development of a specific teaching in the area of software design, That content allows professors to be consistent with the trends and needs of knowledge worldwide and not just locally.

RECOMENDATIONS

It would be interesting to progress in the contents of the software design course moving towards pedagogical models like the socialist, whose priority is the nature and social life.

In the globalized world we live in, future professionals should be able to work with his knowledge in any context using collaborative ways, therefore, they should think about the possibility of teaching using this proposal to the development of skills and teamwork.

The contrast between the excellent performance of international teaching practices of professors and their unorthodox methods of teaching will provide new knowledge to the field of teaching. Furthermore, in the future, the implementation of evaluation in the Didactic of Software Design using a pedagogical model will advance the process of the proposed teaching strategy.

Apply the proposed Didactic of Software Design in universities and study the results can become an interesting research work which help to improve it.

BIBLIOGRAPHY

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

Díaz Barriga , A. (1997). Didáctica y currículum. Paidós.

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

Silberman, M. (1998). Aprendizaje activo, 101 estrategias para enseñar cualquier tema. Argentina: Troquel S.A.

Villalobos, J., & Vela, M. (2007). CUPI2-An Active Learning and Problem Based Learning Approach to Teaching Programming. 8th ALE International Workshop, 8.

INTRODUCCIÓN

La presente investigación se orienta en primera instancia, en la temática relacionada con la didáctica, que a lo largo de la historia en la educación, se trata de definir, desde la perspectiva etimológica, desde sus inicios con autores como Wolfgang Ratke (1571-1635) y Jan Amos Komenský (1592-1670), y desde autores contemporáneos como Mallart, quien 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).

En segunda instancia, esta investigación se aborda desde la temática de la enseñanza del diseño de software para las disciplinas internacionales formalizadas en la Association for Computing Machinery (ACM, 2009), quien es la más grande e importante sociedad informática, científica y académica del mundo en su área, junto con la IEEE – Computer Society (IEEE, 2009) como 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 Ian Sommerville (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, según José Ignacio Peláez Sánchez (Peláez Sánchez, 2008), las reflexiones sobre la práctica docente en dicho campo son muy escazas, aspecto que ocasiona un desconocimiento general sobre la labor docente en cuanto a diseño de software. En este sentido, la enseñanza del diseño de software no cuenta con la misma producción de conocimiento, como sí es manifiesta en la evolución de la didáctica de las matemáticas según (Gascon, 2007); por consiguiente, la investigación de esta problemática se realizó por el interés de estudiar un campo poco explorado de la enseñanza del diseño de software, ya que a nivel internacional se han logrado avances muy significativos en el área estrictamente disciplinar como lo indican (Cavero Barca, Vela Sánchez, & Marcos Martínez, 2008), pero existen globalmente, muy pocos referentes, aportes y reflexiones en relación con la forma en que se enseñan estos temas. Esta situación constituye el interés académico y motivación para pretender mejorar

procesos de enseñanza aprendizaje, que traten de incidir en una mejor comprensión de estos saberes.

Los aportes del presente proyecto investigativo al campo de estudio de la Didáctica del Diseño de Software 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 Shanghai Jiao Tong en China (Academic Ranking of World Universities, 2009); a nivel nacional, se identificaron las 25 mejores universidades según el ranking dado por el Instituto Latinoamericano de Liderazgo (ILL, 2009) con los mejores resultados en los exámenes de calidad de la educación superior ECAES, 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, esta información sirvió de insumo a la investigación.

Esta investigación se desarrolló dentro del campo de estudio de la didáctica en la educación superior, específicamente en la enseñanza del diseño de software para las disciplinas internacionales formalizadas en la Association for Computing Machinery ACM e IEEE - Computer Society. En relación con los factores de orden subjetivo que permitieron definir la factibilidad del proceso investigativo de la temática abordada, se identificaron aspectos como: capacidad para desarrollarlo, es decir, dominio disciplinar y experiencia de los docentes investigadores; tiempo asignado en las respectivas universidades en la modalidad de investigación profesoral, aproximadamente tres horas/semana por investigador y tiempo de dedicación extra laboral, aproximadamente 7 horas/semana; en cuanto a recursos materiales se cuenta con bibliografía, membrecías al Special Interest Group on Computer Science Education SIGCSE - ACM, membrecía San Jose State University (SJSU) Articles & Databases, membrecía San Jose Public Library (SJPL) Articles & Databases, membrecía ACM Digital Library, membrecía Education Research Complete, y membrecía Education Full Text; el talento humano con que contó el trabajo son tres investigadores, un asesor, un tutor de línea y expertos nacionales e internacionales en la enseñanza del diseño de software. Entre los factores de orden objetivo que le dan importancia al proceso de investigación se pueden anotar: interés por brindar a la comunidad docente relacionada con las ciencias de la computación una didáctica de enseñanza en el área del diseño de software; utilidad del tema porque trata de enriquecer la labor docente en el campo de la enseñanza del diseño de software; nuevo enfoque, debido a la presentación de una propuesta internacional sobre Didáctica del Diseño de Software, basada en las experiencias exitosas de docentes nacionales e internacionales en esta área.

El tema investigado se delimita con base en el tiempo y espacio así: en relación con el tiempo, el tema emerge en el año 2005 con reflexiones iniciales de los autores en torno al desempeño de los estudiantes en la construcción de software, cambios permanentes en lo disciplinar y pedagógico, y expectativas en las nuevas tendencias. En relación con el

espacio, se seleccionó los cursos de diseño de software de las carreras con enfoque disciplinar de la ACM e IEEE–CS a nivel mundial.

Para mayor comprensión del problema de investigación, se estableció, como se muestra en la tabla 1, los elementos que permitieron identificar el planteamiento del problema y establecer así, la dirección del presente estudio.

Tabla 1 *Identificación de síntomas, causas diagnóstico y consecuencias del problema de investigación.*

Síntomas
1. Se evidencia que la enseñanza en los cursos de diseño de software, actualmente se realiza con base en la experiencia técnica del profesor, mas no acompañada de un proceso pedagógico formal. 2. Escasa producción divulgada sobre la enseñanza de diseño de software, ya que existen muy pocos documentos y reflexiones sobre la labor docente frente a esta temática

Causas
1. La construcción aislada del curso de diseño de software por parte del profesor a cargo, hace que se descarten las experiencias de sus pares del contexto internacional. 2. Son notables los grandes avances dentro del campo disciplinar del diseño de software (Helm, Johnson, & Vlissides, 2000; Garlan & Shaw, 1994 y Jacobson, 2004); no obstante, se puede apreciar que el campo educativo, específicamente la didáctica y la labor de enseñanza, no tienen el desarrollo como complemento a la disciplina especifica. 3. Los profesores no sistematizan su labor de enseñanza y solo se dedican a la producción de conocimiento de la disciplina *per sé*, haciéndose evidente la escaza reflexión sobre el ejercicio de la docencia.

Diagnóstico
1. Ausencia de un *corpus* teórico que permita sistematizar el ejercicio de la enseñanza del diseño de software basado en las experiencias de profesores en el contexto internacional.

Consecuencias
Las situaciones que pueden generarse por la prevalencia del problema son: 1. Pérdida de conocimiento asociado al ejercicio de la docencia, debido a que los profesores que realizan buenas prácticas de enseñanza no las documentan ni las divulgan, ocasiona la escaza posibilidad de ser analizadas, evaluadas, reflexionadas, criticadas, compartidas ni enriquecidas por académicos de este campo. 2. Alto grado de subjetividad en la preparación y enseñanza del curso de diseño de software por parte de los profesores, debido a la falta de un punto de referencia sobre su didáctica como producto de una investigación.

Consecuencias
3. Subvaloración de la docencia como profesión, ocasionada por la falta de reflexión sobre la labor de la enseñanza y la escaza fundamentación teórica para soportar dicho ejercicio. 4. Escaza integración e interrelación entre el campo disciplinar y la didáctica propia del mismo.

Fuente: Esta investigación

Luego de tener claro el problema de investigación, a continuación se presentan los interrogantes que orientaron el estudio. La gran pregunta se concibió en términos de ¿cómo aportar al campo de la enseñanza de diseño de software en el contexto internacional?, para su realización fue necesario planearse tres cuestiones más, a saber: ¿cuáles son los referentes conceptuales internacionales sobre el diseño de software?, ¿cuál es la estructura de una didáctica específica para el diseño de software? y ¿cuáles son las características de las experiencias de enseñanza de diseño de software a nivel nacional e internacional?, a dichas preguntas se respondieron a través del objetivo general: Aportar al campo de la enseñanza de diseño de software en el contexto internacional mediante una *didáctica específica* y los objetivos específicos: Precisar los referentes conceptuales internacionales sobre el diseño de software; determinar la estructura de la Didáctica del Diseño de Software; caracterizar las experiencias de enseñanza de diseño de software a nivel nacional e internacional.

El estudio tiene acogida porque a nivel internacional, las más importantes asociaciones de computación e informática 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 Association for Computing Machinery, AIS – The Association for Information Systems y la IEEE-CS – The Computer Society), sugiere la presencia de cinco disciplinas específicas en el campo, a saber: a.) Computer engineering (ingeniería en computación), b.) Computer science (ciencia en computación), c.) Information systems (sistemas de información), d.) Information technology (tecnología de información), e.) Software engineering (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 investigación es interesante puesto que enfoca su estudio a un campo poco explorado que es la enseñanza del diseño de software; ya que a nivel internacional se han logrado avances muy significativos en el área estrictamente disciplinar como lo argumentan algunos autores (indicadas en las investigaciones de Gama, Helm, Johnson, & Vlissides, 2000; Garlan & Shaw, 1994 y Jacobson, 2004) para quienes la ingeniería de software como disciplina ha evolucionado significativamente en lo que se refiere a modelos conceptuales y herramientas de trabajo. Sin embargo, existen globalmente muy pocos aportes y reflexiones en relación con la forma en que se enseñan estos temas.

La caracterización de las experiencias de enseñanza de diseño de software brinda una gran utilidad porque permite evidenciar la situación actual de cómo es la labor docente en el contexto internacional. Con la investigación se tendrá la posibilidad explorar el estado del arte sobre la enseñanza del diseño de software y de esta forma, enriquecer la propuesta didáctica.

La estrategia formulada por los investigadores 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, porque 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 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.

Esta investigación se enmarca principalmente en el *paradigma cualitativo*, debido a que se propende por presentar una alternativa práctica de solución, como fruto de la descripción, interpretación, auto reflexión y auto entendimiento. De igual manera, se usaron algunas técnicas de recolección y de procesamiento de información propias de la metodología cuantitativa, que aseguraron hallazgos muy importantes en esta investigación. El estudio se fundamenta bajo el *enfoque crítico-social*, que pretende que la investigación se encamine al logro de una conciencia autoreflexiva 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. El *tipo de investigación es 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 acuerdo con las características de la investigación, se determinó trabajar con el muestreo no probabilístico específicamente con procedimientos de muestreo intencional. La población objeto de estudio para la presente investigación fueron los profesores titulares del curso en diseño de software de las más prestigiosas universidades del mundo y de Colombia en el campo de la computación. Para tal efecto a nivel mundial, se seleccionó las 80 universidades que forman parte del estudio realizado para el 2009 por el Institute of Higher Education de la Universidad Shanghai Jiao Tong (Academic Ranking of World Universities ARWU, 2009); y a nivel nacional se seleccionó las universidades, donde el programa de ingeniería de sistemas se destacó por sus resultados en los exámenes de calidad de la educación superior (Observatorio de la Universidad Colombiana, 2009). El papel que cumplen dichos expertos en la presente investigación

fue fundamental, puesto que, fueron la fuente de información principal en tanto a su labor docente se refiere.

De acuerdo con lo anterior se obtuvo los siguientes resultados:

Tabla 2 *Detalle de resultados para población y muestra.*

Categorías	Población		Muestra
	No. universidades convocadas	Número de profesores titulares convocados	Número profesores titulares participantes
Internacional	80	80	15
Nacional	9	9	5

Fuente: Esta investigación

En detalle el listado de las 80 universidades internacionales se puede consultar en el Anexo A. El listado de los 80 profesores titulares internacionales de cátedra convocados se puede observar en el Anexo B. En el Anexo C está el listado de los 15 profesores titulares internacionales de cátedra que participaron en esta investigación y su registro documental (datos de contacto, profesionales, investigaciones, publicaciones). De igual manera el listado de los 9 profesores convocados de las universidades nacionales se puede consultar en el Anexo D.

Para una mayor comprensión de las técnicas usadas en el proceso investigativo, en relación con la etapa de recolección y procesamiento de datos, se tabuló por cada uno de los objetivos específicos, como lo muestra la tabla 3.

Tabla 3 *Técnicas de recolección de datos por objetivos.*

Objetivos	Técnica de recolección	Técnica de Procesamiento
Precisar los referentes conceptuales internacionales sobre el diseño de software.	Revisión documental	Elaboración de una matriz de referentes conceptuales.
Determinar la estructura de la didáctica específica de diseño de software.	Revisión documental	Elaboración de matriz de elementos de la didáctica. Construcción de síntesis basada en los mapas conceptuales.
Caracterizar las experiencias de enseñanza de diseño de software a nivel nacional e internacional.	Encuesta a través de correo electrónico. Revisión documental	Análisis estadístico. Redes semánticas. Matriz de características.

Fuente: Esta investigación

Debido al grado de novedad que presenta la investigación, no existe un instrumento válido y confiable, que sirva para la recogida de los datos. Por lo tanto, se construyó una

encuesta online aplicada a los docentes expertos en cursos de diseño de software ver Anexo E y matrices de vaciado de información para identificación de categorías didácticas y caracterización de experiencias de enseñanza.

Para la creación de instrumentos se tuvo en cuenta los siguientes pasos:

1. Determinar la información que se quiere recolectar, es decir en concordancia con los objetivos y las categorías o elementos de la didáctica.
2. Decidir sobre el tipo de fuente donde se obtendría la información.
3. Considerar las características de la población en que se aplicaría.
4. Determinar el tipo de instrumento indicado.
5. Elegir la forma de elaborar las preguntas y tipos de preguntas.
6. Determinar la estructura del instrumento.

La encuesta que se utilizó es de sondeo de *opinión vía correo electrónico*, construida para la investigación, con el objetivo de indagar los expertos en diseño de software sobre cada una de las categorías de la didáctica. A continuación en la figura 1, se ilustra la estructura general de la encuesta, su la relación con las categorías de la didáctica y revisión documental.

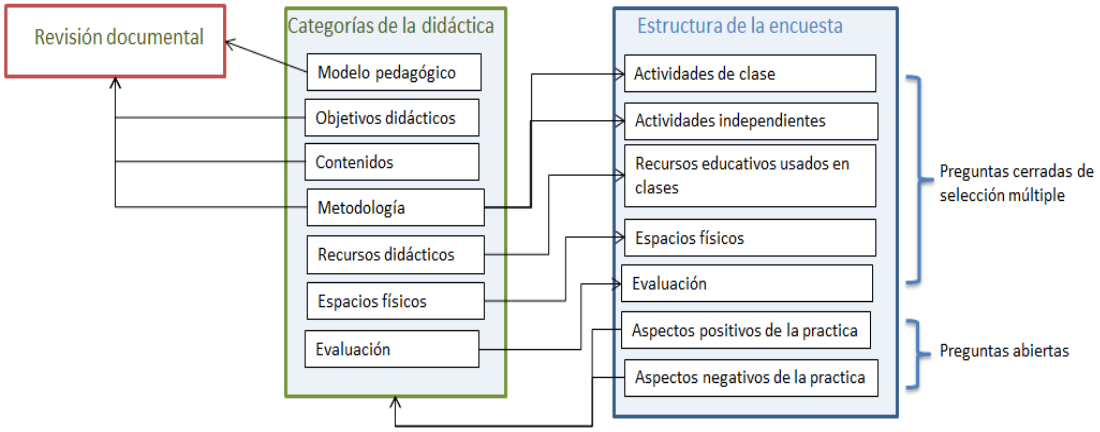


Figura 1 *Relación instrumento de recolección de datos y categorías didácticas.*
Fuente: Esta investigación

Este documento se encuentra organizado en tres capítulos, en los que el lector encontrará información detallada sobre el proceso investigativo llevado a cabo, los resultados encontrados tras el cumplimiento de cada uno de los objetivos específicos y la elaboración de la propuesta didáctica.

El capítulo 1 presenta los antecedentes que incluyen 25 Investigaciones, que ayudaron a tejer el estado del arte de la enseñanza del diseño de software en el contexto internacional y nacional, dando como resultado la identificación de elementos comunes asociados al campo educativo y de la enseñanza. Así mismo presenta el compilado de teorías que

fundamentan la investigación, entre las que se encuentran el diseño de software, didáctica y aprendizaje activo.

El capítulo 2 describe los resultados obtenidos por el cumplimiento de objetivos específicos de la investigación, indicándose las metodologías para: a.) Precisar los referentes conceptuales internacionales sobre diseño de software b.) Identificar los elementos constitutivos de la Didáctica de Diseño de Software c.) Caracterizar la enseñanza del diseño de software.

Finalmente el capítulo 3 hace la presentacion formal de la Didáctica de Diseño de software, que es el resultado del análisis y síntesis de tres fuentes de información: Profesores nacionales y extranjeros en el campo del diseño de software, dos documentos disciplinares *The Joint Task Force for Computing Curricula 2005* y *SWEBOK Guide 2004*, y la fundamentación teórica sobre aprendizaje activo. Gracias al experto grupo de profesores, La Didáctica del Diseño de Software establece un e*stado del arte* relacionado con la enseñanza del diseño de software, obteniendo las mejores características tales como contenidos, ejercicios, clases, trabajo en equipo, proyectos, evaluaciones, etc. Se pretende servir como guía en la enseñanza de un curso introductorio de diseño de software.

Los hallazgos en la presente investigación permiten asegurar que para el componente de *los contenidos* del diseño de software, predomina el modelo conductista, identificado tanto en las experiencias de práctica docente como en los syllabus de los profesores colaboradores de la investigación. El modelo estructuralista o progresista fue encontrado en los demás componentes pedagógicos. La mayoría de docentes hacen explícitos sus objetivos pedagógicos y prefieren 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 colectivamente.

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 de la investigación.

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. Además, tener claro los componentes que estructuran la Didáctica del Diseño de Software, permite construir un quehacer pedagógico íntegro que exalta la labor docente; y 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.

1. FUNDAMENTACIÓN TEÓRICA

1.1 Antecedentes.

Nuevos enfoques de enseñanza en las ciencias de la computación son necesarios en ambientes de educación superior, ya que la enseñanza actual en programación se muestra como el resultado del entendimiento relativamente pobre de procesos fundamentales de análisis y diseño de sistemas, como se indica el artículo de investigación Teaching software design: a new approach to computer science (Sins-Knight & Upchurch, 2001), es por ello que los autores concluyen que se hace necesario la enseñanza del diseño de software antes de enseñarles a los estudiantes como programar usando lenguajes específicos, es decir, brindar las bases sólidas en diseño que permita abrir más los horizontes de aplicaciones tecnológicas para enfrentar los problemas de hoy. Este pensamiento ha desarrollado una serie de experiencias a nivel internacional sobre la enseñanza del diseño de software. La presente investigación incluye 25 antecedentes consignados en la tabla 4 ordenados cronológicamente, que ayudan a tejer el estado del arte de la enseñanza del diseño de software en el contexto internacional y nacional.

Tabla 4 *Compendio de antecedentes en la enseñanza de diseño de software*

#	Año	Investigación / Lugar	Autores
1	2010	Aprendizaje Basado en Problemas un enfoque de la enseñanza de un curso avanzado de ingeniería de Software, Universidad Xiamen (China).	Ming Qiu, Lin Chen
2	2010	Análisis pedagógico y estructural de cursos de ingeniería de software, Universidad Islámica Internacional (Pakistan), universidad del estado Kent, OH (USA).	M. Usman, J. Khan, M. Hardas, N. Ikram
3	2010	Ingeniería del software, cursos de experiencia práctica, Carnegie Mellon Silicon Valley.	Edward P. Katz
4	2009	Un mundo virtual 3D de la enseñanza y el aprendizaje de cursos de ciencias de la computación en Second Life CUNY, NY (USA), China Universidad Tecnológica (Taiwan).	Ching-Song Wei, Yan Chen, Jiann-Gwo Doong
5	2009	Un curso de ingeniería de software multidimensional, Universidad de Tel-Aviv (Israel).	Ohad Barzilay, Orit Hazzan, Amiram Yehudai
6	2009	Discusión de la capacidad de pensamiento computacional en cursos de enseñanza, Universidad Guilin de Tecnología Electrónica (China).	Deng Zhenrong, Huang Wenming, Dong Rongsheng
7	2009	Metodología de enseñanza para la mejora de la capacidad del pensamiento computacional, Universidad Guilin de Tecnología Electrónica (China).	Huang Wenming, Deng Zhenrong, Dong Rongsheng
8	2009	Enseñar un curso introductorio de ingeniería de software en un programa de ciencias de la computación, IIT Delhi (India).	Pankaj Jalote
9	2009	Enseñanza y práctica de la reforma en el curso de	Chunting Yang ,

#	Año	Investigación / Lugar	Autores
		ingeniería de software, Universidad Zhejiang de Ciencia y Tecnología (China).	Yang Liu
10	2009	La enseñanza de diseño de software mediante un estudio de caso sobre modelos de transformación, Universidad Arcadia PA (USA).	Yanxia Jia, Yonglei Tao
11	2008	Enseñanza de diseño de arquitecturas de software, Universidad de Tecnología Helsinki, Centro de Investigaciones Nokia (Finlandia).	Tomi Männistö, Juha Savolainen, Varvana Myllärniemi
12	2007	Un estudio de eficacia del método de casos de estudio en educación de la ingeniería de software, Instituto Internacional de tecnología de información (India).	Kirti Garg, Vasudeva Varma
13	2007	Proyecto Cupi2, Universidad de Los Andes (Colombia).	Jorge Villalobos
14	2007	Software de enseñanza de la ingeniería: algunas dimensiones importantes, Universidad Atilim (Turquía).	Alok Mishra, Nergiz Ercil Cagiltay , Ozkan Kilic
15	2007	Etapas en la enseñanza del diseño de software, Universidad de Sheffield (UK).	A. J. Cowling
16	2006	Sobre las prácticas orientadas a la educación de la ingeniería de software, Universidad Atlantic, FL (USA), Universidad de Sannio (Italia).	Shihong Huang, Damiano Distante
17	2005	Enseñando un cursos sobre arquitectura de software, Universidad Vrije (Holanda).	Patricia Lago, Hans van Vliet
18	2005	Enseñando diseño avanzado, equipos orientados a proyectos de software, Universidad Nacional de Singapore (Singapur).	Stan Jarzabek, Pin-Kwang Eng
19	2004	Un modelo para SWE, curso de arquitectura de software y diseño, Universidad Politécnica Southern, GA (USA).	Orlando Karam, Kai Qian, Jorge Diaz-Herrera
20	2003	Experiencias en enseñanza de diseño orientado a objetos y cursos de estructuras de datos, universidad del estado de Arizona, AR (USA).	Sheikh I. Ahamed
21	2003	Impacto de diferentes paradigmas sobre el aprendizaje de los estudiantes, Universidad tecnológica Tennessee, TN (USA).	T. Brignall, S. Ramaswamy
22	2002	Enseñanza de procesos unificados para estudiantes de pregrado, Universidad Tecnológica de Vienna (Austria).	Michael Halling, Wolfgang Zuser, Monika Köhle, Stefan Biffl
23	1999	Enseñando diseño de arquitecturas en currículos de pregrado, Instituto de Tecnología Rochester, NY (USA).	J. Fernando Naveda
24	1996	Perspectiva de la educación de ingeniería de software, Universidad Murdoch (Australia).	Geoffrey G. Roy, Valerie E. Veraart
25	1993	Enseñando diseño de software: un nuevo enfoque para escuelas, en ciencias de la computación, Universidad de Massachusetts, MA (USA).	Judith E. Sims-Knight, Richard L. Upchurch

Fuente: Esta investigación

A continuación se resaltan los aspectos principales de cada una de las investigaciones relacionadas con la enseñanza de diseño de software.

1. (2010) Universidad Xiamen– China.

En esta investigación se describe el uso del aprendizaje basado en problemas en un curso de "Ingeniería de Software Avanzado" dado en la Universidad de Xiamen, China. Este enfoque se lleva a cabo en un ambiente de aprendizaje mixto, una combinación aprendizaje presencial y el entorno de eLearning. Un conjunto de proyectos integrados fueron seleccionados como estímulo para el aprendizaje. Tanto la interdisciplinariedad y la colaboración dentro del grupo de aprendizaje fueron utilizadas eficientemente. (Qui & Chen, 2010).

2. (2010) Universidad Islámica Internacional - Pakistan, Universidad del estado Kent, OH – USA.

El propósito de esta investigación es la realización de un enfoque multidimensional para el análisis de los cursos de ingeniería de software. Dos aspectos son importantes para este análisis, a saber: qué tanto es cubierto dentro de los cursos de ingeniería de software y qué tan bien cubiertos están dichos contenidos. El primer aspecto es llamado análisis estructural y el segundo aspecto es denominado análisis pedagógico. (Usman, Khan, Hardas, & Ikram, 2010).

3. (2010) Carnegie Mellon Silicon Valle

El curso práctico de ingeniería de software de la Universidad Carnegie Mellon en Silicon Valley es un curso integrado al currículo el cual provee al estudiante practicante la oportunidad de interactuar con un cliente real. En esta investigación, la retroalimentación dada por los estudiantes ha sido bastante positiva en el curso de ingeniería de software puesto que los estudiantes pueden seleccionar sus proyectos. Los estudiantes trabajan en proyectos reales (y no solamente como una actividad de clase). El cliente real se beneficia con el trabajo del equipo de diseño de software en el desarrollo de una solución significativa a su problema específico a lo largo de un semestre. (Katz, 2010).

4. (2009) Universidad Tecnológica de China– Taiwan

Los factores clave que contribuyen a la alta tasa de desgaste para estudiantes de ciencias de la computación son la pérdida de interés en el material y falta de práctica. A fin de colaborar mejor con los estudiantes y mejorar la tasa de retención, esta investigación ha desarrollado una nueva plataforma de enseñanza y aprendizaje en Second Life para sumergir y envolver a los estudiantes en la misma manera que los videojuegos con la misma estrategia de capturar y mantener la atención de los jugadores. El objetivo de esta investigación es estimular el interés de los alumnos, la motivación, la creatividad y ayudar a superar las dificultades en el estudio. (Wei, Chen, & Doong, 2009).

5. (2009) Universidad de Tel-Aviv– Israel

Aunque modernos planes de estudios incluyen cursos para disciplinas en ingeniería de software, un curso avanzado que resuma sintéticamente los aspectos relevantes del diseño de software es esperado. Este curso permitirá a los jóvenes profesionales obtener una descripción completa de la ingeniería de software, a la experiencia de un proceso genuino de desarrollo de software, y apreciar las relaciones y equilibrios entre los diversos campos de dicha disciplina. Esta investigación propone un marco multidimensional en cursos de ingeniería de software dirigido a dar a los estudiantes una formación integral, un paradigma transversal, práctico, y la experiencia teórica. El curso se evalúa de acuerdo a criterios aceptados destacando el alcance del mismo, las decisiones pedagógicas y pertinencia. Los autores también describen sus experiencias de enseñanza del curso en la Universidad de Tel Aviv y el colegio académico de Tel-Aviv-Yaffo, Israel. (Barzilay, Hazzan, & Yehudai, 2009).

6. (2009) Universidad Guilin de Tecnología Electrónica– China

En la visión del curso de diseño de software, esta investigación analiza la percepción de la esencia del pensamiento computacional y presenta un método para el cultivo de la capacidad de pensamiento computacional mediante el uso de cursos como los medios, y un estudio de caso del curso de estructuras de datos y algoritmos. (Zhenrong, Wenming, & Rongsheng, 2009).

7. (2009) Universidad Guilin de Tecnología Electrónica – China

El requisito interno de los cursos de programación es desarrollar una cultura estudiantil sobre la capacidad de utilizar lenguaje de programación para análisis de problemas y resolución de los mismos. De acuerdo con la características de los propios cursos, en esta investigación se propone un sistema de enseñanza basado en el núcleo del pensamiento de cómputo y relacionados con el cultivo de la capacidad método de reforma, que hace hincapié en la resolución de problemas y del diseño de algoritmos determinando la relación entre el método orientado a objetos y el método orientado al procedimiento en la enseñanza teórica y los diseños de experimentos multicapas y la organización de experimentos basado en proyectos. (Zhenrong, Wenming, & Rongsheng, Programming Courses Teaching Method for Ability Enhancement of Computational Thinking, 2009).

8. (2009) IIT Delhi – India

En esta investigación se analiza un supuesto curso de ingeniería de software como un programa de computadora, teniendo una pre-condición y una post-condición, y luego se analiza las distintas condiciones de pre y post del curso, y algunas cuestiones relacionadas con la enseñanza del mismo. (Jalote, 2009).

9. (2009) Universidad de Ciencia y Tecnología Zhejiang– China

La ingeniería de software es un curso altamente práctico y muy completo, la enseñanza

tradicional es insuficiente en los términos de esta investigación. Se ha propuesto que los mapas conceptuales pueden ser empleados para la enseñanza de la ingeniería de software. El papel y el enfoque de aplicación del mapa conceptual se exploran al utilizar el método de enseñanza orientada a los proyectos, al estimular en los estudiantes el entusiasmo de estudiar, y al llevar a cabo la actividad práctica de la enseñanza sostenible. (Yang & Liu, 2009).

10. (2009) Universidad Arcadia, PA – USA

Esta investigación presenta un estudio de caso sobre la transformación de modelos. El uso inicial en las aulas del estudio de caso demostró que la transformación de modelos es un tema que merecería y puede ofrecer a los estudiantes una experiencia única de aprendizaje que no puede ser fácilmente alcanzable de otro modo. (Jia & Tao, 2009).

11. (2008) Universidad de tecnología Helsinki, Centro de Investigación Nokia– Finlandia

Los problemas de diseño de arquitectura de software real son menos claros que lo que los estudiantes utilizan para el aprendizaje, los casos reales del mundo industrial son difíciles de traer a un salón de clases. Esta investigación diseñó un curso especial en un programa académico dentro del plan de estudios de ingeniería de software, teniendo en cuenta las necesidades de la industria en la enseñanza del problema, la comprensión y la solución a través del diseño de arquitectura de software. En esta investigación se discute la motivación industrial para el curso, el desarrollo del curso hasta su forma actual, y las lecciones aprendidas en la ejecución de dicho curso. (Männistö, Savolainen, & Myllärniemi, 2008).

12. (2007) Instituto Internacional de Tecnología de Información - India

Los educadores de ingeniería del software han venido abogando por el uso de enfoques no-convencionales para la educación en diseño de software desde hace algún tiempo. En este contexto, se realizó la investigación-acción para comparar la efectividad de un enfoque de estudio de caso con cátedras convencionales. Se diseñó y se impartió un curso por primera vez en ingeniería de software, que involucró el enfoque de estudio de caso, así como el enfoque de la clase tradicional. Los resultados corroboran que el enfoque de estudio de casos es más eficaz e interesante para aprendizaje que el enfoque de la clase tradicional. Este trabajo aporta datos empíricos para apoyar ese enfoque del estudio de caso ya que es muy eficaz en la educación de la ingeniería de software y por lo tanto útil para los diseñadores del currículo. (Garg & Varma, 2007).

13. (2007) Universidad de Los Andes – Colombia

La propuesta del modelo pedagógico de Cupi2 está fundada en cuatro estrategias exitosas aplicadas en ingeniería y otras áreas de la ciencia. El primer punto del modelo es el aprendizaje activo. Puesto que se trata de cursos donde la generación y desarrollo de habilidades es fundamental, el estudiante debe asumir un papel central en el proceso: debe

ser el protagonista principal del curso. Como segundo punto se tiene el Aprendizaje Basado en Problemas: se espera que los estudiantes enfrenten dificultades derivadas de la baja motivación, a través del aprendizaje basado en problemas que reflejan retos del mundo real. A continuación se tiene el Aprendizaje Incremental. Cada uno de los tres cursos del ciclo de programación se encuentra dividido en niveles en donde se introducen conceptos nuevos, se refuerzan conceptos vistos en niveles anteriores y se aplican todas las habilidades generadas hasta el momento en la solución de uno o varios problemas. El último componente es el modelo de Aprendizaje Basado en Ejemplos. De acuerdo con este modelo los estudiantes tienen acceso a ejemplos de buenas prácticas y soluciones comunes a problemas. (Villalobos & Vela, 2007).

14. (2007) Universidad Atilim– Turquía

El objetivo principal de esta investigación está alrededor de cursos nucleares y de fundamentación y resume los problemas actuales de la educación de la ingeniería de software. También propone algunas dimensiones importantes, como partes integrantes de la educación de ingeniería de software: los conocimientos interdisciplinarios, experiencia práctica, la comunicación, habilidades sobre educación continua y el profesionalismo. El estudio también presenta un estudio de algunas universidades importantes utilizando los programas de pregrado de ingeniería de software para evaluar estas dimensiones. (Mishra, Cagiltay, & Kilic, 2007).

15. (2007) University of Sheffield, Reino Unido

En esta investigación se describe cómo un enfoque gradual para el desarrollo de habilidades de los estudiantes para diseñar sistemas de software se aplica a la cuestión específica de la enseñanza del diseño de software. Evalúa los cursos relacionados con el diseño de software en el volumen de ingeniería de software de computación dentro del Plan de Estudios del 2001 con un modelo teórico que se ha desarrollado a partir de un programa bien establecido en la ingeniería de software. Esta evaluación identifica algunos temas que sería mejor enseñar antes, y otros que se aplazara hasta más adelante en el plan de estudios. (Cowling, 2007).

16. (2006) Universidad Atlantic Florida, FL - USA, Universidad de Sannio, Italia

Esta investigación resume la experiencia en la enseñanza de cursos de ingeniería de software en dos universidades diferentes utilizando un enfoque orientado a la práctica que guía a los estudiantes a través del aprendizaje de los diferentes conceptos tales como los aspectos abstractos y los del proceso de ingeniería de software. (Huang & Distante, 2006).

17. (2005) Universidad Vrije – Holanda

En esta investigación se informa sobre las experiencias con dos cursos de máster en arquitectura de software que se centran en ciertos aspectos de la comunicación. Esto

demuestra que, al centrarse adecuadamente el contenido de este curso, los aspectos fundamentales del campo industrial de gran relevancia dentro de la ingeniería de software se pueden enseñar con éxito en un entorno universitario. (Lago & Van Vliet, 2006).

18. (2005) Universidad Nacional de Singapore – Singapur

Para llenar el vacío entre el conocimiento teórico y experimental, se introdujo un curso de proyecto de equipo basado en el diseño y ejecución de las fases del ciclo de vida de desarrollo de software. Esta investigación enseña los principios de diseño y trabajo en equipo de forma basado en problemas, a través de conceptos arquitectónicos y el proceso de desarrollo iterativo. Los estudiantes deben cumplir con acumulación de productos demostrando requisitos de calidad en términos de fiabilidad, la reutilización y la documentación. La investigación concluye que este tipo de curso es esencial en los programas de ingeniería, ya que permite a los estudiantes asimilar mejor los conocimientos adquiridos en otros cursos de ingeniería de software. Dicho curso también juega un papel importante en una mejor preparación de los estudiantes para el trabajo. Se describe un método de enseñanza, la infraestructura y la experiencia adquirida durante más de tres años de enseñanza de los cursos de ingeniería de software en la Universidad de Singapur. (Jarzabek & Eng, 2005).

19. (2004) Universidad Politécnica Southern, GA – USA

Esta investigación presenta un modelo para el curso de arquitectura y diseño de software del programa de pregrado de ingeniería de software y forma parte del proceso de preparación según los lineamientos de la acreditación ABET. Esta investigación indaga sobre los enfoques pedagógicos para el desarrollo exitoso de este curso. Inicialmente se describe una estructura del curso, sus unidades de conocimiento, las cuales incluyen principios de diseño, estrategias, diseño arquitectónico, diseño detallado, diseño orientado a componentes, y el uso de herramientas de soporte al diseño y a la evaluación. (Karam, Qian, & Diaz-Herrera, 2004).

20. (2003) Universidad del Estado de Arizona, AR – USA

El curso de diseño de software debería pensarse como la forma en que los estudiantes obtienes el interés en el aprendizaje de la programación real con enfoque de objetos. Esta investigación recoge las experiencias de la enseñanza del curso de programación orientada a objetos y estructuras de datos en pregrado de la Universidad Estatal de Arizona, se describe además el objetivo principal del curso, el cual es proveer un entendimiento en profundidad del diseño orientado a objetos y las estructuras de datos motivando a los estudiantes en las labores de programación de computadores. (Ahamed, 2003).

21. (2003) Universidad Tecnológica de Tennessee, TN – USA

En esta investigación, se discuten los resultados preliminares observados a partir de un

curso de análisis y diseño de software ofrecido en los siguientes paradigmas: clase tradicional, clase en línea y clase en línea argumentada. En la clase analizada hubo un gran componente escrito el cual envuelve un proyecto en equipo de hasta 3 estudiantes. Esta clase requiere que el estudiante sea técnicamente eficiente en las bases de los conceptos de computación y las teorías, además que los estudiantes tengan la habilidad de abstraer y resolver creativamente los problemas plateados y que tengan la habilidad de interactuar positivamente con otros miembros del equipo de trabajo asumiendo responsabilidades. En esta investigación, a través de encuestas se obtuvo que los estudiantes valoran bastante la interacción social con el otro afirmando que es un aspecto crucial en el trabajo en el mundo real. Al entender los factores que impactan en el aprendizaje se buscan paradigmas de enseñanza más efectivos. (Brignall & Ramaswamy, 2003).

22. (2002) Universidad de Tecnología de Vienna – Austria

En esta investigación se describe el concepto de un curso de pregrado de un año en ingeniería de software basado en el Proceso Unificado de forma detallada y elaborada. Preguntas como: por qué, dónde y cómo, son factibles de ser respondidas siguiendo fielmente el proceso unificado en la incorporación sistemática del curso. La respuesta de los estudiantes en el proceso fue muy positiva. Se argumenta que el Proceso Unificado (UP) califica muy bien para que se enseñe a los estudiantes de ingeniería de software ya que integra y extiende las prácticas esenciales de dicha área del conocimiento en las ciencias de la computación. Otro argumento clave para la aplicación de la UP fue la de integrar los diferentes aspectos y temas tratados en el curso de ingeniería de software y su incorporación a un proceso de ingeniería de software estructurada y sistemática. De esta manera los estudiantes desarrollan una mejor comprensión de las relaciones e interacciones entre los diferentes modelos, medidas y técnicas de análisis. (Halling, Zuser, Köhle, & Biffl, 2002).

23. (1999) Instituto de Tecnología Rochester, NY – USA

Esta investigación describe la estructura de un curso de introducción a las arquitecturas de software y narra la experiencia en la enseñanza del mismo. También se habla sobre cómo podría evolucionar el curso con el tiempo de acuerdo con la evolución del software mismo y los paradigmas de desarrollo de software. (Naveda, 1999).

24. (1996) Universidad Murdoch– Australia

En esta investigación se estudia la elaboración del plan de estudios para obtener un título profesional en ingeniería de software. Esta propuesta se basa en la recomendación de que un programa de tres años no es lo suficiente para formar profesionales con fortaleza y la amplitud de la experiencia educativa para satisfacer las necesidades de una amplia gama de sectores industriales relacionados con la producción de productos software complejo y de calidad. Es la hipótesis de la investigación de una nueva estrategia de enseñanza de la ingeniería de software que puede ofrecer una oportunidad para formar profesionales con

una mejor apreciación de las demandas y requerimientos de la industria del software. (Roy & Veraart, 1996).

25. (1993) Universidad de Massachusetts, MA – USA

Un enfoque alternativo es enseñar a los estudiantes el diseño de software antes de enseñarles un lenguaje de programación. En este enfoque los estudiantes en realidad no producirían un software completamente terminado, porque no aprenden a programar en forma inicial. Más bien, su tarea es la de aprender a pensar en lo que se debe incluir en un programa en particular y la forma de organizar las piezas en un todo estructurado que tiene las características deseables en un buen software concebido desde su diseño, la no redundancia, la modificabilidad, la reusabilidad, y el encapsulamiento forman parte esencial de los criterios de diseño. Esta investigación es muy interesante en el sentido de cambiar el enfoque instruccionista de desarrollo de software iniciando con el diseño en lugar de la programación. (Sins-Knight & Upchurch, 2001).

Después de estudiar estas investigaciones con sus resultados, se determinan los siguientes elementos que forman un común denominador en casi todos estos antecedentes. Dichos aspectos comunes están asociados al campo educativo y de la enseñanza. Todas las investigaciones que soportan este estado del arte tienen:

1. La idea común de primero enseñar a diseñar en lugar de programar.
2. La idea común de organizar los contenidos del curso según los lineamientos de ACM.
3. La idea común de trabajar en equipo para diseñar software o para enfrentarse a analizar problemas que requieran una solución en software.
4. La idea común de asumir problemas de la vida real para resolverlos mediante ingeniería de software.
5. La idea común de presentar los cursos relacionados con el diseño de software como cursos teórico-prácticos, de hecho, todas las investigaciones hacen hincapié en que el componente práctico del curso es supremamente importante para los fines educativos.
6. Además, algunas de las investigaciones anteriores hablan sobre enfoque o modelo pedagógico a seguir, unas solamente mencionan el enfoque problémico, sin embargo, ninguna acentúa o profundiza en una reflexión netamente didáctica donde se aborden temas sobre el estudio formal de la enseñanza o del aprendizaje. Esta situación lleva a pensar que la construcción de una Didáctica del Diseño de Software es muy pertinente para la educación en ingeniería de software.

1.2 Marco teórico – conceptual.

1.2.1 Referentes sobre diseño de software.

La referencia internacional por excelencia en materia de formación en programas de pregrado en Ingeniería de Sistemas (Informática y ciencias de la computación) está en los

documentos producidos por la Joint ACM/IEEE-CS/AIS Task Force on Computing Curricula. La credibilidad y seriedad reconocidas en las dos principales asociaciones profesionales de la disciplina, la ACM (Association for Computing Machinery), la IEEE (Institute of Electrical and Electronics Engineers) y la AIS (Association for Information Systems), así como la calidad de la documentación producida hacen de ésta, la fuente obligada de referencias para definir lo que es importante en un currículo de informática, sistemas y computación y cómo se puede estructurar. Por otro lado, la IEEE - Computer Society ha desarrollado el cuerpo de conocimientos sobre la Ingeniería de Software, documentos que se relacionan más con el campo del análisis, diseño, implementación y despliegue de Software, escritos que constituyen una fuente importante en materia de diseño. A continuación se mencionan de manera detallada los aspectos de cada documento que apoyarán la presente investigación:

- ***Computing Curricula 2005. An Overview Report.***

Los autores de este documento formaron un comité conjunto en donde participan ACM, IEEE y AIS que inició actividades en 1988. Dicho comité continúa una tarea comenzada por ACM en 1965, quien estableció regularmente recomendaciones curriculares actualizadas sobre la disciplina. Desde 1991 ACM/IEEE producen reportes conjuntos que son ampliamente acogidos por la comunidad internacional (ACM, IEEE Computer Society and AIS, 2005).

El Report of the Joint ACM/IEEE-CS Task Force on Computing Curricula 2005 reconoce cinco perfiles de disciplinas: Ingeniería en Computación, Ciencia en Computación, Sistemas de Información, Tecnología en Computación e Ingeniería de Software.

Este documento se basa en cinco principios que orientan la formación de los currículos: a) El dramático crecimiento en el número de disciplinas en computación y sus impactos colectivos en la sociedad requiere que las disciplinas articulen una identidad compartida. b) Cada disciplina en computación debe articular su propia identidad y reconocer las de las otras disciplinas respetando su campo de acción. c) El documento deberá ser direccionado a una amplia audiencia, esto es, el documento será de utilidad a todas aquellas personas que estén involucradas en el ámbito académico de la computación. d) La caracterización de las disciplinas en computación será derivada del exhaustivo estudio del cuerpo de conocimiento de cada una de ellas. e) El documento pretende ser únicamente una guía y no un estándar riguroso.

Al tener en cuenta la amplitud del conocimiento que se maneja, la identificación de competencias fundamentales que deberán ser desarrolladas en los estudiantes suele variar entre los diferentes programas. A pesar de las diferencias curriculares, los manifiestos internacionales se constituyen como guía para la elaboración de currículos pertinentes al contexto regional, nacional e internacional. Sin embargo, son supremamente comunes las áreas que se describen a continuación y que representan el estado de la educación en computación, dichas áreas son: Estructuras discretas, programación, algoritmos y complejidad, arquitectura y organización, sistemas operativos, computación centrada

en red, teoría de lenguajes, interacción humano-computador, gráficas y computación visual, sistemas inteligentes, administración de la información, ingeniería de software, ciencia computacional y actividades sociales y profesionales (ACM, IEEE Computer Society and AIS, 2005).

Es entonces que, la ingeniería de software, constituye tan solo un área de conocimiento que se desarrolla en la computación como ciencia; y el diseño de software es una parte de la ingeniería de software. Para el documento, la Ingeniería de software es la disciplina orientada al desarrollo y al mantenimiento de sistemas de software con criterios de eficiencia, confiabilidad y calidad. Es por eso que hace necesario explorar el cuerpo de conocimiento del área de ingeniería de software puesto que constituye el escenario de acción del diseño de software.

- ***Visión de la ingeniería de software y del diseño de software según Computing Curricula 2005.***

Según The Overview Report (ACM, IEEE Computer Society and AIS, 2005), la ingeniería de software es la disciplina que se ocupa de la aplicación de la teoría, el conocimiento y la práctica para la construcción de sistemas de software eficientes y con criterios de calidad, que satisfagan los requerimientos de los usuarios y clientes en general. La ingeniería de software es aplicable a sistemas de pequeña, mediana y gran escala. Ella involucra las fases del ciclo de vida de los sistemas de software y tal ciclo contempla: El análisis de requerimientos y especificación, el diseño, la construcción, las pruebas, el despliegue y el mantenimiento. (IEEE-CS, 2004)

Con esto se tiene que el diseño de software forma parte importante del área de la ingeniería de software; entonces, la visión particular que tiene Computing Curricula 2005 sobre el diseño de software es la siguiente:

Diseño de software.

Temas a Tratar:
1. Conceptos y principios fundamentales del diseño.
2. El rol y el uso de contratos.
3. Patrones de diseño.
4. Arquitectura de software.
5. Diseño estructurado.
6. Análisis y diseño orientado a objetos.
7. Diseño basado en componentes.
8. Cualidades del diseño.
9. Debilidad de acoplamiento, alta cohesión, ocultamiento de información y eficiencia.
10. Confiabilidad, manejabilidad, escalabilidad, usabilidad y desempeño.
11. Otros enfoques: diseño centrado en la estructura, orientado a aspectos, orientado a funciones, orientado al servicio, metodologías ágiles, etc.
12. Diseño por reutilización

13. Objetivos de aprendizaje:
14. Discutir las propiedades de un buen diseño de software incluyendo la naturaleza y el rol de la documentación asociada.
15. Evaluar la calidad de múltiples diseños de software basados en los principios y conceptos claves del diseño.
16. Seleccionar y aplicar patrones de diseño apropiados en la construcción de una aplicación de software.
17. Crear y especificar el diseño de software para un producto de software de tamaño mediano usando una especificación de requerimiento de software, una metodología de diseño de programas aceptada y una notación de diseño apropiada.
18. Evaluar un diseño de software al nivel de componentes.
19. Evaluar un diseño de software desde la perspectiva de la reutilización.

- ***SWEBOK (Guide to the Software Engineering – Body of Knowledge).***

El documento SWEBOK (Guide to the Software Engineering – Body of Knowledge) es un proyecto del Comité de Prácticas Profesionales de la IEEE – Computer Society publicado en el 2004. En esta guía se establecen las bases del cuerpo de conocimiento en el área de la ingeniería de software y se manifiesta su responsabilidad de promover el avance tanto en teoría como en la práctica en este campo (IEEE-CS, 2004). A pesar de millones de profesionales del software a nivel mundial y a la presencia ubicua del software en la sociedad, la ingeniería de software ha alcanzado solo recientemente el estatus de una disciplina de ingeniería legítima así como el estatus de una profesión reconocida.

La IEEE – Computer Society ha llegado a un consenso para determinar el cuerpo de conocimiento de la ingeniería de software a través de varios años. Por definición, la ingeniería de software según SWEBOK es la aplicación de un enfoque sistemático, disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento de software.

Los objetivos del proyecto SWEBOK fueron establecidos así: 1) Promover una vista consciente de la ingeniería de software a nivel mundial. 2) Clarificar el lugar y el conjunto de fronteras de la ingeniería de software con respecto a otras disciplinas. 3) Caracterizar los contenidos de la disciplina de la ingeniería de software. 4) Proveer una fundamentación para el desarrollo de currículos en el área. (IEEE-CS, 2004)

La guía SWEBOK establece sus áreas de conocimiento de esta manera: Requerimientos de software, diseño de software, construcción de software, pruebas de software, mantenimiento de software, administración de la configuración de software, administración de la ingeniería de software, procesos de ingeniería de software, herramientas y métodos de ingeniería de software y calidad de software.

- ***El área de diseño de software según SWEBOK.***

De acuerdo a la definición de IEEE-CS, el diseño es tanto el proceso de la definición de la

arquitectura, los componentes, las interfaces y otras características de un sistema, como el resultado de dicho proceso. El área de conocimiento del diseño de software está dividido en seis sub áreas: a) Los fundamentos del diseño de software. b) Las acciones claves en el diseño de software. c) La estructura y arquitectura del software. d) El análisis de calidad del diseño de software y su evaluación. e) Las estrategias y métodos del diseño de software (IEEE-CS, 2004).

Para SWEBOK, el cuerpo de conocimiento sobre el diseño de software de por si es bastante amplio, tal cuerpo puede ser expresado a través de la siguiente jerarquía de conceptos asociados:

Diseño de software.

Fundamentos de diseño de software.
1. Conceptos generales de diseño.
2. El contexto de diseño de software.
3. El proceso de diseño de software.
4. Las técnicas habilitadas.

Acciones claves en diseño de software.
1. Concurrencia.
2. Control y manejo de eventos.
3. Distribución de componentes.
4. Manejo de errores y excepciones, y tolerancia a fallas.
5. Interacción y presentación.
6. Persistencia de datos.

Estructura y arquitectura de software.
1. Estructuras de arquitectura y puntos de vista.
2. Estilos de arquitectura (patrones de macro arquitectura).
3. Patrones de diseño (patrones de micro arquitectura).
4. Familias de programas y marcos de trabajo.

Análisis de calidad del diseño de software y evaluación.
1. Atributos de calidad.
2. Análisis de calidad y técnicas de evaluación.
3. Métricas.

Notaciones en diseño de software.
1. Descripciones estructurales (vista estática).
2. Descripciones comportamentales (vista dinámica).

Estrategias y métodos de diseño de software
1. Estrategias generales.
2. Diseño orientado a funciones.
3. Diseño orientado a objetos.
4. Diseño centrado en la estructura de datos.
5. Diseño basado en componentes.
6. Otros métodos.

1.2.2 Referentes sobre didáctica.

Otro de los elementos o pilares fundamentales de la presente investigación es la didáctica, por lo tanto, se pretende fundamentar este aparte analizando sus concepciones, características, componentes y enfoques que sirven como lineamientos esenciales en la construcción de la Didáctica del Diseño de Software; en este orden de ideas, a continuación se identifican y caracterizan las teorías relacionadas a este aparte.

- ***Consideraciones en torno a la didáctica.***

La palabra didáctica proviene en primera instancia del griego *didaskein didaktiké, didaskalia, didaktikos, didasko,* todos estos términos tienen en común su relación con el verbo enseñar, instruir y exponer con claridad; en segundo lugar proviene del griego teckne, que significa arte; dando lugar a los verbos en latín *docere* y *discere,* enseñar y aprender respectivamente. El concepto didáctica fue empleado por primera vez con el sentido de enseñar en 1629 por Ratke, en su libro Aphorisma Didactici Precipui (Principales Monismos Didácticos*). S*in embargo, fue consagrado por Juan Amós Comenio en su obra *Didáctica Magna,* publicada en 1657. Luego esta palabra cayó en desuso, hasta que en el siglo XIX Herbart y sus discípulos la resucitaron, limitando su contenido al conjunto de los medios educativos e instructivos. Hoy el término Didáctica está completamente extendido en todo el ámbito mundial.

- ***Una aproximación al concepto de didáctica.***

Un criterio valido para acercarse a la definición del concepto de didáctica, es revisar los que muchos autores, desde Comenio hasta los actuales han hecho en este campo. A continuación se indican en orden cronológico algunos autores que aportaron desde sus perspectivas a este concepto.

1. Comenio (1640), en su didáctica Magna, la define como “El artificio universal para enseñar todo a todos los hombres, con rapidez, alegría y eficacia”.

2. Otto William: “Es la teoría de la adquisición de lo que posee un valor formativo, es decir, la teoría de la formación humana”.

3. Mattos (1963): “La disciplina pedagógica, de carácter práctico y normativo, que tiene por objeto especifico la técnica de la enseñanza, la técnica de dirigir y orientar eficazmente a los estudiantes en su aprendizaje”.

4. Stócker (1964): “Es la teoría de la instrucción y la enseñanza escolar de toda índole y a todos los niveles”.

5. Tomascheusky (1966): “La teoría general de la enseñanza que investiga una disciplina particular de la pedagogía, las leyes del proceso unitario de la instrucción y la educación en la clase”.

6. Larroyo (1970): “El estudio de los métodos y procedimientos en las tareas de la enseñanza y del aprendizaje”.

7. Nerici (1973): “Es la ciencia y el arte de enseñar. Esto es, el conjunto de técnicas a través de las cuales se realiza la enseñanza mediante principios y procedimientos aplicables a todas las disciplinas, para que el aprendizaje de las mismas se lleve a cabo con mayor eficacia”.

8. García Hoz (1974): “El objeto inmediato de la didáctica es el trabajo discente y el docente, es decir, la enseñanza con una finalidad instructiva”.

9. Fernández Huertas (1974) “Es la ciencia que estudia el trabajo docente y discente congruente con los métodos de enseñanza y aprendizaje y que tienen como finalidad la instrucción”.

10. Mello de Carvalho (1974): “La didáctica tiene un sentido de planificación. Considera la acción docente en dos planos: el primero incluye los fundamentos de la acción, como son las finalidades de la educación, los objetivos generales, los inmediatos; en el segundo plano están los aspectos principales de la acción docente en donde precisa, la planificación, la incentivación del aprendizaje orientación de la conducta, el aprendizaje y la evaluación de los resultados”.

11. Titone (1981): “Ciencia que tiene como objeto específico y formal la dirección del proceso de enseñanza hacia fines inmediatos y remotos, de eficacia instructiva y formativa”.

12. Pérez Gómez (1982): “La didáctica es la ciencia y la tecnología del sistema de comunicación intencional donde se desarrollan los procesos de enseñanza-aprendizaje en orden a optimizar, principalmente la formación intelectual”.

13. Darós (1982): “La didáctica es una ciencia y una tecnología (conjunto racionalmente organizado y manejado de técnicas) para facilitar ordenando, sistematizando, analizando los resultados, en sus contenidos y sus procesos, que en el aprendizaje ejerce el estudiante sobre un saber organizado”.

14. Stocker (1983): “Comprenderemos como doctrina general de la enseñanza (…) o didáctica (…) la teoría de instrucción y de la enseñanza escolar de toda índole y a todos los niveles. Trata de los principios, fenómenos, formas, conceptos y leyes de toda enseñanza sin reparar en ninguna asignatura en especial”.

15. Fernández (1984): “La didáctica es una ciencia de la educación teórico-normativa que busca la adquisición de hábitos intelectuales mediante la integración del aprendizaje de los bienes culturales”.

16. Bruera (1985): “Es un saber que se articula desde las estructuras epistemológicas de las ciencias, pero que también funcionaliza adecuadas propuestas instrumentales a las imprevisibles variantes de los procesos de aprendizaje que se producen en la situación de clase”.

17. Gimeno Sacristán (1985): “La didáctica como disciplina científica a la que corresponde guiar la enseñanza, tiene un componente normativo importante que, en forma de saber tecnológico, pretende formular las recomendaciones para guiar la acción, es prescriptiva en orden a esa acción”.

18. Benedito (1987): “La didáctica es (está en camino de ser) una ciencia y una tecnología que se construye desde la teoría y la práctica, en ambientes organizados de la relación y comunicación intencional, donde se desarrollan procesos de enseñanza y aprendizaje para la formación del estudiante”.

19. Aebli (1988): “Es una ciencia auxiliar de la pedagogía en la que ésta delega, para su realización en detalle, tareas educativas más generales (…). La didáctica científica tiene por finalidad deducir el conocimiento psicológico de los procesos de formación intelectual, las técnicas metodológicas más aptas para producir procesos formativos”.

20. Rosales (1988): “La didáctica es la ciencia del proceso de enseñanza sistemática en cuanto a optimizadora del aprendizaje”.

21. Zabalza (1990): ”La didáctica actual constituye ese campo de conocimientos, de investigaciones, de propuestas teóricas y prácticas que se centran especialmente en los procesos de enseñanza-aprendizaje”

22. Vasco (1990): “Consideró a la didáctica no como la práctica misma del enseñar, sino como el sector más o menos bien limitado del saber pedagógico que se ocupa explícitamente de la enseñanza”.

23. Alicia W de Camilloni, (2004): “Nuestra disciplina es una teoría de la enseñanza, heredera y deudora de muchas otras disciplinas, su destino, empero, al ocuparse de la acción pedagógica, es constituirse, recíprocamente en oferentes y dadora de teoría en el campo social y del conocimiento”.

24. Ángel Díaz Barriga (1998):“La didáctica es una disciplina: Teórica, histórica y política. Es teórica en cuanto responde a concepciones amplias de la educación (y esto la engarzaría a una teoría de la educación) de la sociedad y, del sujeto, etc. Es histórica en cuanto sus propuestas son resultados de momentos históricos específicos…Es política porque su propuesta se engarza a un proyecto social. Se reclama desarrollar no solo otra perspectiva de conocimiento didáctico, sino estudiar la conformación de este campo desde diversos saberes sociales”.

25. H. Damaris Díaz (1998): “Ciencia imprescindible para orientar la praxis del

enseñante, y que, además, cualifica su actuación".

26. Edith Litwin (2004): "Entendemos la didáctica como teoría acerca de las prácticas de la enseñanza significadas en los contextos socio-históricos en que se inscriben".

Por otro lado, con el fin de identificar concretamente el concepto de didáctica, se encuentran algunas investigaciones (Estebaranz Garcia, 1994), (Oscar Sáenz , 1992) y (Ruiz Ruiz, 1996), que presentan un completo análisis de las definiciones de muchos autores con el fin de hallar los elementos comunes a todas ellas.

En forma general como se observa en la tabla 5, se obtiene en resumen los elementos en la definición de la didáctica.

Tabla 5 *Elementos presentes en las definiciones de didáctica*

Aspectos	Descriptores en la definición de didáctica
Carácter	disciplina subordinada a la Pedagogía, teoría, práctica, ciencia, arte, tecnología
Objeto	proceso de enseñanza-aprendizaje, enseñanza, aprendizaje, instrucción formación
Contenidos	Normativa, comunicación, alumnado, profesorado, metodología
Finalidad	formación intelectual, optimización del aprendizaje, integración de la cultura, desarrollo personal

Fuente: Tabla tomada de (Mallart, 2000).

Tras el rastreo relacionado con la definición que muchos autores hicieron sobre el concepto de didáctica, esta investigación retoma elementos valiosos desarrollados como resultado de acontecimientos históricos, políticos y culturales, tal como lo indica la figura 2:

Figura 2 *Un acercamiento al concepto de didáctica para DDS.*
Fuente: Esta investigación

Los autores de la presente investigación *no pretenden dar una definición puntual de didáctica* pero si indicar posturas que incidieron en la construcción de esta propuesta.

- ***Finalidad de la didáctica.***

Identificar la finalidad propia de la didáctica es uno de los lineamientos fundamentales de la presente investigación, puesto que orienta visión global en la construcción de la Didáctica del Diseño de Software. En la obra Didáctica: Concepto, objeto y finalidades (Mallart, 2000) se presenta un exhaustivo análisis de este apartado, indicando en forma general que la Didáctica presenta una doble finalidad, tal como han puesto de manifiesto otras investigaciones (Zabalza, 1990) y (Bolívar, 1995). La primera finalidad, como ciencia descriptivo-explicativa, representa una dimensión teórica. La segunda, como ciencia normativa, es su aspecto práctico aplicado y consiste en la elaboración de propuestas para la acción. La teoría y la práctica se necesitan mutuamente en el caso de la Didáctica. Sería inimaginable la una sin la otra.

Finalidad teórica. Como todas las ciencias, trata de adquirir y aumentar el conocimiento cierto, aquello que sabemos sobre su objeto de estudio, que es el proceso de enseñanza-aprendizaje. Trata de describirlo mejor y de explicarlo e incluso interpretarlo. Para la descripción, es preciso acercarse sin prejuicios al objeto de estudio, mezclarse con él, verlo de cerca y obtener sobre el mismo diferentes puntos de vista. Para la interpretación, sin embargo, también habrá que distanciarse, reflexionar sobre las causas de los hechos y tratar de establecer, cuando se pueda, generalizaciones, aunque esto último no es nada fácil. Decir cómo se constituye y cómo debería ser analizado el proceso didáctico no es lo mismo que decir qué finalidades tiene la enseñanza. Tampoco equivale a decir de qué manera debería llevarse a término el proceso de enseñanza-aprendizaje. La descripción ni proporciona finalidades educativas, ni sugiere qué métodos deben usarse. Ésta será la finalidad práctica.

Finalidad práctica. Se trata regular, dirigir en la práctica el proceso de enseñanza-aprendizaje, elaborar propuestas de acción, intervenir para transformar la realidad. Es una dimensión eminentemente práctica y normativa. Lo cual no quiere decir que las decisiones no deban estar fundamentadas en criterios científicos, sino más bien que las propuestas para la acción deben ser realistas y adaptadas a cada situación determinada, posiblemente irrepetible. Se trata de intervenir para dirigir procesos, mejorar condiciones de aprendizaje, solucionar problemas, obtener la formación, la instrucción formativa en la línea de conseguir la educación global, el desarrollo de facultades como finalidad práctica, en definitiva, se trata de intervenir en el proceso para provocar en el alumnado su formación intelectual.

- ***Elementos conceptuales de un acto didáctico.***

Para obtener los elementos principales presentes en un acto didáctico, es interesante observar la Figura 3 Mialaret (1984), en el que recoge los principales problemas que debe resolver la Didáctica.

Figura 3 *Problemas generales de la didáctica (Mialaret, 1984, 71).*
Fuente: Tomado de la obra Didáctica: Concepto, Objeto Y Finalidades (Mallart, 2000)

En forma general se puede asegurar que todo proceso de enseñanza – aprendizaje está regido por las seis categorías de la didáctica: objetivos de la educación (¿qué se quiere con la enseñanza?), contenido (¿qué conocimientos enseñar que permitan el cumplimiento de los objetivos? y ¿qué debe aprender a hacer el estudiante?), métodos y procedimientos de enseñanza (¿cómo regular las actividades del profesor y el estudiante para el logro de los objetivos propuestos?), medios de enseñanza (¿con qué enseñar y aprender?), formas de organización (¿cómo organizar el enseñar y el aprender?) y evaluación (¿en qué medida se cumplen los objetivos?). Todas estas categorías conforman un sistema en el cual al modificar una de estas es necesario el reanálisis y la redefinición del resto, que garanticen la continuidad del funcionamiento sistémico.

- ***La didáctica general.***

La parte fundamental y global es la Didáctica General, ya que se ocupa de los principios generales y normas para dirigir el proceso de enseñanza-aprendizaje hacia los objetivos educativos. Estudia los elementos comunes a la enseñanza en cualquier situación ofreciendo una visión de conjunto. También ofrece modelos descriptivos, explicativos e interpretativos generales aplicables a la enseñanza de cualquier materia y en cualquiera de las etapas o de los ámbitos educativos. Aunque debe partir de realidades concretas, su función no es la aplicación inmediata a la enseñanza de una asignatura o a una edad determinada. Se preocupa más bien de analizar críticamente las grandes corrientes del pensamiento didáctico y las tendencias predominantes en la enseñanza contemporánea (Matos, 1963) (Díaz Barriga , 1997)**.**

- ***Didáctica especial o didácticas específicas.***

Trata de la aplicación de las normas didácticas generales al campo concreto de cada disciplina o materia de estudio. Hoy día se utiliza también la denominación de didácticas específicas, entendiendo que hay una para cada área distinta: Didáctica del lenguaje, de la matemática, de la física, de las ciencias sociales o naturales, de la expresión plástica, de la

educación física, etc. (Díaz Barriga , 1997)**.**

- ***La enseñanza en la ingeniería de software.***

En el II Simposio de formación del Ingeniero Informático y uso de las TICS en la educación superior (UCIENCIA, 2006) se clasifica en forma general los medios de enseñanza en el proceso de enseñanza – aprendizaje de la Ingeniería de software en tres tipos: los tradicionales (láminas, dibujos, transparencias, maquetas, esquemas, etc.), los generados a partir del desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) (láminas en Power Point, Internet, correo electrónico, etc.) y las propias herramientas software de la asignatura (herramientas CASE, herramientas para la planificación, estimación y seguimiento de proyectos, entre otros). Se presenta también lineamentos generares asociados a estándares internacionales sobre educación y construcción de software (ACM, IEEE Computer Society and AIS, 2005 y ACM, IEEE-CS, AIS, 2005), muy importantes para la presente investigación.

Tabla 6 *Modelos pedagógicos*.

	Tradicional	**Romántico**	**Conductista**	**Estructuralista o progresista**	**Socialista**
Intención	Formar el carácter	Desarrollar el interior del estudiante	Instruir y reforzar	Orientar de acuerdo con la necesidad y condición de cada uno	Producir hombres nuevos para la sociedad
Método	Imitación, ejemplo, repetición	Suprimir interferencias. Conducir la transformación de etapas. Dejar hacer, dejar pasar.	Fijar, reforzar, asociar	Crear ambientes de acuerdo con etapas de desarrollo. Parte de experiencias del alumno. Interacción dialéctica sujeto – objeto. Reflexión científica	Trabajo productivo. Análisis crítico de la sociedad capitalista.
Metas	Formar en el humanismo, religiosidad y buen ejemplo	Desarrollo natural en cuanto a salud mental como libertad, felicidad y principios éticos e intelectuales	Modelar conducta. Desarrollar la conformidad social. Tecnología educativa. Modificar el comportamien to	Estimular el desarrollo humano. Propiciar el acceso a niveles intelectuales superiores biopsicosociales. Posibilitar el desarrollo intelectual como proceso continuo en espiral. Configuración del conocimiento. Estructuras del pensamiento Transforma al hombre y la naturaleza. Fomentar la formación consciente y crítica del valor.	Transformar al hombre y la naturaleza. Fomentar la formación consciente y crítica del valor. Luchar por la liberación del hombre.

	Tradicional	Romántico	Conductista	Estructuralista o progresista	Socialista
Concepto del desarrollo del niño	Desarrollo de las facultades y el carácter a través de la disciplina	El conocimiento es el resultado de la intuición. Aproximación empírica a la realidad.	Se aprende por acumulación de contenidos. El conocimiento es repetitivo y objetivo	La madurez consiste en la reorganización de las estructuras psicológicas resultantes de la interacción organismo – medio ambiente. El conocimiento se da desde la reflexión científica	Desarrollo progresivo secuencial. El mundo es materia: es primario. La conciencia y el pensamiento es lo secundario. El quehacer educativo está íntimamente ligado a la militancia política
Contenidos	La disciplina, autores y clásicos	Epistemología existencialista o fenomenológica. Enfatiza en estados, sentimientos internos. Exigencia y solicitud del alumno	Competencias observacionales. Habilidades, destrezas, aspectos técnicos. Conocimientos como producto de algo	Respeto por la libertad. Razonamiento lógico. Operaciones formales. Elaboración de una teoría ética y epistemológica formal de la verdad y el valor	Naturaleza y vida social
Relación	Vertical, autoritaria	Docente auxiliar	Intermediario, ejecutor, programador	Docente, guía, orientador bidireccional	Directa, horizontal
Evaluación	Conductas observadas, esperadas. Formativa, sumativa, resultados	No hay confrontación, ni calificación.	Niveles altos de conocimiento. Diferencia individual.	Niveles altos de conocimiento	Grupal

Fuente: Tomado de (Restrepo Mesa, 2008)

1.2.3 Referentes sobre aprendizaje activo.

Hace ya unos 2.400 años el sabio Confucio en el oriente, declaró en una de sus más importantes frases, la necesidad del aprendizaje activo en la educación:

Lo que escucho, lo olvido.
Lo que veo, lo recuerdo.
Lo que hago, lo comprendo.

El psicólogo estadounidense Mel Silberman, pionero en el área de aprendizaje activo, la modificó para crear lo que él llama “El credo del aprendizaje activo”:

Lo que escucho, lo olvido.
Lo que escucho y veo, lo recuerdo poco.
Lo que escucho, veo y pregunto o converso con otra persona, comienzo a comprenderlo.
Lo que escucho, veo, converso y hago, me permite adquirir conocimiento y aptitudes.
Lo que enseño a otro, lo domino.

"El credo" lo argumenta en el hecho de que al escuchar al profesor es muy fácil olvidar debido a la diferencia de velocidad con que se habla y se escucha: se piensa mucho mientras se escucha. Si a la dimensión auditiva, se le incluye una visual, el mensaje se ve reforzado porque interactúan dos sentidos y el profesor tiene la oportunidad de satisfacer las necesidades de más tipos de estudiantes, pero aún no es suficiente para aprender. La mente humana permanentemente cuestiona la información entrante, es decir, la procesa, y para que esta labor sea efectiva es conveniente, además de auto reflexionar, exteriorizar y compartir los interrogantes con otras personas de este modo existen diferentes puntos de vista que se complementan para aprender todos. Y si además se tiene la oportunidad de hacer algo con la información recibida y compartida, se produce el beneficio de la retroalimentación sobre los aciertos o errores cometidos. Pero cuando se prueba la información, se la recapitula e inclusive se la explica a otras personas, el cerebro guarda de forma imborrable el aprendizaje (Silberman, 1998). En este sentido la Didáctica del Diseño de Software propuesta llega con su quehacer hasta que los estudiantes puedan adquirir conocimiento y aptitudes, es decir, al penúltimo planteamiento del credo de Silberman, pero deja abierta la posibilidad para avanzar hasta permitir a los estudiantes dominar el conocimiento.

La motivación para utilizar aprendizaje activo en la Didáctica del Diseño de Software es el aporte de Mel Silberman en cuanto a la formulación y respuesta a algunas preguntas sobre algunos mitos que se crean alrededor del credo. La tabla 7 resume los interrogantes mencionados:

Tabla 7 *Preguntas y respuestas sobre aprendizaje activo.*

Interrogante	Respuesta
¿El aprendizaje activo no es más que un puñado de "juegos y diversión"?	No. Y argumenta que esta forma de aprendizaje presenta desafíos inusuales que requieren mucho trabajo del estudiante
¿El aprendizaje se centra en la actividad y deja un poco de lado la reflexión de lo hecho?	No. Y argumenta que el valor de esta forma de aprendizaje es pensar y compartir con otros las actividades y su significado. Incluyendo en una exposición final sobre lo realizado se logra la reflexión.
¿El aprendizaje activo requiere mucho tiempo para abarcar el material del curso?	Sí. Y enfatiza la utilización de mucho más tiempo en este aprendizaje que en enfoques tradicionales, pero se puede evitar el desperdicio de minutos, moderando los contenidos de tal forma que se brinde oportunidad de reflexionar.
¿Los métodos de aprendizaje activo pueden otorgar interés a una información árida y poco interesante?	Sí. La sola emoción de los métodos de este aprendizaje puede atrapar a los estudiantes y desarrollar los temas áridos y tediosos.
¿En aprendizaje activo, el	Sí. Cuando no se tiene experiencia en la conformación y

Interrogante	Respuesta
trabajo en grupo puede ocasionar pérdida de tiempo y poca productividad?	estructuración de los grupos y las actividades a desarrollar en ellos. Los estudiantes pueden abordar superficialmente los temas; pero asignando roles, reglas y procedimientos se puede corregir muchos inconvenientes.
¿Se puede exagerar el uso de grupos a través del aprendizaje activo?	Sí. Pero se puede corregir atendiendo a la variedad. “Las diversas modalidades de aprendizaje constituyen el condimento de la buena docencia”.
¿Existe el peligro de compartir información errónea en el trabajo en grupo?	Sí. Pero el aspecto social del aprendizaje logra superar las desventajas ocasionadas
¿Los estudiantes están convencidos, como el docente, de las ventajas del aprendizaje activo?	Al principio no, porque la mayoría de estudiantes están acostumbrados a metodologías tradicionales, pero si se introduce gradualmente, al final se convencerán y les será atractivo.
¿Se necesita más preparación y creatividad para enseñar con los métodos de aprendizaje activo?	Sí y no. Al inicio se necesitará mayor preparación, pero una vez ganada la experiencia, el docente sentirá emoción y energía que transmitirá a sus estudiantes.

Fuente: Esta investigación

En un compendio de experiencias sobre la aplicación de aprendizaje activo se define una estrategia de aprendizaje activo como “aquella que propicia una actitud activa del estudiante en clase, en contraposición con lo que ocurre en el método expositivo clásico, en el que el alumno se limita a tomar notas de lo que ve en la pizarra (se dice que en una clase expositiva la información pasa directamente de las notas del profesor a la libreta del alumno, sin pasar por sus cabezas)”(Barrado, y otros, 2001). Y se presentan tres razones por las cuales se debería utilizar aprendizaje activo. La primera es porque los alumnos mantienen mejor el nivel de atención; la segunda porque facilita la adquisición de conocimientos; y la última porque facilita la obtención de feedback sobre el nivel de comprensión.

En el mismo trabajo se menciona el costo que produce aplicar aprendizaje activo en clase desde dos puntos de vista. El primero tiene que ver con el tiempo en términos de cubrir menos temario en los periodos planeados; y el segundo habla acerca del costo emocional tanto para el profesor como para el estudiante, aquí se pone de manifiesto que al principio los docentes se alegran mucho porque sus estudiantes son más activos en clase, pero ese ánimo se convierte en decepción al comprobar la baja comprensión de los temas. Efectivamente, esa retroalimentación que se produce cuando hay actividad (y que no se produce en la clase expositiva) pone de manifiesto algo que antes no se veía (pero que también pasaba): el hecho de que no todos los alumnos han comprendido. Este desanimo puede ser tan grande que empuje al profesor a dar un paso atrás y regresar a la “emocionalmente más estable” clase expositiva.

En el artículo denominado Active Learning Workshop (Bonwell) publicado en la ACM, se presentan las siguientes características más relevantes del aprendizaje activo:

1. Los estudiantes son más participativos en actividades de lectura, escritura y discusiones.
2. Existe un menor énfasis en la transmisión de información y más énfasis en el desarrollo de habilidades.
3. Se produce mayor desarrollo de actitudes y valores en el proceso de enseñanza – aprendizaje.
4. Conforme se desarrollan las clases y la temática, la motivación de los estudiantes aumenta.
5. La dinámica del aprendizaje activo hace que los estudiantes reciban retroalimentación inmediata por parte del docente.
6. Los estudiantes se ven involucrados en actividades que implican desarrollo de habilidades de pensamiento superior como análisis, síntesis y evaluación.
7. Los estudiantes tienen la oportunidad de hacer y reflexionan sobre lo hecho.
8. El aprendizaje activo como proceso continuo, permite desarrollar tanto tareas que van de lo sencillo a lo complejo, en donde las tareas simples son cortas y no estructuradas; como las complejas que son de larga duración y necesitan ser cuidadosamente planeadas y estructuradas.

El mismo autor justifica la importancia de la inclusión del aprendizaje activo en la educación, a través de los siguientes planteamientos:

1. La cantidad de información retenida por el estudiante se reduce notablemente después de 10 minutos.
2. Todo aprendizaje genuino es activo, es un proceso de descubrimiento en el cual el estudiante es el protagonista principal, no el maestro.
3. Los estudiantes aprenden lo que les importa y recuerdan lo que entienden.
4. El aprendizaje no es un deporte para espectadores, los estudiantes deben hablar y escribir sobre lo que están aprendiendo, relacionar con experiencias que les ha pasado y aplicar sobre su vida diaria, es decir, deben hacer parte de sí mismos lo que han aprendido.
5. Se debe aprender haciendo, esto implica constantemente mantener incertidumbre de algo hasta que se lo intenta hacer.

Además de las características y justificación del aprendizaje activo, Bonwell plantea los obstáculos y barreras para su utilización, y propone algunas alternativas para superarlos (ver tabla 8), posteriormente también analiza los riesgos que se pueden ocurrir al aplicar dicho aprendizaje, tanto para el estudiante como para el docente (ver tabla 9) y realiza una comparación del nivel de riesgo contra las dimensiones del aprendizaje activo (ver tabla 10).

Tabla 8 *Obstáculos o barreras para la utilización de aprendizaje activo vs solución.*

Obstáculo o barrera	Solución
Imposibilidad de cubrir el tema en el tiempo destinado para ello.	Encontrar otras estrategias para lograr cubrir los temas.
La elaboración de estrategias de aprendizaje activo demanda mucha preparación antes de clase.	Llegar a un equilibrio en la preparación de clases activas.
Grupos numerosos.	Dividir los grupos según las actividades, de tal forma que participen todos.
La mayoría de profesores se consideran buenos y no reflexionan sobre su quehacer	Reflexionar en torno a que la enseñanza no es igual al aprendizaje.
Carencia de material o equipos que soporten el aprendizaje activo.	Encontrar formas de superar la carencia de material y equipos con estrategias novedosas.
Preferencia de enfoques tradicionales por parte de los estudiantes	Con estrategias novedosas los estudiantes paulatinamente lograrán salir de los enfoques de aprendizaje tradicionales.
El riesgo implícito de utilizar estrategias de aprendizaje activo.	Seleccionar las estrategias de aprendizaje activo más cómodas para minimizar el nivel de riesgo.

Fuente: Esta investigación

Tabla 9 *Riesgos para el estudiante y docente en la utilización de aprendizaje activo.*

Actor	Riesgo
Estudiante.	Inexistencia de participación activa. Aprendizaje de contenidos insuficiente. Desarrollo mínimo de habilidades de pensamiento superior. Disfrute mínimo de la experiencia.
Profesor	No sentir control de la clase. No confiar en sí mismo. No poseer las habilidades necesarias para utilizar aprendizaje activo. Ser visto diferente en el profesorado por no enseñar de la forma ya establecida.

Fuente: Esta investigación

Tabla 10 *Comparación del riesgo de las estrategias de aprendizaje activo*.

Dimensión	Estrategias de bajo riesgo	Estrategias de alto riesgo
Tiempo de clase requerido.	Cortas.	Extenso.
Grado de planeación.	Planeadas cuidadosamente.	Espontáneas.
Grado de estructuración.	Más estructuradas.	Menos estructuradas.
Tema.	Concreto.	Abstracto.
Controversia.	Menor.	Mayor.
Conocimiento previo del tema por parte del estudiante.	Mejor informado.	Menos informado.
Conocimiento previo de la técnica de enseñanza por parte del estudiante.	Familiar.	Desconocido.
Experiencia del profesor en la técnica de	Importante.	Limitada.

Dimensión	Estrategias de bajo riesgo	Estrategias de alto riesgo
enseñanza.		
Patrón de interacción.	Entre el profesorado y estudiantes.	Entre estudiantes.

Fuente: Tomado de Active Learning Workshops (Bonwell).

Finalmente Bonwell enumera algunas estrategias de aprendizaje con su correspondiente nivel de riesgo que pueden ser utilizadas por los docentes (ver tabla 11) y presenta las siguientes recomendaciones para promover el aprendizaje de forma activa en los estudiantes: uso adecuado del humor, motivar el rendimiento de los estudiantes, involucrar a los estudiantes fuera de clase, incrementar el nivel de auto evaluación, motivar a los estudiantes a hablar, preguntar acerca de los puntos de vista y sensaciones de los estudiantes, seguir los temas que proponen los estudiantes aunque no estén relacionados con el tema y referirse al espacio de clase como propio, es decir, "nuestra" clase y lo que "nosotros" hicimos.

Tabla 11 *Estrategias de enseñanza por nivel de riesgo para el instructor.*

Actividad	Riesgo
Procedimiento con pausa	Bajo
Escritos cortos - Resúmenes, lecturas - Listas analíticas - Noticias de periódico	
Respuestas si/no a preguntas	
Encuestas y cuestionarios	
Quizzes	
Lluvia de ideas	
Compartir ideas en binas	
Desarrollo de la clase en binas y grupos	
Discusiones en grupo con preguntas específicas (estructurado)	
Discusiones en grupo (no estructurada)	Alto
Conferencia guiada	
Presentaciones individuales y en grupo	
Desarrollo de aplicaciones en binas y grupos relacionadas con la conferencia	
Desarrollo en binas y grupos de pruebas de preguntas relacionadas con el material de la conferencia	
Análisis de los estudiantes de un material específico	
Trabajo en un problema y evaluación del trabajo de otros	
Juego de roles para ilustrar un concepto de la conferencia	
Conferencia participativa	

Fuente: Tomado de Active Learning Workshops (Bonwell).

En la Didáctica del Diseño de Software se tiene en cuenta los riesgos que acarrea implementar aprendizaje activo tanto en el aula como fuera de ella, así estudiantes y profesores tendrán la oportunidad de lograr un equilibrio en el proceso de enseñanza – aprendizaje.

La decisión de incluir aprendizaje activo en la Didáctica del Diseño de Software se toma con base en varios estudios comparativos sobre la implementación del aprendizaje activo (Barrado, y otros, 2001) en las clases frente a una metodología tradicional, en donde se explican aspectos como cambios tanto en docentes como estudiantes, diferencias y resultados cuantitativos y cualitativos de las investigaciones. Por ejemplo, en la Universidad de Washington entre el otoño del 2008 y primavera del 2009 se realizó un estudio en donde se implementó el aprendizaje activo en los cursos de sistemas software y desarrollo de software (Ross Sowell), cuya característica principal es que contienen temas sencillos y el énfasis es la aplicación de conceptos. La información de análisis fue cuantitativa (calificaciones, récord de exámenes y registro de evaluaciones) y cualitativa (entrevistas). Se trabajó con grupo focal y de control.

La investigación arrojó los siguientes resultados en cuanto a cambios producidos: en los dos cursos la cátedra formal se eliminó en gran parte y fue remplazada por cátedra informal corta combinada con sesiones de aprendizaje activo; las cátedras tradicionales fueron grabadas para ser utilizadas como preparación previa de clase por parte de los estudiantes o para revisiones posteriores; el tiempo de clase se utilizó principalmente en la solución de problemas, tareas de programación en equipos y los docentes estuvieron disponibles para proporcionar ayuda y orientación en el trabajo. Las diferencias presentadas fueron la manera de conformar grupos, la forma de calificación y el tipo de trabajo asignado. En cuanto a desempeño el promedio y media de calificaciones aumentó, la desviación estándar disminuyó, la valoración general de los cursos y del docente en su gran mayoría aumentó y la satisfacción general tanto de docente como estudiante aumentó.

La entrevista se estructuró con tres aspectos, a saber: lecturas en línea, uso del tiempo de clase y uso del tiempo de laboratorio, en el primer aspecto, los estudiantes manifestaron carecer de tiempo de retroalimentación y por tanto su participación en clase es baja, catalogan como ventaja tener a disposición permanente el material pero que no tenga un índice con la clase grabada es para ellos un desventaja. La revisión del material apenas fue un proceso aleatorio de selección de temas y existieron problemas técnicos para acceder. En los otros dos elementos de la entrevista se manifestó que la nueva metodología necesita más tiempo para completar cada una de las actividades.

El estudio sirvió también para que los docentes reflexionen su quehacer, al respecto se encontró que el trabajo en grupo es bueno, el nivel de compromiso con el material de lectura fue alto, las discusiones del material fueron activas y eficaces, la rotación de roles en los ejercicios fue clave en el trabajo en grupos, aparecen soluciones alternativas a problemas planteados y el estudiante no tiene miedo de penalizaciones y puede proponer soluciones aunque no funcionen, cuando el trabajo no es calificado. Finalmente invita a mejorar la aplicación del aprendizaje activo con trabajos futuros que tengan que ver con el indexando las clases grabadas, inclusión de notas en línea de refuerzo, eliminar progresivamente la cátedra tradicional y mejorar la revisión de trabajo en binas.

1.3 Marco contextual

La investigación hace uso de la infraestructura y los recursos de la Universidad de Nariño en la ciudad de Pasto – Colombia. La recolección de información involucra las universidades localizadas en Australia, Bélgica, Canadá, China, Colombia, Dinamarca, Estados Unidos, Francia, Alemania, Holanda, India, Israel, Italia, Japón, Korea del Sur, Suecia, Suiza, Reino Unido, Taiwán y Colombia, según los Anexos 3 y 4.

2. ANÁLISIS DE RESULTADOS

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 4.

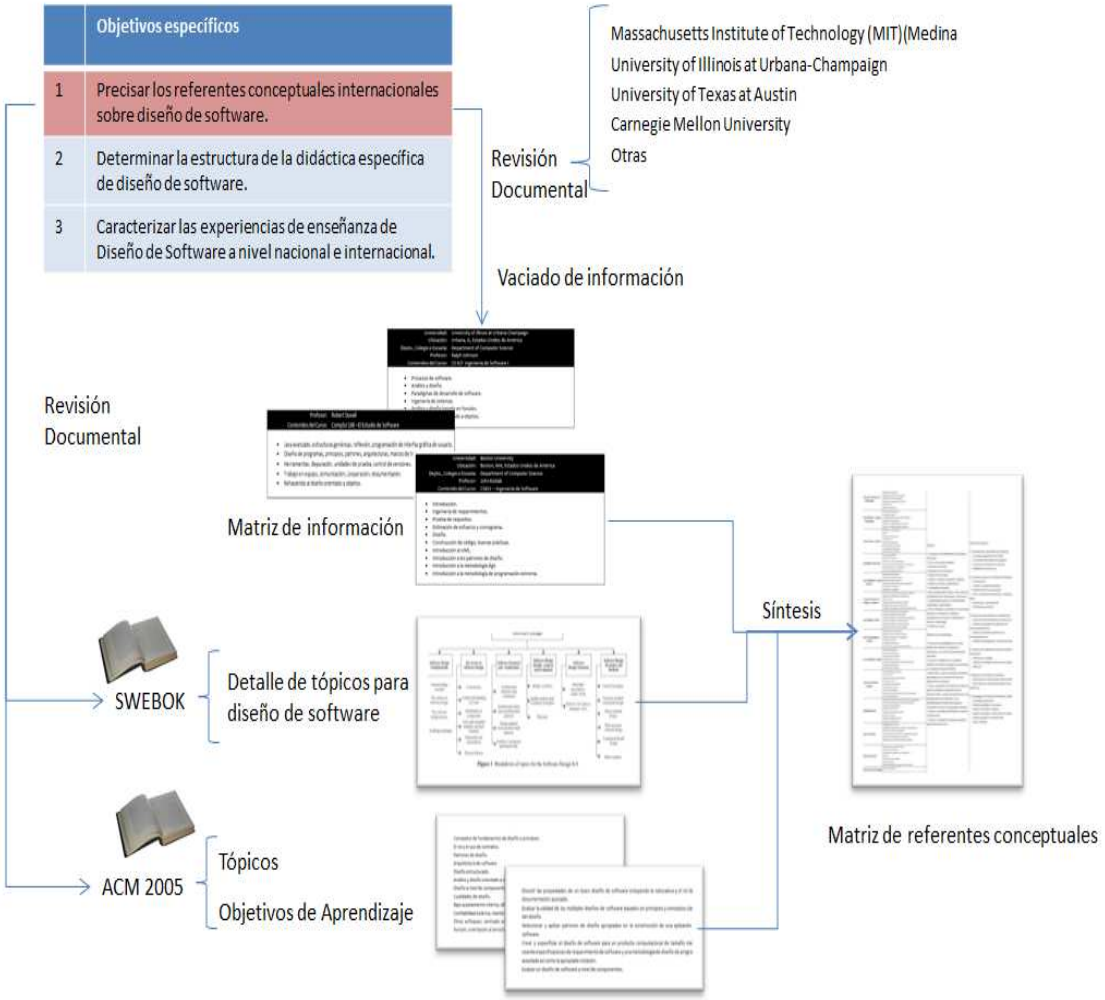


Figura 4 *Metodología para precisar los referentes conceptuales internacionales sobre diseño de software.*
Fuente: Esta investigación.

2.1 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, esta investigación 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 de la presente investigación 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 lo que se debería enseñar en un curso de diseño de software, se ha tomado como segunda referencia el texto titulado "SWEBOK: Guide to Software Engineering Body 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 Computer Society.

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 5.

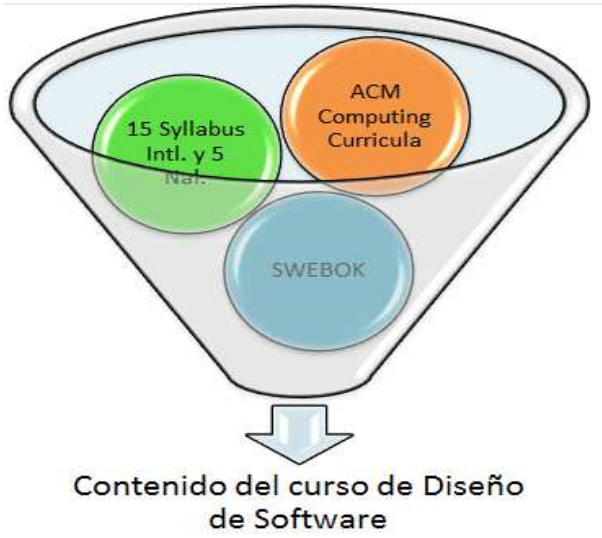


Figura 5 *Estructura orientadora de contenidos para la Didáctica del Diseño de Software.* Fuente: Esta investigación.

A continuación se analiza las perspectivas de cada insumo para finalmente realizar la propuesta de los contenidos del curso de diseño de software. La Didáctica del Diseño de Software utilizará dichos contenidos como cuerpo de conocimiento.

2.1.1 Contenidos según ACM Computing Curricula.

La ingeniería de software es la disciplina que se encarga de la aplicación de la teoría, el conocimiento y la práctica para la construcción eficiente y efectiva de sistemas software que satisfagan los requerimientos de los usuarios y los clientes. La ingeniería de software es aplicada a sistemas de pequeña, mediana y gran escala. Esta disciplina contempla todas las fases del ciclo de vida de un sistema software; el ciclo de vida incluye análisis de requerimientos y especificación, **diseño**, construcción, prueba y despliegue operacional y mantenimiento.

Partiendo del hecho que el objetivo de la presente investigación es la construcción de una Didáctica para el Diseño de Software, es el diseño en donde se enfocan los contenidos del curso. En este sentido, la recomendación de ACM son los siguientes aspectos:

Tópicos:

1. Conceptos de fundamentos de diseño y principios.
2. El rol y el uso de contratos.
3. Patrones de diseño.
4. Arquitectura de software.
5. Diseño estructurado.
6. Análisis y diseño orientado a objetos.
7. Diseño a nivel de componentes.
8. Cualidades de diseño.
9. Bajo acoplamiento interno, alta cohesión, ocultamiento de información y eficiencia.
10. Confiabilidad externa, mantenibilidad, usabilidad, y desempeño.
11. Otros enfoques: centrado en la estructura de datos, orientación a aspectos, orientación a la función, orientación al servicio, diseño ágil.
12. Diseño por reúso.

Objetivos de Aprendizaje:

1. Discutir las propiedades de un buen diseño de software incluyendo la naturaleza y el rol de la documentación asociada.
2. Evaluar la calidad de los múltiples diseños de software basados en principios y conceptos claves del diseño.
3. Seleccionar y aplicar patrones de diseño apropiados en la construcción de una aplicación de software.
4. Crear y especificar el diseño de software para un producto computacional de tamaño medio usando especificaciones de requerimiento de software y una metodología de diseño de programa aceptada así como la apropiada notación.
5. Evaluar un diseño de software a nivel de componentes.
6. Evaluar un diseño de software desde la perspectiva del reúso.

2.1.2 Contenidos según los 15 syllabus internacionales y 4 nacionales.

15 expertos profesores de diseño de software a nivel mundial han ensamblado sus propios syllabus de los cursos. A continuación se presenta los contenidos generales de cada uno de ellos:

Tabla 12 *Contenidos del curso de diseño de software, Institúto de Tecnología Massachusetts (MIT).*

Universidad:	Instituto Tecnológico de Massachusetts (MIT)
Ubicación:	Cambridge, MA, Estados Unidos de América
Depto., Colegio o Escuela:	Dept. de Ingeniería Electrónica y Ciencias de la Computación
Profesor:	Daniel Jackson
Contenidos del Curso:	6.005 Elementos de construcción de software

1. Sintaxis básica de Java y semántica, Preámbulo de objetivos y estructura del curso.
2. Más Java, excepciones, entrada y salida, clases, control de acceso, estática.
3. Subclases, herencia, sobrecarga, interfaces, paquetes, distinción entre tipos declarados y tipos actuales, conversión de tipos y clases anónimas.
4. Diseño de máquinas de estado, notación gráfica y textual, semántica de máquinas de estado, combinaciones paralelas de máquinas.
5. Patrones de implementación de máquinas de estado, concurrencia y colas, modularidad e interfaces.
6. Seguridad, propiedades de estado e invariantes, razonamiento inductivo, combinación paralela, explosión de estado, tolerancia a fallos, interbloqueos y la idea de una base de confianza.
7. Programa de procesamiento de flujos, gramáticas contra máquinas, métodos JSP de derivación de programa, gramática regular y expresiones.
8. Modularidad, desacoplamiento, ocultamiento de la información, diagramas de dependencia modular, usando interfaces para desacoplamiento.
9. Representando el comportamiento usando estructuras de datos. Tipos de datos de Lenguaje, visitantes, objetos funcionales, funciones de orden superior. Resolución de problemas a través de la creación de lenguajes de dominio especifico.
10. Semánticas de montículo, alcanzabilidad y almacenamiento conceptual, el contrato de objeto y las propiedades de igualdad, mapas hash y su representación de invariabilidad, problemas causados por la mutación de claves.
11. Fundamentos de programación de interfaces graficas de usuario. Vista jerárquica, patrones compuestos, controlador de vista y modelo. Seguimiento a la programación basada en eventos.
12. El paradigma relacional; el modelo conceptual; la sintaxis del modelo de objetos y semánticas.
13. La implementación como la transformación del modelo de objeto; acciones clave de donde residen los estados; patrones estándar, navegación, inmutabilidad y encapsulamiento, consideraciones del modelo vista controlador.

Fuente. Esta investigación

Tabla 13 *Contenidos del curso de diseño de software Universidad de Illinois - Urbana-Champaign.*

Universidad:	Universidad de Illinois en Urbana-Champaign
Ubicación:	Urbana, IL, Estados Unidos de América
Depto., Colegio o Escuela:	Departamento de Ciencias de la Computación
Profesor:	Ralph Johnson
Contenidos del Curso:	CS 427 ingeniería de software I
1. Procesos de software. 2. Análisis y diseño. 3. Paradigmas de desarrollo de software. 4. Ingeniería de sistemas. 5. Análisis y diseño basado en función. 6. Análisis y diseño orientado a objetos.	

Fuente: Esta investigación.

Tabla 14 *Contenidos del curso de diseño de software Universidad de Texas - Austin.*

Universidad:	Universidad de Texas – Austin
Ubicación:	Austin, TX, Estados Unidos de América
Depto., Colegio o Escuela:	Depto. de Ingeniería Eléctrica y Ciencias de la Computación
Profesor:	Dewayne E. Perry
Contenidos del Curso:	EE 360F – Introducción a la ingeniería de software
1. Introducción a los sistemas y la complejidad. 2. Ética en ingeniería de software, código de ética de ingeniería de software de ACM/IEEE, código de ética de la sociedad australiana de computación, código de ética de profesionales en tecnología de información. 3. Elementos de sistemas ingeniados a través de software, dimensiones de evolución de software. 4. Requerimientos, análisis de requerimientos basados en consulta, El mundo y la máquina, prototipado asistido por computador. 5. Arquitectura, fundamentos para el estudio de arquitectura de software, arquitectura de línea de producto. 6. Principios de diseño, diseñando software para fácil extensión y contracción, el proceso de diseño racional. 7. Métodos de diseño, metodologías de diseño de programas. 8. Experiencia de diseño, consejos para el diseño de sistemas. 9. Construcción y composición, análisis y captura de datos. 10. Despliegue y mantenimiento, evolución, fallas de software. 11. Artefactos y su manejo, configuración y administración. 12. Medidas y evaluación. Pruebas de software. 13. Procesos, medida y evaluación.	

Fuente: Esta investigación.

Tabla 15 *Contenidos del curso de diseño de software Universidad Carnegie Mellon.*

Universidad:	Universidad Carnegie Mellon
Ubicación:	Pittsburg, PA, Estados Unidos de América
Depto., Colegio o Escuela:	Escuela de Ciencias de la Computación
Profesor:	Jonathan Aldrich
Contenidos del Curso:	15-413 Práctica de ingeniería de software
1. El Proceso SCRUM, Introducción, la situación del desarrollo actual, metodología, fases, control, despliegue, equipo de proyecto, características, ventajas y estimaciones. 2. Análisis tradicional, prueba e inspección. 3. Análisis y diseño, patrones y marcos de trabajo. 4. Especificación de programa y verificación. 5. Análisis estático y verificación del modelo. 6. Análisis a través del ciclo de vida del software.	

Fuente: Esta investigación

Tabla 16 *Contenidos del curso de diseño de software Universidad de California - Santa Bárbara.*

Universidad:	Universidad de California - Santa Barbara
Ubicación:	Santa Barbara, CA, Estados Unidos de América
Depto., Colegio o Escuela:	Departamento de Ciencias de la Computación
Profesor:	Tevfik Bultan
Contenido del Curso:	CS189A/172 ingeniería de software
1. Enfrentando el cambio con la programación extrema. 2. Cómo Microsoft construye el software. 3. Sin balas de plata, esencia y accidentes en ingeniería de software. 4. Crisis en las crónicas de software. 5. La catedral y el bazar 6. El mítico hombre del mes 7. Iniciando: captura de requerimientos. 8. Practicas recomendadas por IEEE para la especificación de requerimientos de software. 9. El criterio de descomponer sistemas en módulos. 10. Aplicando el diseño por contratos. 11. Patrones de diseño: abstracción y reúso de diseño orientado a objetos. 12. Validación, verificación y pruebas.	

Fuente: Esta investigación.

Tabla 17 *Contenidos del curso de diseño de software EPFL – Instituto Tecnológico Federal de Suiza.*

Universidad:	EPFL – Instituto Tecnológico Federal de Suiza
Ubicación:	Lausanne, Suiza
Depto., Colegio o Escuela:	División de Ingeniería Electrónica
Profesor:	George Candea
Contenido del Curso:	CS-305 ingenierías de software.
1. Diseño y razonamiento orientado a objetos. 2. Seguridad, estabilidad y desempeño.	

Universidad:	EPFL – Instituto Tecnológico Federal de Suiza
Ubicación:	Lausanne, Suiza
Depto., Colegio o Escuela:	División de Ingeniería Electrónica
Profesor:	George Candea
Contenido del Curso:	CS-305 ingenierías de software.
3. Concurrencia. 4. Recolección de requisitos y análisis. 5. Especificaciones y documentación. 6. Pruebas, validación y actualización. 7. Usabilidad. 8. Administración del código fuente.	

Fuente: Esta investigación.

Tabla 18 *Contenidos del curso de diseño de software Colegio Imperial de Londres.*

Universidad:	Colegio Imperial de Londres
Ubicación:	Londres, Inglaterra
Depto., Colegio o Escuela:	Departamento de Computación
Profesor:	Alexander L. Wolf
Contenidos del Curso:	Comp. 202 ingeniería de software - Algoritmos
1. Revisión de análisis de complejidad asintótica. 2. Diseño de algoritmos aleatorios: esquemas de aleatoriedad, arboles de búsqueda binaria aleatoria, listas de salto, filtros de flor. 3. Diseño de algoritmos de análisis de cadenas de caracteres, fuerza bruta. 4. Diseño de algoritmos de búsqueda raíz, arboles de búsqueda digital, arboles de búsqueda binaria, arboles de búsqueda ternaria, arboles sufijo. 5. Diseño de algoritmos Divide y vencerás: multiplicación matricial, números Fibonacci y transformadas de Fourier rápidas. 6. Diseño de Transformada de Wavelet. 7. Diseño de algoritmos de compresión: compresión de menos perdida, Huffmann, LZW, programación dinámica de compresión Wavelet/DCT. 8. Diseño de algoritmos Greedy. 9. Diseño de algoritmos de grafos avanzados: flujo de red, problemas de flujo máximo, partición de grafos, algoritmo Ford-Fulkerson; Algoritmo Edmonds-Karp.	

Fuente: Esta investigación.

Tabla 19 *Contenidos del curso de diseño de software Universidad de Washington*

Universidad:	Universidad de Washington
Ubicación:	Seattle, WA, Estados Unidos de América
Depto., Colegio o Escuela:	Departamento de Ciencia de la Computación e Ingeniería
Profesor:	David Notkin
Contenidos del Curso:	CSE403: ingeniería del software
1. Crisis del software. 2. Manejando la complejidad. 3. Especificación de requerimientos. 4. Diseño arquitectónico y detallado. 5. Pruebas y análisis.	

Universidad:	Universidad de Washington
Ubicación:	Seattle, WA, Estados Unidos de América
Depto., Colegio o Escuela:	Departamento de Ciencia de la Computación e Ingeniería
Profesor:	David Notkin
Contenidos del Curso:	CSE403: ingeniería del software
6. Procesos de software. 7. Herramientas y entornos de desarrollo.	

Fuente: Esta investigación.

Tabla 20 *Contenidos del curso de diseño de software Universidad de California – Davis*

Universidad:	Universidad de California – Davis
Ubicación:	Davis, CA, Estados Unidos de América
Depto., Colegio o Escuela:	Departamento de Ciencias de la Computación
Profesor:	Premkumar T. Devanbu
Contenidos del Curso:	ECS 160 – Introducción a la ingeniería de software
1. Introducción. 2. Patrones de diseño. 3. Procesos de software. 4. Ingeniería de requisitos. 5. Programación extrema. 6. Principios de diseño de interfaces gráficas de usuario. 7. Diseño arquitectónico. 8. Redes Petri 9. Verificación e inspección. 10. Pruebas.	

Fuente: Esta investigación.

Tabla 21 *Contenidos del curso de diseño de software Universidad de Colorado – Boulden*

Universidad:	Universidad de Colorado - Boulden
Ubicación:	Boulden, CO, Estados Unidos de América
Depto., Colegio o Escuela:	Departamento de Ciencias de la Computación
Profesor:	Kenneth M. Anderson
Contenidos del Curso:	CSCI 5828 – Fundamentos de ingeniería de software
1. Introducción. 2. Complaciendo al cliente. 3. Introducción a la concurrencia. 4. Recolectando requerimientos. 5. Procesos e hilos. 6. Planeación del proyecto. 7. Ejecución concurrente. 8. Historias de usuario y tareas. 9. Objetos compartidos y ejecución mutual. 10. Un buen diseño. 11. Monitoreo y sincronización de la condición.	

Universidad:	Universidad de Colorado - Boulden
Ubicación:	Boulden, CO, Estados Unidos de América
Depto., Colegio o Escuela:	Departamento de Ciencias de la Computación
Profesor:	Kenneth M. Anderson
Contenidos del Curso:	CSCI 5828 – Fundamentos de ingeniería de software
12. Control de versiones. 13. Administración de la construcción de software. 14. Pruebas. 15. Desarrollo conducido por pruebas. 16. Seguridad. 17. Diseño basado en el modelo. 18. Finalizando una iteración. 19. La siguiente iteración. 20. Enfoque alternativo a la concurrencia. 21. Tratando con problemas.	

Fuente: Esta investigación.

Tabla 22 *Contenidos del curso de diseño de software Universidad McGill.*

Universidad:	Universidad McGill
Ubicación:	Montreal, QB, Canadá
Depto., Colegio o Escuela:	Escuela de Ciencias de la Computación
Profesor:	Joseph Vybihal
Contenido del Curso:	COMP-335 Métodos de ingeniería de software
1. Elementos de un plan, introducción a la ingeniería de software. 2. Procesos de software y modelo de procesos. 3. Sistemas dependientes. 4. El entorno del proyecto. 5. Ingeniando el programa. 6. Ingeniería de requerimientos. 7. La documentación sobre requerimientos. 8. Estudios de caso de la vida real. 9. Diseño y especificaciones. 10. Pruebas e instalación.	

Fuente: Esta investigación.

Tabla 23 *Contenidos del curso de diseño de software Universidad Duke.*

Universidad:	Universidad Duke
Ubicación:	Durham, NC, Estados Unidos de América
Depto., Colegio o Escuela:	Departamento de Ciencias de la Computación
Profesor:	Robert Duvall
Contenidos del Curso:	CompSci 108 –El Estudio de Software
1. Java avanzado, estructuras genéricas, reflexión, programación de interfaz gráfica de usuario. 2. Diseño de programas, principios, patrones, arquitecturas, marcos de trabajo. 3. Herramientas, depuración, unidades de prueba, control de versiones.	

Universidad:	Universidad Duke
Ubicación:	Durham, NC, Estados Unidos de América
Depto., Colegio o Escuela:	Departamento de Ciencias de la Computación
Profesor:	Robert Duvall
Contenidos del Curso:	CompSci 108 –El Estudio de Software
4. Trabajo en equipo, comunicación, cooperación, documentación. 5. Rehaciendo al diseño orientado a objetos.	

Fuente: Esta investigación.

Tabla 24 *Contenidos del curso de diseño de software Universidad de Boston.*

Universidad:	Universidad de Boston
Ubicación:	Boston, MA, Estados Unidos de América
Depto., Colegio o Escuela:	Departamento de Ciencias de la Computación
Profesor:	John Keklak
Contenido del Curso:	CS411 – ingeniería de software
1. Introducción. 2. Ingeniería de requerimientos. 3. Prueba de requisitos. 4. Estimación de esfuerzo y cronograma. 5. Diseño. 6. Construcción de código, buenas prácticas. 7. Introducción al UML. 8. Introducción a los patrones de diseño. 9. Introducción a la metodología Ágil. 10. Introducción a la metodología de programación extrema.	

Fuente: Esta investivación.

Tabla 25 *Contenidos del curso de diseño de software Universidad Tecnológica de Delft.*

Universidad:	Universidad Tecnológica de Delft
Ubicación:	Delft, Holanda
Depto., Colegio o Escuela:	Programa de Ciencias de la Computación
Profesor:	P.G.Kluit
Contenidos del Curso:	IN2705 – ingeniería de software I
No hay información.	

Fuente: Esta investivación.

Tabla 26 *Contenidos del curso de diseño de software Universidad de Roma "La Sapienza"*.

Universidad:	Universidad de Roma "La Sapienza"
Ubicación:	Roma, Italia
Depto., Colegio o Escuela:	Departamento de Informática
Profesor:	Paolo Bottoni
Contenidos del Curso:	ingeniería del software I
1. Ciclos de vida del software.	

Universidad:	Universidad de Roma “La Sapienza”
Ubicación:	Roma, Italia
Depto., Colegio o Escuela:	Departamento de Informática
Profesor:	Paolo Bottoni
Contenidos del Curso:	ingeniería del software I
2. Métricas para el dimensionamiento del esfuerzo. 3. Introducción a UML. 4. Casos de Uso. 5. Diagrama de clases. 6. Diagrama de interacción. 7. Diagrama de colaboración. 8. Diagrama de estados. 9. Diagrama de actividades.	

Fuente: Esta investivación.

Cuatro expertos profesores de diseño de software en Colombia han ensamblado sus propios syllabus de los cursos. A continuación se presenta los contenidos generales de cada uno de ellos:

Tabla 27 *Contenidos del curso de diseño de software Escuela de Ingeniería de Antioquia.*

Universidad:	Escuela de Ingeniería de Antioquia
Ubicación:	Medellín, Colombia
Depto., Colegio o Escuela:	Ingeniería Informática
Profesor:	Diego Hernán Montoya Bedoya
Contenidos del Curso:	Ingeniería de software
1. Planear proyectos informáticos 2. Administrar la construcción de software 3. Evaluación y análisis de calidad de los productos de software 4. Administrar el mantenimiento del software	

Fuente: Esta investivación.

Tabla 28 *Contenidos del curso de diseño de software Universidad de Nariño*

Universidad:	Universidad de Nariño
Ubicación:	Pasto, Colombia
Depto., Colegio o Escuela:	Departamento de Sistemas
Profesor:	Alexander Barón
Contenidos del Curso:	Introducción a la ingeniería de software
1. Visión general del desarrollo de software 2. Ingeniería de requisitos 3. Análisis y Diseño 4. Calidad de software	

Fuente: Esta investivación.

Tabla 29 *Contenidos del curso de diseño de software Universidad del Cauca*

Universidad:	Universidad del Cauca
Ubicación:	Popayán, Colombia
Depto., Colegio o Escuela:	Departamento de Sistemas
Profesor:	Carlos Cobos
Contenidos del Curso:	Ingeniería de software I
1. Estimación de software 2. Ingeniería de requerimientos 3. Análisis y diseño orientado a objetos 4. Gestión de riesgos 5. Fundamentos de Interacción humano-computador	

Fuente: Esta investivación.

Tabla 30 *Contenidos del curso de diseño de software Fundación Universitaria San Martin*

Universidad:	Fundación Universitaria San Martin
Ubicación:	Bogotá, Colombia
Depto., Colegio o Escuela:	Ingeniería de Sistemas
Profesor:	Armando Rafael Acuña Martínez
Contenidos del Curso:	Fundamentos de construcción de software I
No hay Información	

Fuente: Esta investivación.

2.1.3 Contenidos según SWEBOK: The Guide to the Software Engineering.

Finalmente, el proyecto de la sociedad de computación de IEEE y el comité de prácticas profesionales de la misma organización establece un documento como la guía para el cuerpo del conocimiento en el área de la ingeniería de software. Este es el tercer insumo que nutre la programación de contenidos a impartirse en el curso de diseño de software a través de su didáctica.

En esta guía, la Sociedad de Computación de IEEE establece por primera vez una línea base para el cuerpo del conocimiento en el campo de la ingeniería de software, y este trabajo está motivado por el desarrollo y evolución desde la teoría y la practica en diseño de software. Esta guía pretende dar las directrices generales en cuanto a contenidos para los cursos relacionados con el diseño de software.

Esta guía como lo muestra la Figura 6 trata principalmente los siguientes temas:

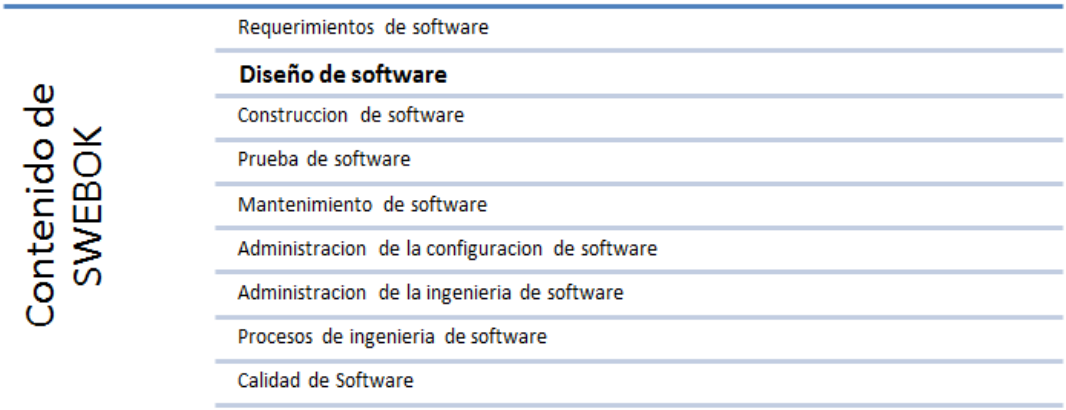


Figura 6 *Aspectos generales de la ingeniería de software según SWEBOK*
Fuente: Tomado de IEEE – SWEBOK

Teniendo en cuenta que el diseño de software es solo una parte del texto, será menester estudiar únicamente los aspectos de dicha temática. De acuerdo con la definición de IEEE, diseño son dos cosas: la primera trata del proceso de definición de la arquitectura, componentes, interfaces, y otras características de un sistema o componente. Y la segunda es el resultado de un proceso. El diseño de software para IEEE se subdivide en 6 subáreas:

Sub áreas del diseño de software según SWEBOK.

1. Fundamentos del diseño de software:
   La cual conforma una base subrayada al entendimiento del rol y el ámbito del diseño de software. Hay conceptos generales de software, el contexto del diseño, el proceso del diseño mismo, y la habilitación de técnicas para el diseño.

2. Acciones claves en el diseño de software:
   Ellas incluyen concurrencia, control y manejabilidad de eventos, distribución de componentes, error y manejo de excepciones, tolerancia a fallas, interacción, presentación y persistencia de datos.

3. Estructuras del software y arquitectura:
   Son los tópicos en los cuales las estructuras arquitectónicas aparecen junto con los puntos de vista, los estilos arquitectónicos, patrones de diseño, y finalmente las familias de programas y marcos de trabajo.

4. Análisis de calidad del diseño de software y la evaluación:
   Esta sub área presenta los tópicos estrictamente relacionados con el aseguramiento de la calidad, los cuales incluyen atributos, análisis, técnicas de evaluación y métricas.

5. Notación del diseño de software:
   Esta sub área trata sobre las descripciones de estructura y comportamiento dentro del diseño de software.

6. Estrategias de diseño de software y métodos:
Se describen estrategias generales junto con métodos de diseño orientado a funciones, métodos de diseño orientado a objetos, diseños centrados en las estructuras de datos, diseño basado en componentes, diseño orientado al servicio entre otros.

Esta configuración se puede observar en la Figura 7:

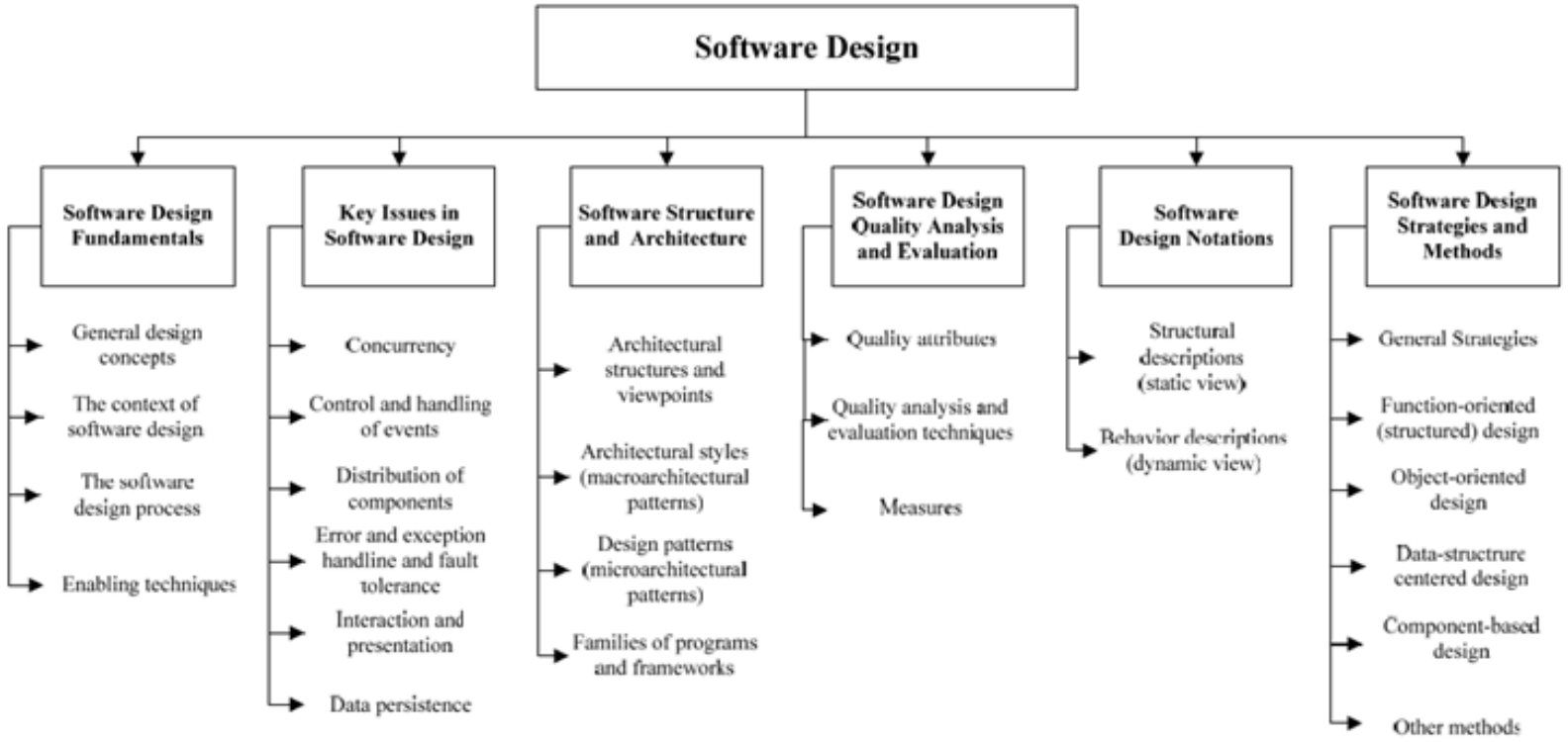


Figura 7 *Detalle de elementos para el diseño de software para IEEE en SWEBOK*
Fuente: Tomado de IEEE - SWEBOK

2.2 Construcción de los contenidos del curso de diseño de software

Tabla 31 *Matriz de referentes conceptuales de contenidos en diseño de software*

	Contenidos generales	Computing Curricula - ACM	SWEBOK - IEEE Computer Society
Instituto de Tecnología de Massachusetts	- Programación en Java - Maquinas de Estado - Seguridad y Tolerancia a Fallos - Diseño Orientado a Objetos - Diseño de Interfaces de Usuario - Semánticas - Modelo relacional	- Tópicos: - Conceptos de fundamentos de diseño y principios. El rol y el uso de contratos. - Patrones de diseño. - Arquitectura de software. - Diseño estructurado. - Análisis y diseño orientado a objetos. - Diseño a nivel de componentes. - Cualidades de diseño. - Bajo acoplamiento interno, alta cohesión, ocultamiento de información y eficiencia. - Confiabilidad externa, mantenibilidad, usabilidad, y desempeño. - Otros enfoques: centrado en la estructura de datos, orientación a aspectos, orientación a la función, orientación al servicio, diseño ágil. - Diseño por reúso. Objetivos de Aprendizaje: - Discutir las propiedades de un buen diseño de software incluyendo la naturaleza y el rol de la documentación asociada. - Evaluar la calidad de los múltiples diseños de software basados en	Diseño de software 1. Fundamentos del diseño de software: - Conceptos generales de diseño - El contexto del diseño de software - El proceso de diseño de software - Habilitación de técnicas 2. Acciones claves en el diseño de software: - Concurrencia - Control y manejo de eventos - Distribución de componentes - Error, manejo de excepciones y tolerancia - Interacción y presentación - Persistencia de datos 3. Estructuras del software y arquitectura: - Estructura de arquitectura - puntos de vista - Estilos de arquitectura (macro arquitectura) - Patrones de diseño (micro arquitectura) - Familias de programas y marcos de trabajo
Universidad de Illinois - Urbana Champaign	- Procesos de software - Análisis y diseño - Paradigmas de desarrollo de software - Ingeniería de sistemas - Análisis y diseño basado en función - Análisis y diseño basado en objetos		
Universidad de Texas - Austin	- Sistemas y complejidad - Ética - Análisis de requisitos - Arquitectura - El procesos racional - Metodologías de diseño - Despliegue de software - Evaluación de software		
Universidad Carnegie Mellon	- El proceso SCRUM - Análisis tradicional - Patrones y marcos de trabajo - Especificaciones de programa y verificación - Análisis estático y verificación - Ciclos de vida del software		

	Contenidos generales	Computing Curricula - ACM	SWEBOK - IEEE Computer Society
Universidad de California - Santa Barbara	- Programación Extrema - Captura de requerimientos - Descomposición modular - Patrones, abstracción, reúso de software - Orientación a objetos - Validación y pruebas	principios y conceptos claves del diseño. - Seleccionar y aplicar patrones de diseño apropiados en la construcción de una aplicación de software. - Crear y especificar el diseño de software para un producto computacional de tamaño medio usando especificaciones de requerimiento de software y una metodología de diseño de programa aceptada así como la apropiada notación. - Evaluar un diseño de software a nivel de componentes. - Evaluar un diseño de software desde la perspectiva del reúso.	4. Análisis de calidad del diseño de software y la evaluación: - Atributos de calidad - Análisis de calidad y técnicas de Evaluación - Métricas 5. Notación del diseño de software: - Descripciones estructurales (vista estática) - Descripciones comportamentales (vista Dinámica) 6. Estrategias de diseño de software y métodos: - Estrategias generales - Diseño orientado a funciones - Diseño orientado a objetos - Diseño centrado en estructuras de datos - Diseño basado en componentes - Otros métodos
Swiss Instituto Federal de Tecnología Lausanne	- Diseño y razonamiento orientado a objetos - Seguridad, estabilidad y desempeño - Concurrencia - Recolección de requisitos y análisis - Especificaciones y documentación - Pruebas, usabilidad y manejo de código fuente		
Colegio Imperial de Londres	- Diseño de análisis de complejidad - Diseño de algoritmos aleatorios - Diseño de algoritmos de análisis de cadenas - Diseño de algoritmos de búsquedas - Diseño de algoritmos de compresión - Diseño de transformadas y grafos		
Universidad de Washington – Seattle	- Manejo de la Complejidad - Especificación de requerimientos - Diseño arquitectónico y detallado - Pruebas y análisis - Procesos de software - Herramientas y entornos de desarrollo		
Universidad de California – Davis	- Patrones de diseño - Procesos de software - Ingeniería de requisitos - Programación extrema - Principios de diseño de interface de usuario - Diseño arquitectónico - Verificación y pruebas		

	Contenidos generales	Computing Curricula - ACM	SWEBOK - IEEE Computer Society
Universidad de Colorado – Boulder	- Comprendiendo a cliente - Concurrencia - Recolección de requerimientos - Procesos e hilos - Planeación del proyecto - Historias de usuario - Control de versiones - Administración de la construcción de software - Pruebas y seguridad		
Universidad McGill	- Planeación de proyectos de desarrollo de software - Procesos de software y modelo de procesos - Sistemas dependientes - Modelo del proyecto - Ingeniería de requerimientos - Estudio de casos - Diseño y especificaciones - Pruebas y verificación		
Universidad Duke	- Programación en Java - Diseño de programas, principios y arquitecturas - Patrones y marcos de trabajo - Herramientas, depuración y unidades de prueba - Trabajo en equipo y documentación - Rehaciendo el diseño orientado a objetos		
Universidad Boston	- Ingeniería de requerimientos - Pruebas de requisitos - Estimación de esfuerzo y cronograma - Diseño - Construcción de código - UML y patrones - Metodología ágil y programación extrema		
Universidad de Tecnología Delft	No hay información		

	Contenidos generales	Computing Curricula - ACM	SWEBOK - IEEE Computer Society
Universidad de Roma - La Sapienza	- Ciclos de vida de desarrollo de software - Métricas para dimensionar el esfuerzo - UML		
Escuela de Ingeniería de Antioquia	- Planear proyectos informáticos - Administrar la construcción de software - Evaluación y análisis de calidad de los productos de software - Administrar el mantenimiento del software		
Universidad de Nariño	- Visión general del desarrollo de software - Ingeniería de requisitos - Análisis y Diseño - Calidad de Software		
Universidad del Cauca	- Estimación de software - Ingeniería de requerimientos - Análisis y diseño orientado a objetos - Gestión de riesgos - Fundamentos de Interacción humano-computador		
Fundación Universitaria San Martin	No hay información		

Fuente: Esta investigación

Una vez descritos y analizados los contenidos según ACM, los 15 expertos internacionales y 4 nacionales, y el documento SWEBOK de IEEE Computer Society, el curso de diseño de software que se está planteando deberá tener los siguientes contenidos. Para ello se ha contemplado los aspectos comunes y las experiencias descritas como positivas según las encuestas, cabe anotar que los aspectos comunes de cada syllabus están orientados por las recomendaciones tanto de ACM como de SWEBOK.

Tabla 32 *Síntesis de contenidos para curso de diseño de software*

	Contenidos generales	**Computing Curricula - ACM**	**SWEBOK - IEEE Computer Society**
Universidades…	…	…	…
CONTENIDOS PARA EL CURSO DE DISEÑO DE SOFTWARE (Resultado del análisis de los tres insumos)	Unidad 1: Paradigmas del diseño de software. - El rol del diseñador de software. - Conceptos generales de diseño de software (compatibilidad, escalabilidad, tolerancia a fallos, mantenibilidad, modularidad, Confiabilidad, reusabilidad, seguridad, etc.) - Diseño orientado a funciones. - Diseño orientado a objetos. - Diseño centrado a la estructura de datos. - Diseño basado en componentes. - Metodologías ágiles. - Programación extrema. - Otras metodologías. Unidad 2: Características inherentes al software. - Estudio de la concurrencia. - Control de eventos, procesos e hilos. - Manejo de excepciones. - Persistencia de datos. Unidad 3: Notación para el diseño con UML. - Simbología. - Vistas. - Diagramas. Unidad 4: Requerimientos de Software. - Análisis de requerimientos. - Especificación de requerimientos. Unidad 5: Estructura y arquitectura del software. - Arquitectura conducida por el modelo. - Especificación de infraestructura. - Especificación de superestructura. Unidad 6: Construcción de software a partir del diseño. - Estándares de codificación. - Control de versiones. Unidad 7: Pruebas de software. - Técnicas de pruebas y verificación.		

Fuente: Esta investigación

Al final, los contenidos propuestos son:

Unidad 1: Paradigmas del diseño de software.

1. El rol del diseñador de software.
2. Conceptos generales de diseño de software (compatibilidad, escalabilidad, tolerancia a fallos, mantenibilidad, modularidad, confiabilidad, reusabilidad, seguridad, etc.)
3. Diseño orientado a funciones.
4. Diseño orientado a objetos.
5. Diseño centrado a la estructura de datos.
6. Diseño basado en componentes.

7. Metodologías ágiles.
8. Programación extrema.
9. Otras metodologías.

Unidad 2: Características inherentes al software.
1. Estudio de la concurrencia.
2. Control de eventos, procesos e hilos.
3. Manejo de excepciones.
4. Persistencia de datos.

Unidad 3: Notación para el diseño con UML.
1. Simbología.
2. Vistas.
3. Diagramas.

Unidad 4: Requerimientos de software.
1. Análisis de requerimientos.
2. Especificación de requerimientos.

Unidad 5: Estructura y arquitectura del software.
1. Arquitectura conducida por el modelo.
2. Patrones de diseño (Especificación de infraestructura).
3. Estilos de arquitectura (Especificación de superestructura).

Unidad 6: Construcción de software a partir del diseño.
1. Estándares de codificación.
2. Control de versiones.

Unidad 7: Pruebas de software.
1. Técnicas de pruebas y verificación.

2.3 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 8, indicándose los insumos, procesos y resultados frutos de la investigación del segundo objetivo específico.

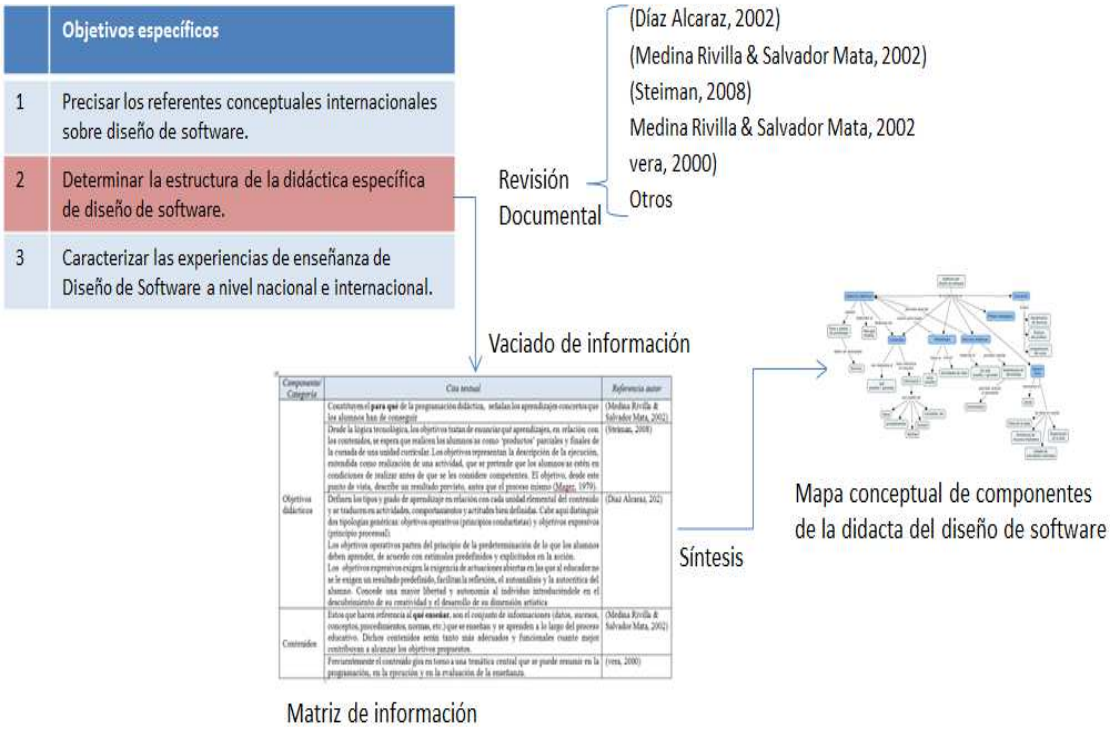


Figura 8 *La metodología seguida para identificar los elementos constitutivos de la Didáctica de Diseño de Software*
Fuente: Esta investigación

Para determinar la estructura de elementos que constituyen la didáctica específica del diseño de software, se construyó la matriz de referentes como se observa en la tabla 33 de diversos autores que explican las características que deben tener los componentes de una didáctica.

Tabla 33 *Matriz de vaciado de información para precisar los componentes de la Didáctica del Diseño de Software*

	Cita textual (referencia autor)
Objetivos didácticos	Constituyen el **para qué** de la programación didáctica, señalan los aprendizajes concretos que los alumnos han de conseguir (Medina Rivilla & Salvador Mata, 2002)
	Desde la lógica tecnológica, los objetivos tratan de enunciar qué aprendizajes, en relación con los contenidos, se espera que realicen los alumnos/as como 'productos' parciales y finales de la cursada de una unidad curricular. Los objetivos representan la descripción de la ejecución, entendida como realización de una actividad, que se pretende que los alumnos/as estén en condiciones de realizar antes de que se les considere competentes. El objetivo, desde este punto de vista, describe un resultado previsto, antes que el proceso mismo (Mager, 1979). (Steiman, 2008)
	Definen los tipos y grado de aprendizaje en relación con cada unidad elemental del contenido y se traducen en actividades, comportamientos y actitudes bien definidas. Cabe aquí distinguir dos tipologías genéricas: objetivos operativos (principios conductistas) y

	Cita textual (referencia autor)
	objetivos expresivos (principio procesual). Los objetivos operativos parten del principio de la predeterminación de lo que los alumnos deben aprender, de acuerdo con estímulos predefinidos y explicitados en la acción. Los objetivos expresivos exigen la exigencia de actuaciones abiertas en las que al educador no se le exigen un resultado predefinido, facilitan la reflexión, el autoanálisis y la autocrítica del alumno. Concede una mayor libertad y autonomía al individuo introduciéndole en el descubrimiento de su creatividad y el desarrollo de su dimensión artística (Diaz Alcaraz, 2002)
Contenidos	Estos que hacen referencia al **qué enseñar**, son el conjunto de informaciones (datos, sucesos, conceptos, procedimientos, normas, etc.) que se enseñan y se aprenden a lo largo del proceso educativo. Dichos contenidos serán tanto más adecuados y funcionales cuanto mejor contribuyan a alcanzar los objetivos propuestos. (Medina Rivilla & Salvador Mata, 2002)
	Frecuentemente el contenido gira en torno a una temática central que se puede resumir en la programación, en la ejecución y en la evaluación de la enseñanza. (vera, 2000)
	Los contenidos representan el eje central de todo proyecto didáctico y es aquello que primero se nos representa mentalmente a la hora de pensar la cátedra. Los contenidos son la respuesta a una pregunta crucial de la práctica docente: ¿qué enseñar? La SELECCIÓN de los contenidos que vamos a enseñar suele ser, en general, una de las decisiones más 'fuertes' que tomamos como docentes. El hecho de poder elegir los contenidos a enseñar no es, sin embargo, algo que pueda hacerse al margen del escenario global que representan el plan de estudios y el proyecto curricular institucional, cuando lo hay. En este sentido la primera prescripción que atraviesa el trabajo en torno a los contenidos, está representada por la presencia de los contenidos mínimos presentes en el plan de estudios. (Steiman, 2008)
Metodología (actividades de clase)	La metodología representa el **cómo enseñar**, es el camino por medio del cual se pretenden conseguir los objetivos previstos, incluye pues las **actividades (tareas) o actuaciones de toda índole** que los alumnos deben realizar para alcanzar los objetivos previstos y dominar los contenidos seleccionados, es importante dispones de un amplio y variado repertorio de actividades para poder atender el estilo y ritmo de aprendizaje de cada alumno (Medina Rivilla & Salvador Mata, 2002)
	Creo que la palabra método no puede dejar de estar porque así fue como Comenio lo denomina en su Didáctica Magna, al fin y al cabo la piedra fundacional del campo, preocupado por indicar el tipo de comportamiento que el verdadero maestro debe asumir ante sus discípulos. Gloria Edelstein (1996) en la línea de análisis del propio Ángel Díaz Barriga (1985) y de Alfredo Forlán (1986), entre otros, afirma que es el propio docente quien construye su propuesta de trabajo, de allí que retome, tal como ella lo afirma, 'tentativamente' la categoría de construcción metodológica. Dos cuestiones resultan importantes para la consideración del método: no podrán obviarse ni las características específicas del contenido, ni las de los sujetos (reales y concretos) que aprenden. (Steiman, 2008)
Recursos didácticos	Toda programación didáctica requiere el uso de unos materiales para su puesta en marcha, hemos de prever con antelación el **con qué** enseñar, los alumnos deben disponer de los materiales y recursos necesarios, que les permitan alcanzar plenamente los objetivos previstos (Medina Rivilla & Salvador Mata, 2002)
	Capacidad que los distintos medios poseen de poner al estudiante directa o indirectamente ante experiencias de aprendizaje, en la que predomina más la razón práctica que academicista como son: Recursos o medios reales: son los objetos de cualquier tipo que

	Cita textual (referencia autor)
	considere el profesor útiles para enriquecer las actividades, mejorar la motivación, dar significado al contenido, enriquecer la evaluación (ej. Planas, objetos de uso cotidiano, animales, etc.) Recursos o medios escolares: Cuyo único y prioritario destino es colaborar en el proceso de enseñanza (laboratorios, pizarras, tableros, etc.) Recursos o medios simbólicos: Son aquellos que pueden aproximar a la realidad a los estudiantes, a través de símbolos o imágenes. Puede darse por medio impreso o con el uso de las nuevas tecnologías. (Medina Rivilla & Salvador Mata, 2002)
	Es entonces en la línea que aquí presentamos desde donde planteamos este capítulo. Queremos hacer eje en algunos 'instrumentos' para trabajar metodológicamente en las aulas de la educación superior. "Las técnicas y procedimientos, en la visión que sostengo, se constituyen, en consecuencia, en instrumentos válidos, formas operativas articuladas en una propuesta global signada por un estilo de formación, que integra a modo de enfoque, perspectivas de corte filosófico ideológico, ético, estético, científico y pedagógico" (Edelstein, 1996). No hablaremos nosotros de procedimientos ni de técnicas porque la intención no es proponer formas secuenciadas de intervención. Nos limitaremos a presentar algunos recursos didácticos. Entendemos, genéricamente a los recursos didácticos como los materiales de apoyo a la enseñanza. (Steiman, 2008)
	Los recursos didácticos son algunos medios que se utilizan para alcanzar los objetivos del proceso de enseñanza– aprendizaje. "En la actualidad, el término didáctica tiene dos sentidos: uno restringido, que es el más usado, como teoría de la enseñanza o de la institución, y otro más amplio, que es el que nosotros le damos como teoría de los medios de la educación." (Lorenzo Luzuriaga, Diccionario de pedagogía, p. 115.) Desde luego que el sentido restringido se refiere a los métodos, los procedimientos y las formas de enseñar, todo esto dirigido a un plan de estudios y una asignatura, y en cuanto al sentido amplio, se refiere a la "técnica de educación en todos sus aspectos". (Lorenzo Luzuriaga, Diccionario de pedagogía, p. 115.) El material o recurso didáctico es un auxiliar o herramienta que debe tener como objetivo principal acercar el conocimiento al educando, es decir, "llevar al alumno a trabajar, a investigar, a descubrir y a construir". (Imídeo G. Nérici, Hacia una didáctica general dinámica, p. 285) En la actualidad algunos recursos didácticos deben colaborar con el educando en la construcción de su propio conocimiento con la finalidad de adquirir experiencia sobre los asuntos apegados a las actividades y procesos que se realizan diariamente en su entorno socioeconómico. (Wals, 2010)
Espacio físico	Capacidad que los distintos medios poseen de poner al estudiante directa o indirectamente ante experiencias de aprendizaje, en la que predomina más la razón práctica que academicista como son: Recursos o medios reales: son los objetos de cualquier tipo que considere el profesor útiles para enriquecer las actividades, mejorar la motivación, dar significado al contenido, enriquecer la evaluación (ej. Planas, objetos de uso cotidiano, animales, etc.) Recursos o medios escolares: Cuyo único y prioritario destino es colaborar en el proceso de enseñanza (laboratorios, pizarras, tableros, etc.) Recursos o medios simbólicos: Son aquellos que pueden aproximar a la realidad a los estudiantes, a través de símbolos o imágenes. Puede darse por medio impreso o con el uso de las nuevas tecnologías. (Medina Rivilla & Salvador Mata, 2002)
	La dinámica que siguen los procesos de aula, y su influencia en el aprendizaje y el rendimiento de los alumnos también ocupa buena parte de la preocupación de los profesores. Por eso se plantean problemas sobre la organización del aula, el clima de la clase, la interacción social del grupo-clase, la calidad de las actividades que se realizan, la pertinencia de los recursos que se emplea, etc. (Medina Rivilla & Salvador Mata, 2002)

	Cita textual (referencia autor)
	Los espacios escolares, la distribución, le mobiliarios, las paredes, los escritorios, los dibujos del aula están cargados de significados. En la vida cotidiana se construyen estos significados. El espacio registra "huellas" de esa vida que los grupos-clase pasaran en el aula. En este sentido podemos afirmar metafóricamente que los "espacio hablan". Esta investigación realiza una aproximación a la escuela concentrándonos en sus espacios: fachadas aulas, pasillos, profesores, etc. (de Camilloni, Davini, Edelstein, Litwin, Souto, & Barco, 2001)
	S i hay algo que al pensarlo se nos representa a los docentes con un estereotipo clásico, eso es el aula. La imagen mental inmediatamente se convierte en una pintura en la que aparecen, con ciertos particularismos idiosincráticos, un espacio físico en el que se identifican alumnos (variarán las edades según el caso), un docente (¿variará el género según el caso?), un pizarrón (clásicamente negro, verde o los relativamente modernos blancos para escribir con fibrones) y pupitres o bancos o mesitas… Pero el aula real, esa aula en la que trabajamos todos los días, lejos de cualquier estereotipo es increíblemente única. Y si decimos el 'aula de tercero' se nos aparecen con nombre y apellido las caras borrosas, toman cuerpo las escrituras a medio escribir del pizarrón y sentimos en la piel misma entrometerse las voces que encierran sus paredes: "… no entiendo" "… ¿qué te puso?" "… no estudié" "… lo que pasa que a mí esta materia no me gusta" "… me saqué un diez" Las voces del aula son el registro de lo que en ella transcurre y el registro de una parte de tu historia (¿cuántas horas de tu vida has pasado en un aula?). Las voces del aula son tu memoria, la memoria de tus alumnos/as, la memoria de tu profesión, la de la enseñanza, la del aprendizaje. Y allí, en el aula, pensamos nuestro trabajo, actuamos y pensamos casi sin detenernos, urgidos por la espontaneidad de lo cotidiano. Quiero, por un instante, poder pensar 'mi' aula. Me interno en ella y pienso en mí, en mi práctica y pienso con mis alumnos/as mi enseñanza, y pienso con mis alumnos/ as sus aprendizajes. Y pienso el aula y pienso en mi esfuerzo y des esfuerzo por plantear un 'escenario didáctico', por proponer genuinas situaciones de aprendizaje. Y pienso el aula y pienso mis prácticas de evaluación y se entrelazan estas reflexiones que aquí empiezo a construir. (Steiman, 2008)
Evaluación	La evaluación de la programación persigue tomar decisiones en torno a una determinada intervención docente, con un grupo concreto de alumnos, para comprobar su eficacia, pero dicha evaluación no debe restringirse solo a la valoración del rendimiento de los alumnos, sino que ha de abarcar, la evaluación de la práctica del profesor y la evaluación de la programación misma como técnica didáctica. (Medina Rivilla & Salvador Mata, 2002)
	En general el término evaluar viene de valorar o fijar un valor. En el ámbito educativo la "evaluación implica una valoración o un juicio de valor" (Carlos Zarzar Charur, Habilidades básicas para la docencia, p. 62.). A la evaluación se le conoce como el juicio de experto "ya que toda su ciencia consiste en que una persona posee la autoridad necesaria, emite un juicio acerca de la calidad o valor de algo" (Clifton B. Chadwick y Nelson Rivera, Evaluación formativa para el docente, p. 55.). La evaluación debe ser un proceso enfocado a "un análisis consciente e inconsciente de algo, para emitir una opinión acerca de algo" (Clifton B. Chadwick y Nelson Rivera, Evaluación formativa para el docente, p. 55.). Es importante comentar que durante sus clases en el aula, el profesor generalmente realiza evaluación, la cual se verifica con lo expuesto en algún tema a través de preguntas y respuestas hechas a los integrantes del grupo, y con esto constata que el alumno adquirió el conocimiento y está en condiciones de resolver prácticas o cuestionamientos sobre problemas referentes al

	Cita textual (referencia autor)
	tema de estudio. Con lo anterior el profesor estará en condiciones de verificar si el alumno ha tenido cambios en su recepción y aplicación del conocimiento. También contribuye esto a constatar si el maestro y los alumnos están cumpliendo con el objetivo fijado en el proceso de enseñanza– aprendizaje. (Wals, 2010)

Fuente: Esta investigación.

En síntesis, se construyó un mapa conceptual como se indica en la figura 9, como resultado del análisis de la revisión documental mostrada anteriormente, 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.

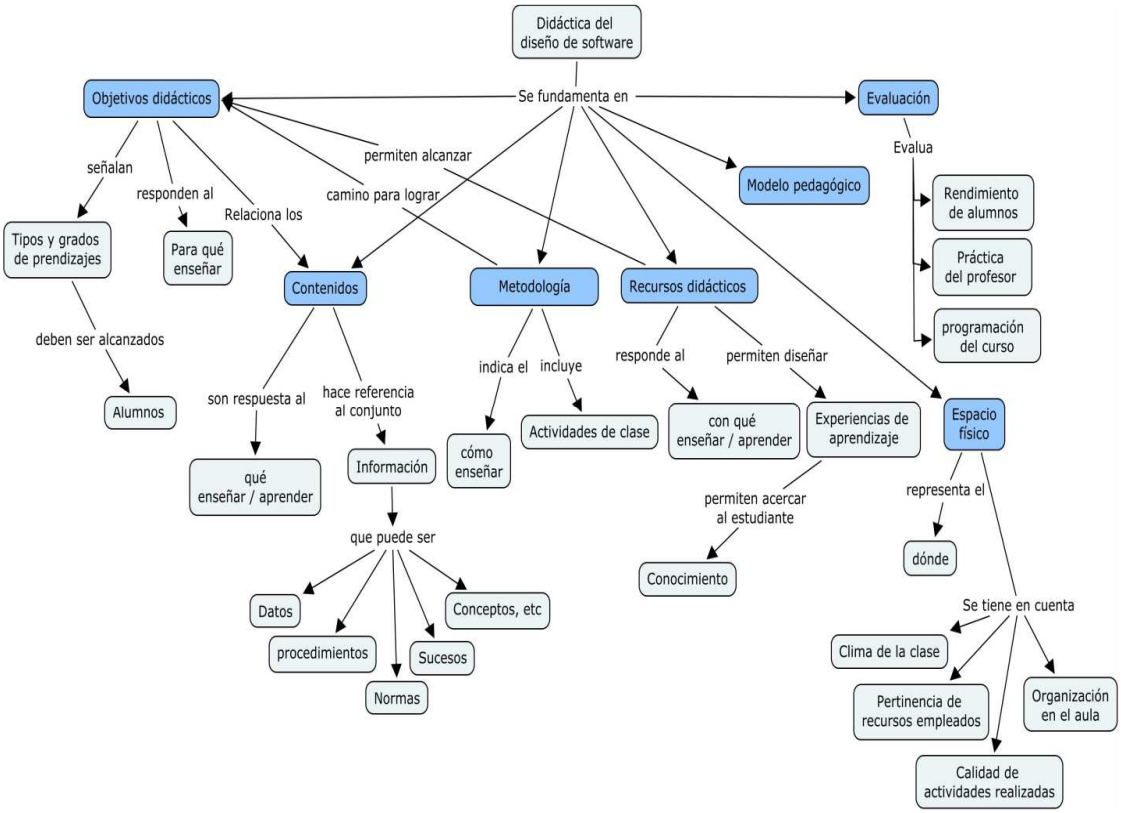


Figura 9 *Mapa conceptual síntesis de los componentes de la Didáctica del Diseño de Software.* Fuente: Esta investigación.

2.4 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 investiga9ción 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 10.

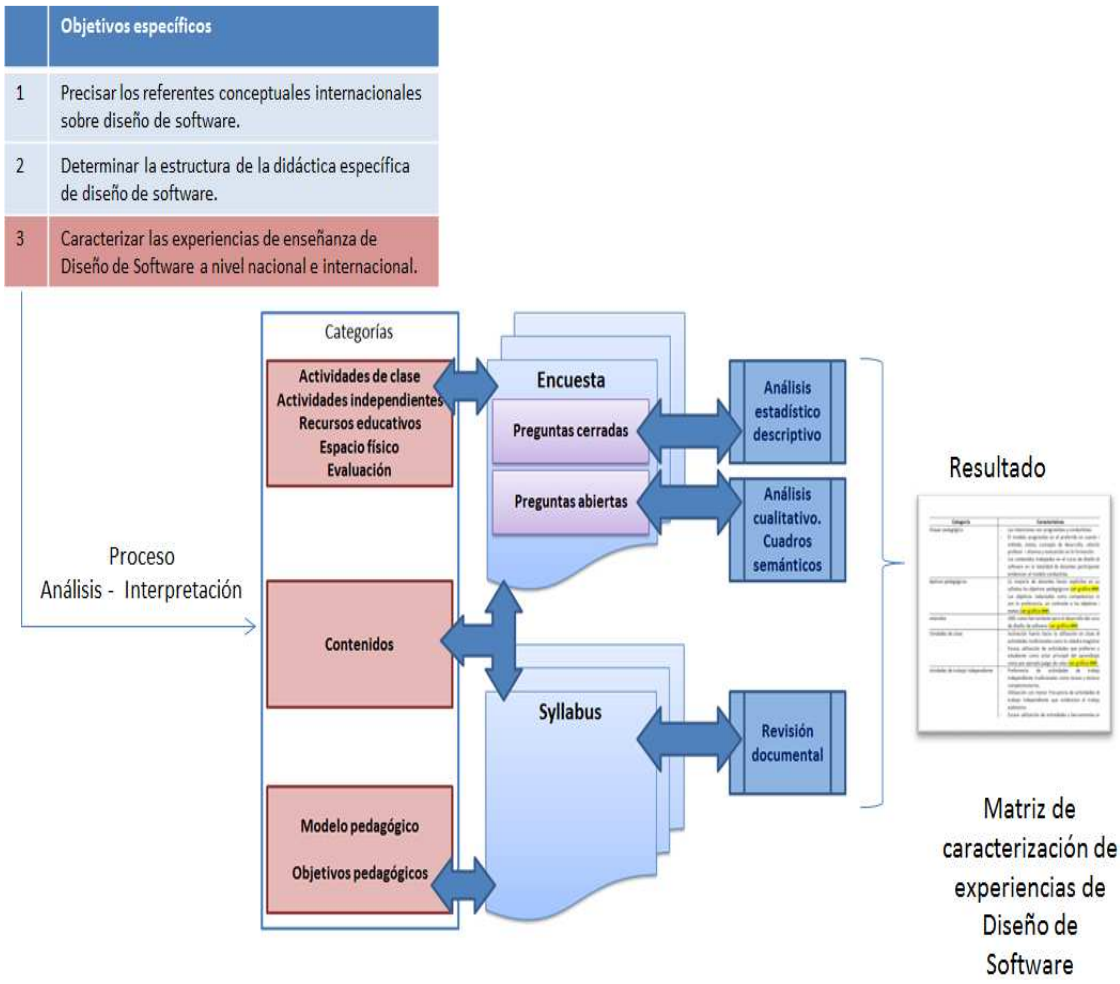


Figura 10 *Metodología para caracterizar la enseñanza del diseño de software*
Fuente: Esta investigación.

2.4.1 Análisis estadístico descriptivo.

Pregunta 1

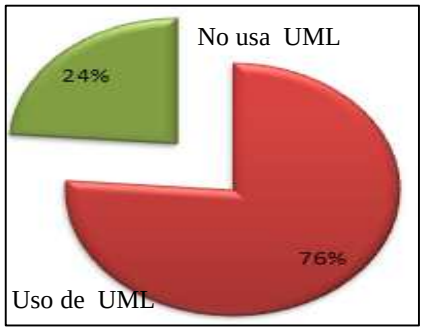


Figura 11 *Uso de UML en la enseñanza del curso de diseño de software*
Fuente: Esta investigación.

En la figura 11 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.

Pregunta 2

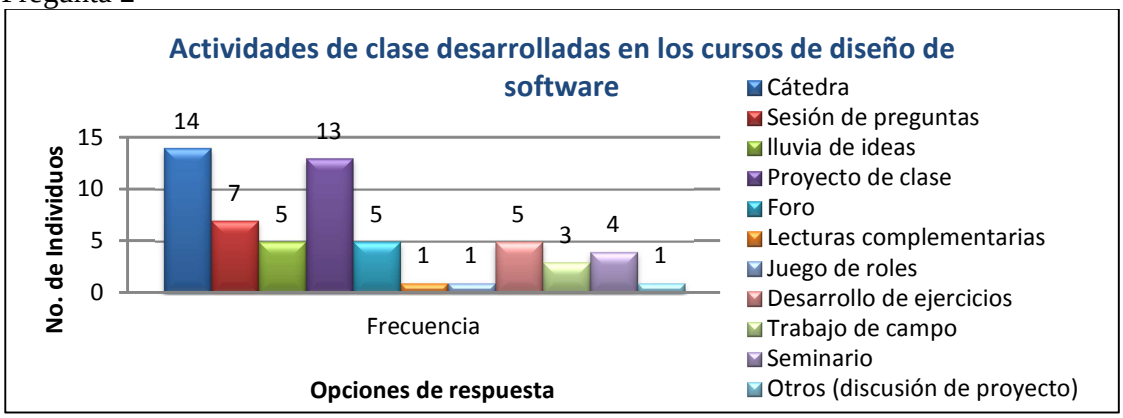


Figura 12 *Actividades de clase desarrolladas en los cursos de diseño de software*
Fuente: Esta investigación.

La figura 12 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.

Pregunta 3

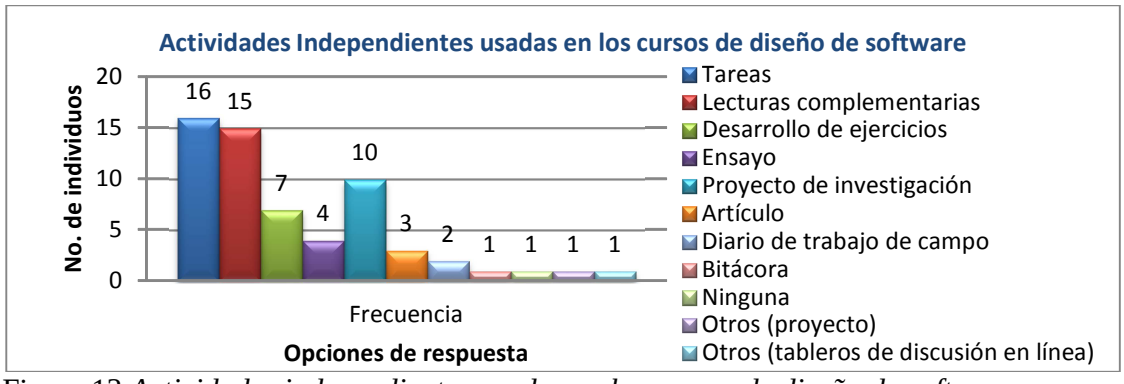


Figura 13 *Actividades independientes usadas en los cursos de diseño de software*
Fuente: Esta investigación.

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 13 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.

Pregunta 4

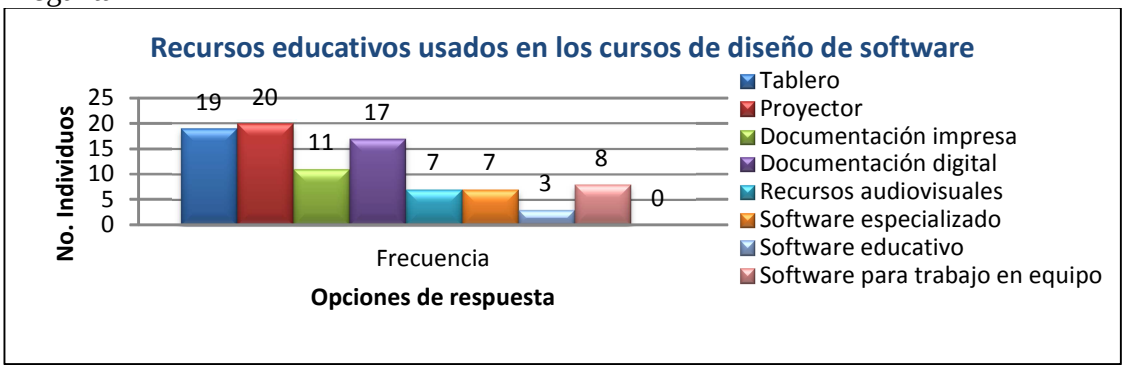


Figura 14 *Recursos educativos usados en los cursos de diseño de software*
Fuente: Esta investigación.

En cuanto a los recursos educativos usados en el desarrollo del curso de diseño de software, la figura 14 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.

Pregunta 5

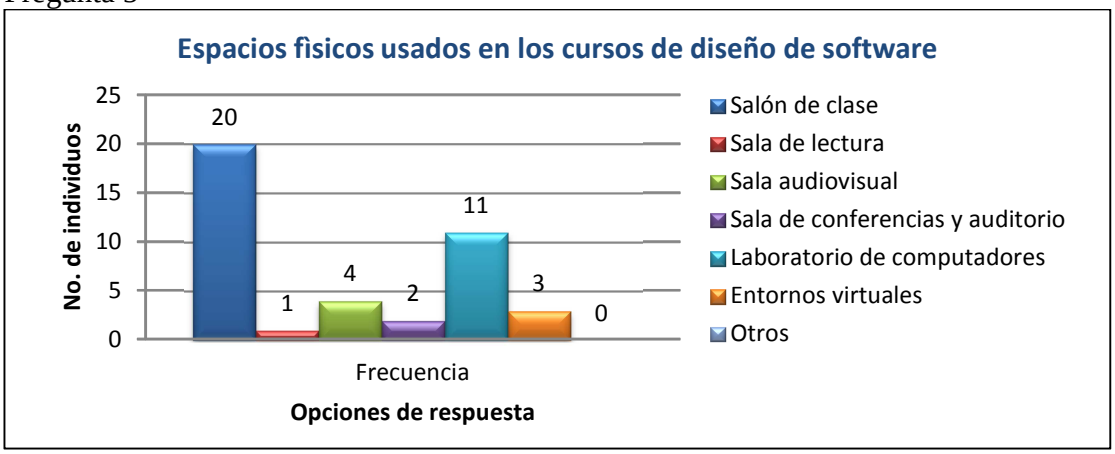


Figura 15 *Espacios físicos usados en los cursos de diseño de software*
Fuente: Esta investigación.

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 (ver capítulo 3) en este sentido.

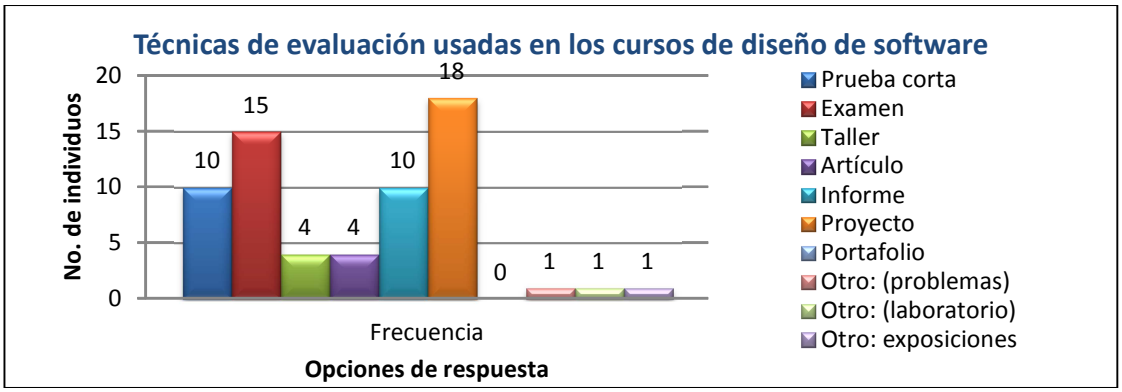


Figura 16 *Técnica de evaluación usadas en los cursos de diseño de software*
Fuente: Esta investigación.

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%. 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 17 *Manifestación de objetivos de aprendizaje en los Syllabus*
Fuente: Esta investigación.

En cuanto a la manifestación de los objetivos de aprendizaje en los Syllabus de los cursos de diseño de software, la 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 18 *Clase de redacción en los objetivos pedagógicos*
Fuente: Esta investigación.

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 los Syllabus que expresan los objetivos pedagógicos, 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.

2.4.2 Análisis cualitativo.

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 19, 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 respuestas por recurrencias; a partir de ello se construyen predicados y descriptores, que se constituyen en insumos para determinar las mezclas que serán integradas en los cuadros semánticos. El proceso completo se encuentra disponible en http://www.udenar.edu.co/dfsd-uml/file.axd?file=2010%2f9%2fQA.pdf

Figura 19 *Metodología de análisis cualitativo*
Fuente: Esta investigación.

2.5 Cuadros semánticos

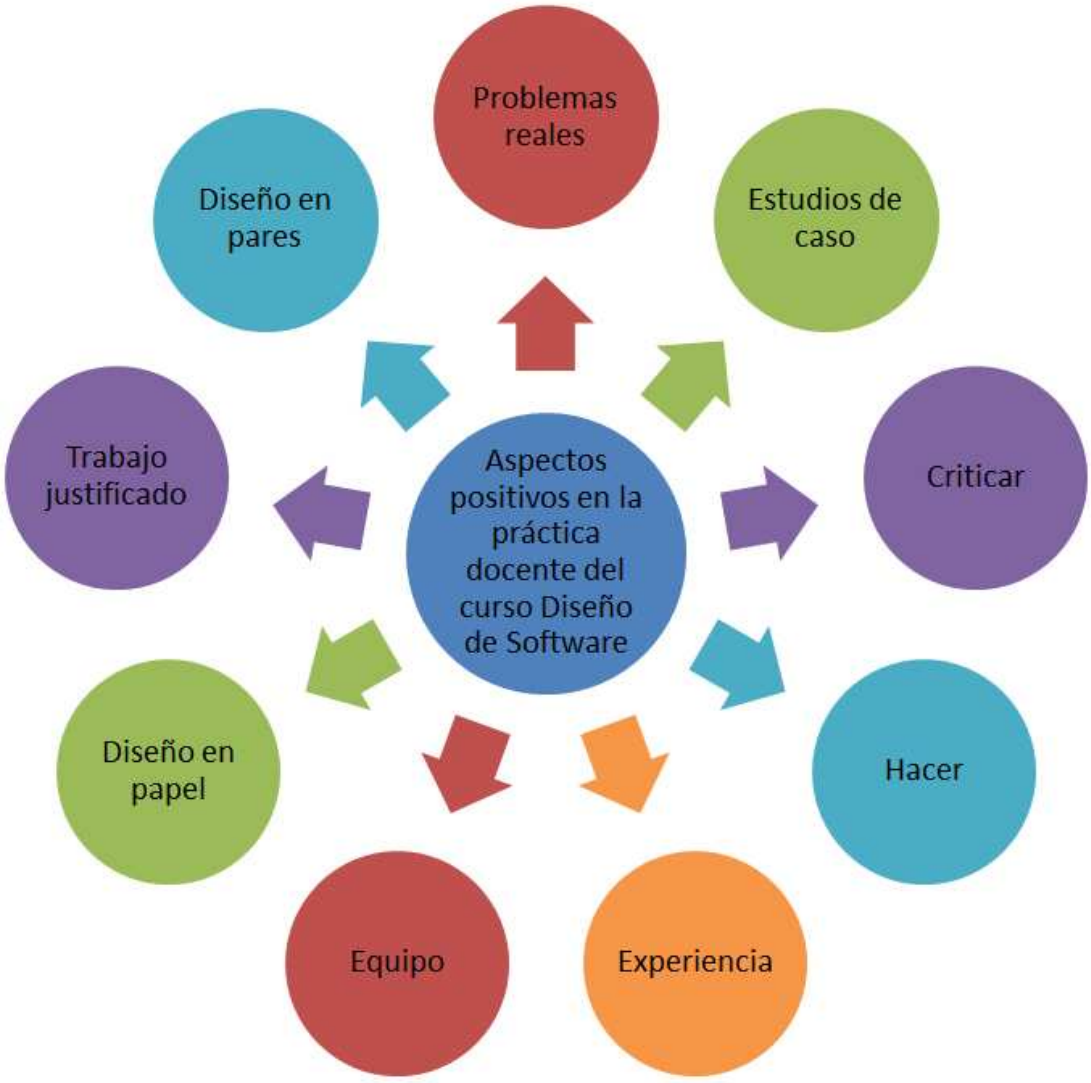


Figura 20 *Cuadro semántico aspectos positivos*
Fuente: Esta investigación.

Figura 21 *Cuadro semántico aspectos negativos*
Fuente: Esta investigación.

Revisión documental Syllabus

Con el objeto de caracterizar los elementos del modelo pedagógico presentes en cada uno de los Syllabus de las universidades participantes en la investigación, se realizó la interpretación a la luz del marco teórico planteado en la tabla 6, los resultados se pueden observar en la tabla 34.

Tabla 34 *Universidades por modelo pedagógico*

Rank	Universidad	Intención	Método	Metas	Concepto de desarrollo	Contenidos	Relación profesor – estudiante	Evaluación
01	MIT	C	P	P	P	C	P	P
02	Universidad de Illinois - Urbana	C	P	-	P	C	P	P
05	Universidad de Texas at Austin	C	C	C	C	C	P	C
06	Universidad Carnegie Mellon	P	P	P	P	C	P	P
10	Universidad de California Santa Barbara	P	P	P	P	C	P	P
14	EPFL - Swiss Inst. Fed. Tecn.	P	P	P	P	C	P	P
17	Colegio Imperia de Londres	P	P	P	P	C	P	P
18	Universidad de Washington	P	P	P	P	C	P	P
31	Universidad de California Davis	P	P	P	P	C	P	P
35	Universidad de Colorado Boulder	P	P	C	P	C	P	P
36	Universidad de McGill	C	P	P	P	C	C	P
40	Universidad Duke	C	P	P	P	C	C	P
59	Universidad de Boston	C	P	P	P	C	C	P
62	Universidad Tecnológica Delft	-	-	-	-	-	-	-
79	Universidad La Sapienza Roma	C	P	P	P	C	C	P
-	Escuela de Ingeniería de Antioquia	P	P	P	P	C	P	P
-	Universidad de Nariño	P	T	P	T	C	P	C
-	Universidad del Cauca	P	C	P	P	C	C	P
-	Universidad San Martin	P	P	P	P	C	P	P

Convenciones			
Modelo pedagógico	Conductista	Progresista	Tradicional
Letra asociada	C	P	T

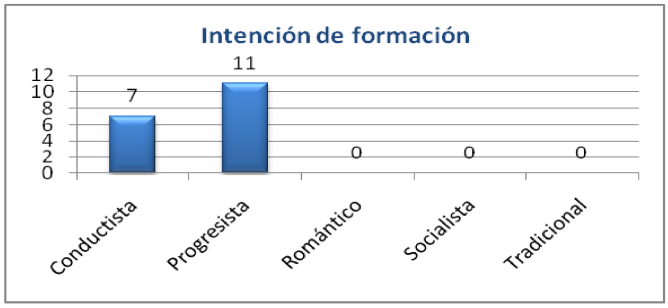


Figura 22 *Intención de formación*
Fuente: Esta investigación.

En la figura 22 se puede apreciar que la intención de la mayoría (61%) de los syllabus se inclinan por el modelo pedagógico progresista que orienta su formación de acuerdo con la necesidad y condición de cada uno (Restrepo Mesa, 2008), en un segundo lugar están quienes prefieren seguir el modelo conductista que instruye y refuerza (Restrepo Mesa, 2008). Los modelos tradicional, romántico y socialista no se encuentran presentes en la intención de formación.

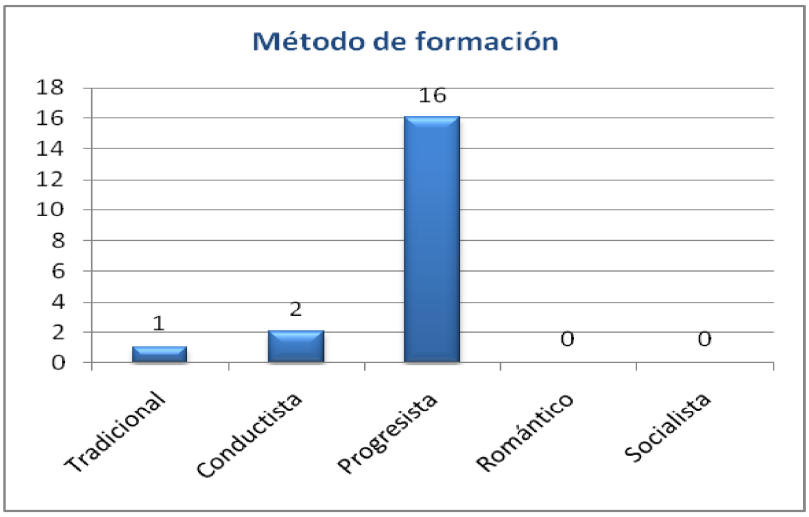


Figura 23 *Método de formación*
Fuente: Esta investigación.

El modelo pedagógico progresista es el preferido (84%) por los docentes porque permite crear ambientes y partir de experiencias del alumno, interacción sujeto – objeto y reflexión científica (Restrepo Mesa, 2008), los modelos conductista y tradicional aparecen con porcentajes mínimos y los modelos romántico y socialista no obtienen puntaje. Los resultados evidencian, dada la naturaleza disciplinar del diseño de software, que el modelo progresista es el pertinente.

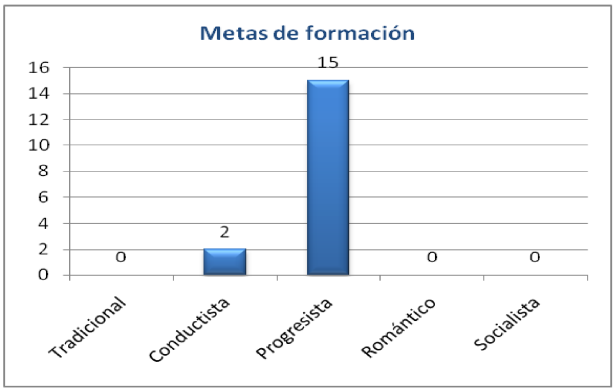


Figura 24 *Metas de formación*
Fuente: Esta investigación.

El modelo progresista, que propicia el acceso a niveles intelectuales superiores (Restrepo Mesa, 2008), es el que predomina (88%) frente al conductista (12%), en lo que hace se refiere a metas de formación. Los modelos tradicional, romántico y socialista no obtuvieron puntaje en este aspecto.

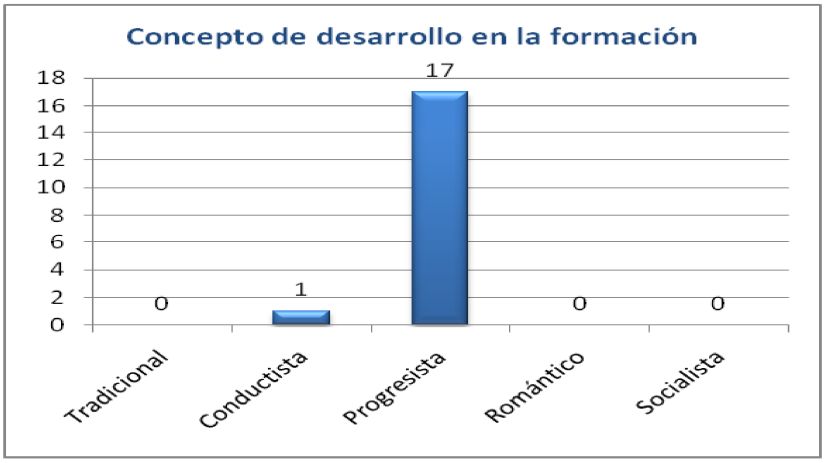


Figura 25 *Concepto de desarrollo en la formación*
Fuente: Esta investigación.

En la mayoría de los syllabus (94%) se puede observar la preferencia del modelo progresista que define el concepto de desarrollo como oportunidad de realización desde la reflexión científica (Restrepo Mesa, 2008), el conductismo tiene apenas un valor del 6%, y los demás modelos no se encuentran presentes.

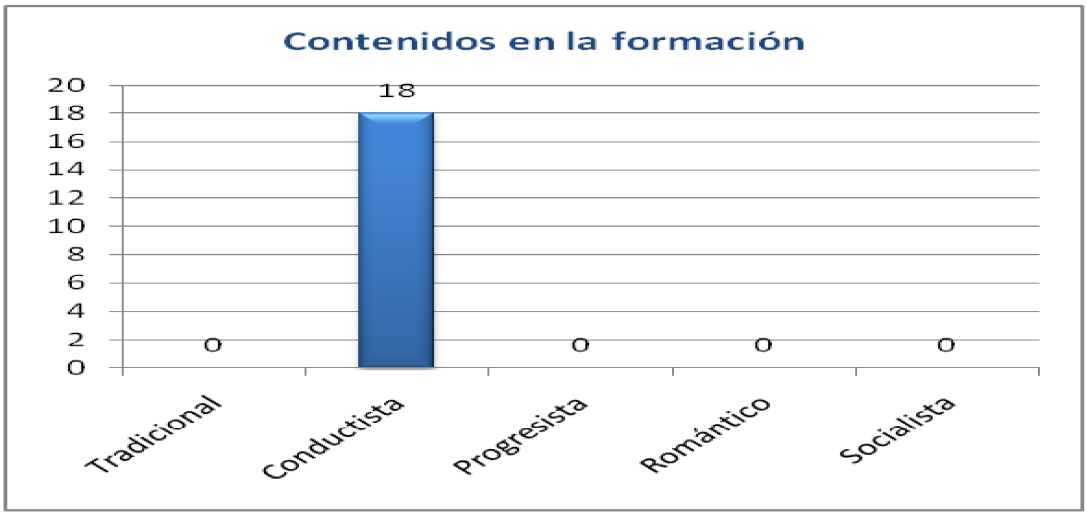


Figura 26 *Contenidos en la formación*
Fuente: Esta investigación.

En cuanto a contenidos el 100% apuesta por el razonamiento lógico, operaciones formales y conocimientos como producto de algo (Restrepo Mesa, 2008), es decir, por el modelo conductista.

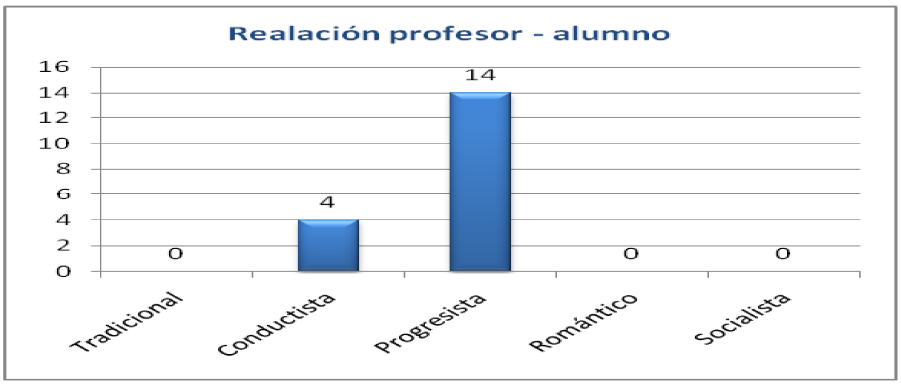


Figura 27 *Relación profesor – alumno*
Fuente: Esta investigación.

La relación entre profesor y alumno en donde el primero es un guía y orientador bidireccional (Restrepo Mesa, 2008) es la que se encuentra en la mayoría (78%) de syllabus de los docentes participantes de la investigación, con un 22% se encuentra el docente como un intermediario, ejecutor y programador. Los modelos tradicional, romántico y socialista no obtuvieron puntaje.

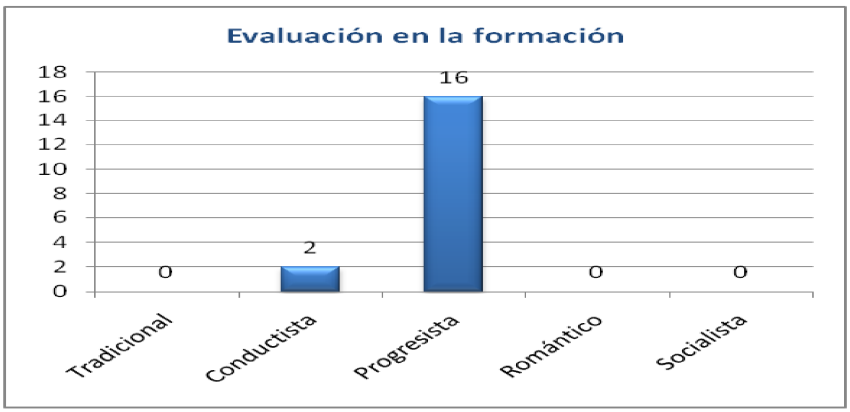


Figura 28 *Evaluación en la formación*
Fuente: Esta investigación.

El modelo progresista que persigue con la evaluación niveles altos de conocimiento (Restrepo Mesa, 2008), es el utilizado por gran parte (89%) de los docentes participantes en la investigación y apenas un 11% son conductistas y prefieren ser conductistas y utilizar la evaluación como estrategia para la diferencia individual de sus estudiantes.

La Didáctica del Diseño de Software propuesta en esta investigación pretende formar profesionales como personas activas capaces de tomar decisiones, emitir juicios y

resolver problemas ante diferentes contextos, pretende también desarrollar las habilidades del pensamiento de los individuos de modo que puedan progresar, evolucionar secuencialmente en las estructuras cognitivas para acceder a conocimientos cada vez más elaborados, esto implica compromiso de los actores del proceso frente a la construcción, creación, facilitación, liberación, interrogación, crítica y reflexión de la comprensión de las estructuras profundas del conocimiento.

2.5 Matriz de características de experiencias de enseñanza de diseño de software

Tabla 35 *Matriz de características de experiencias de enseñanza de diseño de software*

	Características
Enfoque pedagógico	- Las intenciones son progresistas y conductistas. - El modelo progresista es el preferido en cuanto a método, metas, concepto de desarrollo, relación profesor – alumno y evaluación en la formación. - Los contenidos trabajados en el curso de diseño de software en la totalidad de docentes participantes evidencian el modelo conductista.
Objetivos pedagógicos	- La mayoría de docentes hacen explícitos en sus syllabus los objetivos pedagógicos (ver figura 15) - Los objetivos redactados como competencias no son la preferencia, en contraste a los objetivos y metas (ver figura 16)
Contenidos	- UML como herramienta para el desarrollo del curso de diseño de software (ver figura 9)
Actividades de clase	- Inclinación fuerte hacia la utilización en clase de actividades tradicionales como la cátedra magistral. - Escasa utilización de actividades que prefieren al estudiante como actor principal del aprendizaje, como por ejemplo juego de roles (ver figura 10)
Actividades de trabajo independiente	- Preferencia de actividades de trabajo independiente tradicionales como tareas y lecturas complementarias. - Utilización con menor frecuencia de actividades de trabajo independiente que evidencian el trabajo autónomo. - Escasa utilización de actividades y herramientas en línea (ver figura 11)
Recursos educativos	- Utilización de recursos educativos que resaltan la labor transmisora del docente - Utilización en un alto porcentaje de documentación tanto digital como impresa. - Mediana utilización de software educativo y especializado - Escasa utilización de recursos para trabajo en línea y en equipo (ver figura 12)
Espacio físico	- El salón de clase es el lugar para desarrollar el curso de diseño de software - El laboratorio de computadores es utilizado como espacio de clase en un poco más de la mitad de docentes - Escasa utilización de entornos virtuales - No se hace uso de lugares diferentes a los planteados en la encuesta como espacios de clase. (ver figura 13)

	Características
Evaluación	- No se hace uso del portafolio como técnica de evaluación. - El proyecto es la técnica más empleada en la evaluación - El examen no desaparece como forma de evaluación y es utilizado por un alto porcentaje de docentes. - El artículo, taller, solución de problemas, laboratorios y exposiciones son escasamente utilizados como técnica de evaluación del curso. (ver figura 14)
Aspectos positivos en la práctica docente	- Utilización de problemas reales - Realización de estudios de caso - Trabajo en pares - Justificación del trabajo del curso - La crítica como proceso de enriquecimiento - El diseño sin utilizar herramientas computacionales como primera etapa del curso - Trabajos en equipo - La experiencia de hacer como semillero del aprendizaje (ver figura 17)
Aspectos negativos en la práctica docente	- Grupos con niveles académicos heterogéneos - Nivelaciones académicas necesarias antes de empezar el curso - Ineficacia de las conferencias y cátedras magistrales - Falta de experiencia real - Falta de aulas equipadas con software especializado en diseño de software - Tiempo no suficiente de clase - Amplia gama de plataformas disponibles en el medio - Escasa participación estudiantil. - Grupos de clase numerosos. - Libros y clases con desarrollo de temas inútiles. - Escaso trabajo en equipo por parte de los estudiantes. - Evolución vertiginosa de la profesión - El diseño es casi totalmente ignorado - Trabajo artesanal en la construcción de software - Fuerte corriente cultural del código y prueba en la construcción de software - Inmadurez del campo del diseño de software

Fuente: Esta investigación.

3. PROPUESTA: LA DIDÁCTICA DEL DISEÑO DE SOFTWARE

1. INTRODUCCIÓN

La Didáctica del Diseño de Software (DDS) es el resultado del análisis y síntesis usando tres fuentes de información: Profesores nacionales y extranjeros en el campo del diseño de software, dos documentos disciplinares *The Joint Task Force for Computing Curricula 2005* y *SWEBOK Guide 2004*, y la fundamentación teórica sobre aprendizaje activo.
Gracias al experto grupo de profesores, La DDS pudo establecer un e*stado-del-arte* relacionado con la enseñanza del diseño de software, obteniendo las mejores características tales como contenidos, ejercicios, clases, trabajo en equipo, proyectos, evaluaciones, etc. Los autores de la DDS expresan su agradecimiento a:

1. Prof. Daniel Jackson (Massachusetts Institute of Technology - USA)
2. Prof. Ralph Johnson (University of Illinois @ Urbana Champaign – USA)
3. Prof. Dewayne E. Perry (University of Texas @ Austin – USA)
4. Prof. Jonathan Aldrich (Carnegie Mellon University – USA)
5. Prof. Tevfik Bultan (University of California @ Santa Barbara – USA)
6. Prof. George Candea (EPFL – Swiss Federal Institute of Technology)
7. Prof. Alexander L. Wolf (Imperial College London – UK)
8. Prof. David Notkin (University of Washington – USA)
9. Prof. Premkumar T. Devanbu (University of California @ Davis – USA)
10. Prof. Kenneth M. Anderson (University of Colorado @ Boulden – USA)
11. Prof. Joseph Vybihal (McGill University – USA)
12. Prof. Robert Duvall (Duke University – USA)
13. Prof. John Keklak (Boston University – USA)
14. Prof. P.G. Kluit (Delft University of Technology – Holland)
15. Prof. Paolo Bottoni (University of Rome “La Sapienza”)

El documento titulado *The Joint Task Force for Computing Curricula 2005* fué un proyecto colaborativo entre *The Association for Computing Machinery (ACM), The Association for Information Systems (AIS)* y *The Computer Society (IEEE-CS)*. Dicho documento representa una guía para el cuerpo de conocimiento relacionado con computación en ambientes académicos. La DDS extrajo la parte pertinente de ingeniería de software.

Finalmente, El documento *SWEBOK* (*Guide to the Software Engineering Body of Knowledge - 2004*) es llamado a indagar y describir el conocimiento disciplinar de los profesionales en ingeniera de software. Dicho documento del IEEE-CS promueve el avance tanto teórico como práctico en el campo de estudio de la ingeniería de software.

La DDS no pretende ser un estándar de enseñanza. Sin embargo, tiene elementos de referencia educativa los cuales podrían servir como guía en la enseñanza de un curso introductorio de diseño de software a nivel mundial.

La Didáctica de Diseño de Software se estructura en ocho elementos, a saber: Enfoque pedagógico, objetivos pedagógicos, contenidos, actividades de clase, actividades de trabajo independiente, recursos educativos, espacio físico y evaluación, que mantienen relación entre si, como se ilustra a continuación en la figura 29.

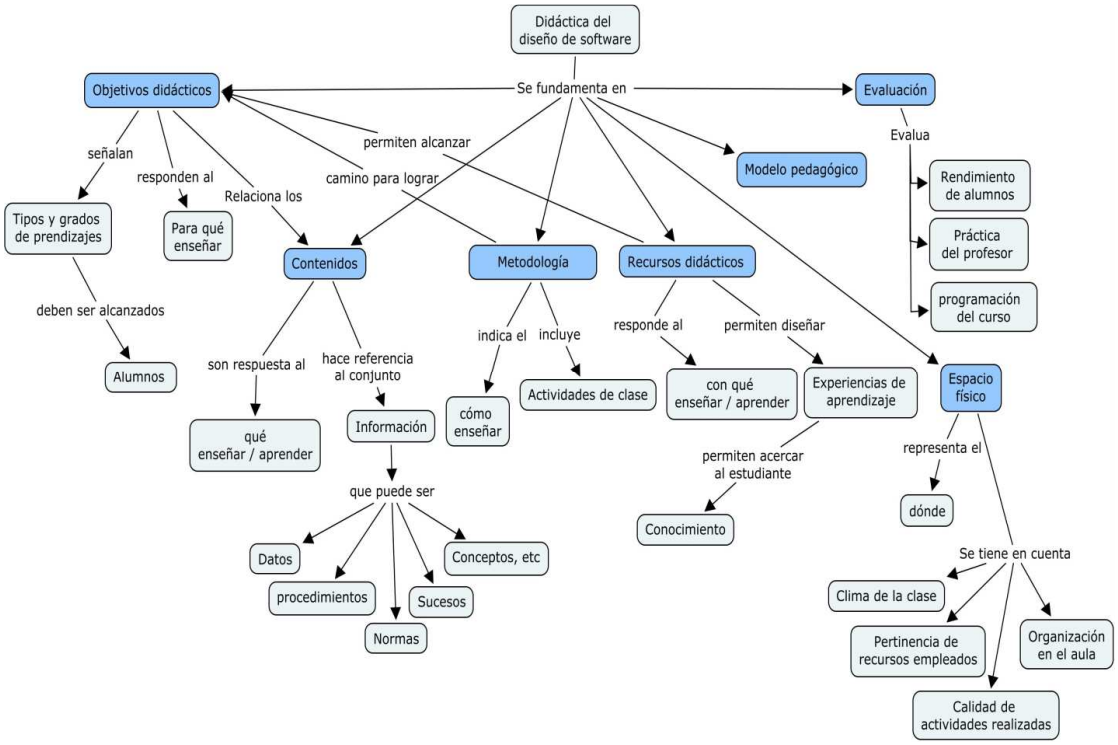


Figura 29 *Mapa conceptual de la Didáctica del Diseño de Software.*
Fuente: Esta investigación.

2. Contexto de aplicación de la didáctica del diseño de software

El curso introductorio de diseño de software fue concebido con el análisis de los contenidos sugeridos por ACM, IEEE-CS, y los syllabus de los profesores extranjeros y nacionales. Estos insumos describen el escenario de pregrado dentro de las carreras con enfoque disciplinar de las ciencias de la computación. Teniendo en cuenta el nivel de formación de pregrado, los elementos que constituyen el contexto de aplicación de la Didáctica del Diseño de Software son, en primer lugar los conocimientos previos del estudiantado son:

- *Fundamentos de paradigmas de programación.*

En el estudio de los paradigmas de diseño de software, es necesario que el estudiante ya haya tenido un grado mínimo de interacción con los lenguajes de programación. Esta situación podría contradecir aparentemente el principio propuesto por Sins-Knight y Upchurch que afirma la importancia de enseñar a diseñar antes que programar, sin embargo, en los planes de estudio basados en créditos (que son el tipo de plan tomado como base de la presente investigación), es una característica común encontrar talleres de programación en los primeros periodos (o semestres) de formación a nivel de pregrado.

- *Programación orientada a objetos.*

Las temáticas asociadas a las actividades clave en el diseño de software tales como: la concurrencia, el control de eventos, manejo de procesos e hilos, el control de excepciones y la persistencia de datos forman parte del paradigma de programación orientado a objetos. Esto implica que la formación preliminar del estudiante en cuanto a paradigmas de programación deberá tener un énfasis especial en el paradigma orientado a objetos.

3. Ubicación dentro del plan de estudios.

ACM, IEEE-CS y AIM a través de su documento Computing Curricula 2005, proponen tan solo una recomendación guía que orienta la construcción de planes de estudio a nivel global. Como las características culturales son diferentes en cada parte del planeta, así como la libertad de desarrollo curricular del cual gozan cada contexto, es difícil precisar el lugar exacto del curso introductorio de diseño de software en el plan de estudios. Sin embargo, teniendo en cuenta sus contenidos y los conocimientos previos requeridos, es preciso que el curso introductorio de diseño de software esté ubicado a comienzos de la segunda mitad de un plan de estudios dado.

4. Contexto de validación de la didáctica de diseño de software

Al revisar los objetivos planteados en la presente investigación, la validación de la didáctica no fue contemplada como parte del proyecto. No obstante, un proyecto interesante que ha sido plasmado en las recomendaciones de la presente investigación podría formularse con los siguientes criterios:

1. Establecimiento del grupo experimental y el grupo control.

2. Aplicación de la estrategia siempre y cuando se hayan cumplido los conocimientos previos.

3. Análisis de los resultados de las evaluaciones teniendo en cuenta ambientes reales de aplicación.

5. Enfoque pedagógico

La didáctica propuesta sigue los principios generales del aprendizaje activo. En este sentido, tanto las actividades presenciales (dentro del desarrollo de la cátedra) como los trabajos independientes y las actividades de evaluación buscan la acción permanente del estudiante. Esta constante práctica de exploración empírica del conocimiento, fortalece las relaciones cognitivas del estudiante con su objeto de estudio (Silberman, 1998). Según Mel Silberman, la interacción constante con entornos de aprendizaje activo permitirá al estudiante cambiar sus hábitos de cómo enfrentar los desafíos en ambientes académicos, generando así una cultura familiarizada de aprendizaje.

La DDS es un constructo teórico basado en el enfoque de aprendizaje activo. Una estrategia de aprendizaje activo promueve una actitud activa en los estudiantes. Dicho enfoque tiene tres características principales:

1. *Los estudiantes tienen mejores niveles de atención.*

Una clase típica suele producir un decremento de la atención estudiantil pasados 15 minutos; la atención podría elevarse al final debido a la conclusión de la clase. Una serie de pequeñas introducciones cada 15 o 20 minutos ayudarían a los estudiantes a mantener su atención.

2. *Fácil apropiación de conocimientos.*

Los estudiantes apropian el conocimiento de mejor forma si ellos interactúan con él. Si es menos probable que un estudiante interactúe con el conocimiento una vez la clase finaliza, entonces es muy importante mantener a los estudiantes activos dentro de la clase.

3. *Fácil conocimiento de la retroalimentación del aprendizaje.*

Al final de la sesión de clase, el profesor y los estudiantes tienen suficientes elementos para evaluar el grado de entendimiento del tema tratado.

De acuerdo con el enfoque de aprendizaje activo, algunos beneficios son explícitos en el desarrollo de las sesiones de clase (Charles Bonwell & James Eison, 1991), estos beneficios incluyen:

1. Los estudiantes se involucran más en la clase que como solo oyentes.
2. Menos énfasis en la transmisión de la información y mayor énfasis en el desarrollo de

las habilidades del estudiante (enfoque por competencias).
3. Los estudiantes están involucrados en estructuras de pensamiento de orden superior (análisis, síntesis y evaluación del conocimiento).
4. Se exige a los estudiantes el desarrollo de actividades de clase (ej. lecturas, discusiones, y propuestas escritas).
5. Un énfasis mayor en la determinación de la auto-exploración de los estudiantes en cuanto a sus propias actitudes y valores.

6. Objetivos pedagógicos

Al final del curso de diseño de software los estudiantes serán capaces de: (ACM, IEEE-CS & AIS, 2005)

1. Discutir las propiedades de un buen diseño de software incluyendo la naturaleza y el rol de la documentación adjuntada.
2. Evaluar la calidad de múltiples diseños de software basados en principios y conceptos clave.
3. Seleccionar y aplicar patrones apropiados de diseño en la construcción de una aplicación software.
4. Crear y especificar el diseño de software para un producto software de mediana escala usando una especificación de requerimientos, una metodología de diseño de programas aceptada (ej. metodología funcional u orientada a objetos, etc.), y la apropiada notación del diseño.
5. Evaluar un diseño de software al nivel de componentes.
6. Evaluar un diseño de software desde la perspectiva de la reutilización y escalabilidad.

7. Contenidos de Aprendizaje

Los contenidos de la Didáctica del Diseño de Software se organizaron en siete unidades acorde con los estándares de ACM e IEEE y se establecieron los tiempos, en horas de trabajo, de cada tema, la figura 30 muestra en detalle cada uno de los temas que hacen parte de cada una de las unidades de contenido del curso y la tabla 36 resume los tiempos:

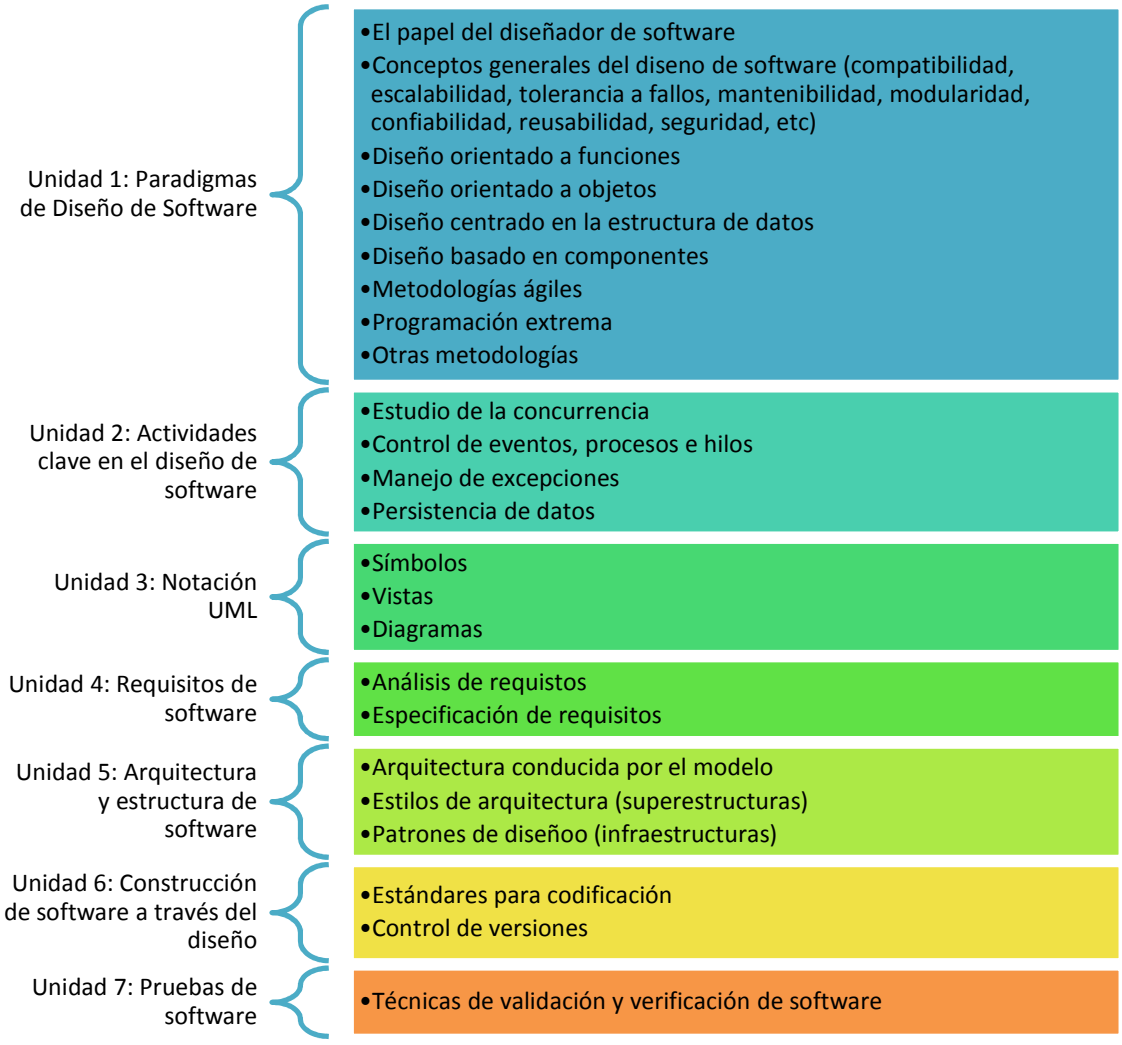


Figura 30 *Contenidos de aprendizaje*
Fuente: Esta investigación.

Tabla 36 *Asignación de tiempos presencial e independiente de los contenidos del curso de diseño de software.*

Unidad	**Contenido**	**Descripción**	**HC**[1]	**HI**
Paradigmas de diseño de software	El papel del diseñador de software	Muestra la importancia del diseñador enfrentando problemas reales. Deberes y responsabilidades, habilidades sociales y profesionales.	2	2
	Conceptos Generales de diseño de software	Este contenido está diseñado para incluir los fundamentos de las ciencias de la computación los cuales soportan el diseño y la construcción de software. Esta área también incluye el conocimiento acerca de la transformación del diseño en implementación real, las herramientas usadas durante el este proceso, y los métodos de construcción formal de software.	2	2
	Diseño orientado a funciones	Centros de descomposición para la identificación de funcionalidades del sistema para su refinamiento. En gran forma, el diseño estructurado es generalmente usado después del análisis, los diagramas de flujos de datos y las descripciones de procesos asociados.	2	2
	Diseño orientado a objetos	Muchos métodos de diseño de software basados en objetos han sido propuestos donde la herencia y el polimorfismo juegan un papel clave, hacia el diseño basado en componentes, donde la meta información puede ser definida y consultada. El diseño orientado a objetos tiene sus raíces en el concepto de abstracción de datos, el diseño conducido por responsabilidades, la escalabilidad y la reutilización.	8	8
	Diseño centrado en la estructura de datos	Inicia desde la definición de la estructura de datos. Para diseñar, el primer paso es la declaración de estructuras de datos para entrada y salida y luego el desarrollo de las estructuras de control para manipular la información basada en los diagramas.	2	2
	Diseño basad en componentes	Un componente de software es una unidad independiente, con interfaces bien definidas y dependencias que pueden ser compuestas y desplegadas en forma separada. El diseño basado en componentes direcciona actividades relacionadas a la reorganización e integración con fines de reutilización del software.	4	4

[1] HC horas de trabajo presencial / HI horas de trabajo independiente dedicado a los contenidos pertinentes en un curso de diseño de software

	Metodologías agiles.	El desarrollo de software ágil es un conjunto de metodologías de desarrollo basada en la iteración y el desarrollo incremental, donde los requisitos y soluciones se entrelazan a través de actividades de colaboración y organización empresarial.	2	2
	Programación extrema	La programación extrema es una disciplina de desarrollo de software basada en valores de simplicidad, comunicación, retroactividad, coraje y pasión por el desarrollo. Esta metodología funciona en ambientes de colaboración para el desarrollo de software con prácticas simples y vinculación de historias de usuarios.	2	2
	Otras metodologías	Una revisión a nuevas propuestas en metodologías de diseño de software que pueden ser encontradas como tesis de maestría o doctorales.	2	2
Actividades claves en el diseño de software	Estudio de la concurrencia	Los estados y los diagramas de máquinas de estado; estructuras; intercambios de contexto de los procesos; el papel de las interrupciones; ejecución concurrente; problemas mutualmente excluyentes y sus soluciones; modelos y mecanismos de sincronización.	2	2
	Control de eventos, procesos e hilos	La descomposición del software en procesos, eventos e hilos, y tratar con eficiencia relacionada, atomicidad, sincronización, y actividades programadas. El control y el manejo de eventos permiten organizar el control del flujo de información, el manejo de eventos reactivos y temporales a través de varios mecanismos como la invocación implícita entre otros.	2	2
	Manejo de excepciones	Prevención y tolerancia a fallos y enfrentar condiciones excepcionales. Control de errores en tiempo de ejecución.	2	2
	Persistencia de datos	Administración y control de datos de larga vida útil. Frecuentemente, la persistencia de datos necesita un contenedor de información que por lo general son las bases de datos.	4	4
Notación UML	Símbolos	Entender la notación UML acerca del diseño de software	2	2
	Vistas	Trabajar con vistas de diseño (estructura y comportamiento), vista de implementación (componentes, ensamblaje del sistema, administración de la configuración), vista de casos de uso (funcionalidades, servicios del sistema, actores), vista de procesos (concurrencia, hilos, desempeño) y vista de despliegue (topología del sistema, distribución, despliegue e instalación).	2	2

	Diagramas	Trabajar con casos de uso, clases, objetos, interacciones, actividades, despliegue, estados, secuencias, paquetes, y componentes.	4	4
Requisitos de software	Análisis de requisitos	El análisis de requisitos incluye su clasificación, modelado conceptual, diseño arquitectónico y ubicación, así como la negociación de requisitos.	2	2
	Especificación de requisitos	La especificación de requisitos típicamente se refiere a la producción de un documento o su equivalente electrónico, que puede ser revisado, evaluado, y aprobado. Para sistemas complejos, se producen 3 tipos de documentos: la definición del sistema, la especificación de requerimientos del sistema, y la especificación de requerimientos del hardware.	2	2
Software arquitectura y estructura	Arquitectura conducida por el modelo	Tópicos sobre estructuras de arquitecturas y puntos de vista, estilos arquitecturales, patrones de diseño, y finalmente, familias de programas y marcos de trabajo.	6	6
	Estilos de arquitecturas	Un estilo de arquitectura es un conjunto de restricciones y arquitecturas que definen una familia de arquitecturas. Un estilo de arquitectura puede ser visto como una meta modelo el cual provee una organización de alto nivel de arquitectura (macro arquitectura).	2	2
	Patrones de diseño	Un patrón es una solución común a un problema común en un contexto dado. Mientras los estilos de arquitectura pueden ser vistos como patrones que describen entidades de software que de alto nivel (macro arquitecturas), los patrones de diseño pueden ser usados para describir a bajo nivel los componentes del software (micro arquitecturas).	2	2
Construcción de software a través del diseño	Estándares de codificación	Organización de código fuente (en sentencias, rutinas, clases, paquetes, u otras estructuras), documentación del código fuente y afinamiento.	2	2
	Control de versiones	El soporte de evolución requiere numerosas actividades antes y después de la liberación del sistema, Control de actualizaciones que conforman un sistema evolutivo.	2	2

Pruebas de software	Técnicas de validación y verificación de software	Usa tanto técnicas estáticas como dinámicas sobre la verificación del software para asegurar que los resultados del producto satisfagan las necesidades de los clientes.	2	2
		TOTAL:	64	64

Fuente: Esta investigación.

El trabajo total del estudiante es de 128 horas por periodo académico regular de 16 semanas discriminadas en trabajo presencial de 64 horas, es decir, se trabaja por semana cuatro horas. Se debe entender trabajo presencial como aquella labor que el alumno desarrolla en compañía del profesor. El trabajo independiente del estudiante es de 64 horas por periodo académico y debe entenderse por dicho trabajo como la labor que realiza el alumno sin acompañamiento del profesor.

8. Actividades de Clase

Tabla 37 *Duración de las estrategias de clase según sus contendido.*

#	Contenido	Estrategias de clases	Duración (horas)
1	El papel del diseñador de software	- Clase activa - Juego de rol	1 1
2	Conceptos generales de diseño de software (compatibilidad, escalabilidad, tolerancia a fallos, mantenibilidad, modularidad, confiabilidad, reutilización, seguridad, etc.)	- Clase activa - Sesión de preguntas	1 1
3	Diseño orientado a funciones	- Clase activa - Desarrollo de ejercicios	1 1
4	Diseño orientado a objetos	- Clase activa - Desarrollo de ejercicios - Proyecto de aula	2 2 4
5	Diseño centrado en la estructura de datos	- Clase activa - Desarrollo de ejercicios	1 1
6	Diseño basado en componentes	- Clase activa - Desarrollo de ejercicios - Proyecto de aula	1 1 2
7	Metodologías ágiles	- Clase activa - Desarrollo de ejercicios	1 1
8	Programación extrema	- Clase activa - Desarrollo de ejercicios	1 1
9	Otras metodologías	- Clase activa - Sesión de preguntas	1 1
10	Estudio de la concurrencia	- Clase activa - Desarrollo de ejercicios	1 1
11	Control de eventos, procesos e hilos	- Clase activa - Desarrollo de ejercicios	1 1
12	Manejo de excepciones	- Clase activa - Desarrollo de ejercicios	1 1
13	Persistencia de datos	- Clase activa - Desarrollo de ejercicios	2 2
14	Símbolos	- Clase activa - Sesión de preguntas	1 1
15	Vistas	- Clase activa - Sesión de preguntas	1 1
16	Diagramas	- Clase activa - Proyecto de aula	2 2
17	Análisis de requisitos	- Clase activa - Desarrollo de ejercicios	1 1
18	Especificación de requisitos	- Clase activa - Desarrollo de ejercicios	1 1
19	Arquitectura conducida por el modelo	- Clase activa	2

#	Contenido	Estrategias de clases	Duración (horas)
		- Proyecto de aula	4
20	Estilos de arquitectura	- Clase activa - Proyecto de aula	1 1
21	Patrones de diseño	- Clase activa - Proyecto de aula	1 1
22	Estándares de codificación	- Clase activa - Proyecto de aula	1 1
23	Control de versiones	- Clase activa - Proyecto de aula	1 1
24	Técnicas para la validación y verificación del software	- Clase activa - Proyecto de aula	1 1

Fuente: Esta investigación

9. Acerca de las estrategias de clase

Tabla 38 *Acerca de las estrategias de clase y su duración.*

Estrategia	Descripción	Tiempo[2]
Clase activa	Una clase activa es considerada como una exposición oral: un discurso sobre una temática especifica frente a una audiencia. Adicionalmente, es una sesión de enseñanza.	28
Juego de rol	Los jugadores asumen diferentes roles de los personajes en la puesta en escena ficticia del juego. Los estudiantes asumen responsabilidades para representar dichos roles mostrando el comportamiento común de un diseñador de software.	1
Sesión de preguntas	Unas cuantas pruebas sobre terminología, reconocimiento simbólico asociaciones funcionales.	4
Desarrollo de ejercicios	Un espacio en clase para resolver problemas usando conceptos específicos. El desarrollo de ejercicios les permite a los estudiantes aplicar sus conocimientos en contextos reales.	14
Proyecto de aula	Para este curso, se requiere un solo proyecto de aula. El proyecto de aula se propone al inicio del curso, y deberá ser desarrollado en equipos durante la totalidad del curso. (Entre 3 y 5 estudiantes).	17
	Total:	64

Fuente: Esta investigación.

[2] Tiempo presencial total en el aula (en horas)

10. Actividades independientes

Tabla 39 *Acerca de las actividades independientes y su duración*

Estrategia	Descripción	Tiempo[3]
Tareas	Las tareas se refieren a actividades asignadas para ser completadas fuera de clase. Generalmente se otorga un periodo de tiempo considerable para hacerlas y algunas lecturas de soporte, suele estar acompañadas de producción escrita. Esta estrategia debe ser evaluada.	16
Lecturas complementarias	Para soportar los conceptos vistos en clase, las lecturas complementarias son necesarias como elementos para profundización del conocimiento. Generalmente, no es evaluada; sin embargo, es preferible valorar las tareas o trabajos que se requieran apoyar en lecturas complementarias.	6
Proyecto de Investigación	Durante la totalidad del curso, Los estudiantes deberán desarrollar un proyecto de investigación cuyas líneas guías son tomadas del proyecto de aula. Este proyecto podría ser desarrollado en equipos de trabajo entre 3 y 5 estudiantes. Esta actividad debe ser evaluada.	32
Desarrollo de ejercicios	Así como las actividades de aula, el desarrollo de ejercicios les permite a los estudiantes aplicar su conocimiento para resolver problemas. Esta estrategia deberá ser evaluada.	10
	Total:	64

Fuente: Esta investigación.

11. Recursos Educativos y Formatos

Con el uso de proyectores, tableros, documentación digital e impresa, algunas actividades podrían enriquecer la dinámica de la clase. Además, estos recursos educativos son organizados a través de los siguientes formatos para la clase activa, la sesión de preguntas, el desarrollo de ejercicios, el proyecto de investigación y aula.

Estos formatos son desarrollados por el profesor (*) y los estudiantes (**), y todos estos documentos conforman el portafolio del curso como una bitácora relacionada con los procesos de enseñanza y aprendizaje.

El portafolio evidencia las actividades de aula y los trabajos independientes, a continuación se presentan los formatos:

[3] Tiempo total independiente (en horas)

CLASE ACTIVA (Formulario de Planeación)	
Título de la Clase Activa: *[4]	
Objetivos de Aprendizaje: * …	
Duración de la Clase Activa: *	
Actividades de Motivación/Recursos: * …	**Duración:*** …
Contenidos:* …	
Lecturas Complementarias:* - <<Código>> …	
Sesión de Preguntas:* - <<Código>> …	
Desarrollo de Ejercicios:* - <<Código>> …	

<<Código>> - LECTURAS COMPLEMENTARIAS (Formulario de Planeación)
Nombre del Estudiante:**[5]
Título de la Lectura: *
Fuente (Disponible en):*
Palabra Claves:** …
Idea Principal:**

[4] Actividad del profesor

[5] Actividad del estudiante

Mapa Conceptual de la Idea Principal:**

<<Código>> - SESIÓN DE PREGUNTAS (Formulario de Planeación)	
Nombre del Estudiante:**	
Título de la Clase Activa: *	
Preguntas Abiertas:* - …	**Respuestas:**** - …
Preguntas Cerradas:* - …	**Respuestas:**** -[ ] Verdadero [ ] Falso -[ ] Opción 1…. …

<<Código>> - DESARROLLO DE EJERCICIOS (Formulario de Planeación)	
Nombre el Estudiante:**	
Título de la Clase Activa: *	
Ejercicios:* -	
Escala de valoración subjetiva:** **Criterio:** …	 **Valor:** …

<<Código>> - PROYECTO DE AULA/INVESTIGACION (Formulario de Planeación)
Nombre del Equipo/ Nombre de los Estudiantes:**
Título de la Clase Activa: *
Conceptos Aplicados:** …
Fecha y Fase:**
Competencias de la fase:*

…
Productos esperados:* …

En general, los profesores podrían usar estos formatos para organizar su clase activa junto con las actividades independientes. La asociación entre los formatos se muestra a continuación en el siguiente esquema:

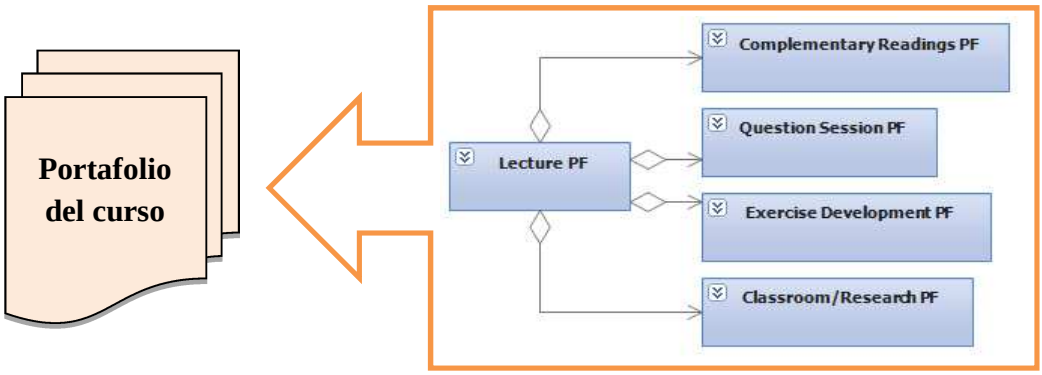
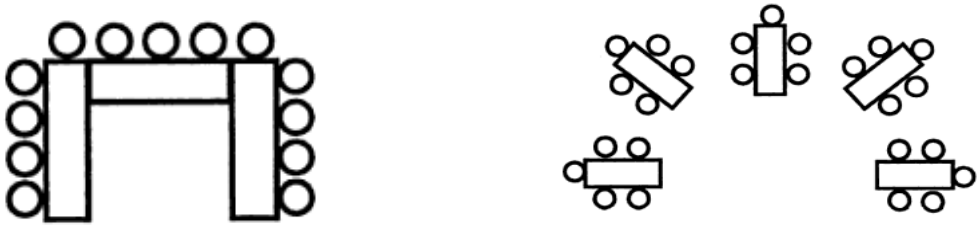


Figura 31 *Detalle de la asociación entre formatos.*
Fuente: Esta investigación.

12. Espacio Físico

Es recomendable usar un espacio con la siguiente distribución física con el propósito de atender las Clases Activas y sus actividades internas tales como el desarrollo de ejercicios, la sesión de preguntas, y el proyecto de aula.



Sesión de Clase Activa

Equipos de Trabajo:

- Desarrollo de ejercicios
- Proyecto de Aula
- Sesión de Preguntas

Figura 32 *Distribución física para atender las clases activas*
Fuente: Esta investigación.

13. Evaluación

Esta es la lista de actividades que pueden utilizarse para evaluar:

- Proyectos
- Pruebas escritas grandes (Test)
- Pruebas escritas pequeñas (Quiz)
- Informes

Los profesores deberán determinar los porcentajes de los instrumentos de evaluación.

CONCLUSIONES

El modelo conductista es el que se encuentra en los contenidos, tanto en las experiencias de práctica docente como en los syllabus de los profesores colaboradores de la investigación. El modelo estructuralista o progresista fue encontrado en los demás elementos pedagógicos.

La mayoría de docentes hacen explícitos sus objetivos pedagógicos y prefieren 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 colectivamente.

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 de la investigación.

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.

La didáctica del diseño de software fue creada con el propósito de aportar al campo de la enseñanza de dicho conocimiento y puede ser puesta en práctica en programas profesionales, técnicos y tecnológicos de instituciones tanto públicas como privadas, que incluyan en la formación cursos de fabricación de productos software independientemente del período académico dentro del pensum en que se incluyan.

Acoger la Didáctica del Diseño de Software permitirá a los docentes reflexionar sobre su quehacer en este campo, al tener como referente la compilación de las mejores prácticas de enseñanza desde una fundamentación pedagógica.

RECOMENDACIONES

Resultaría interesante se avance en los contenidos del curso diseño de software dando pasos hacia modelos pedagógicos como el socialista, que tienen como prioridad la naturaleza y vida social.

En el mundo globalizado en que vivimos, los futuros profesionales deberían ser capaces de desenvolverse con su saber en cualquier contexto y de forma colaborativa, por tanto debería pensarse en la posibilidad de perfilar la didáctica aquí propuesta hacia el desarrollo de competencias y trabajo en equipo en todos sus elementos.

Contrastar los excelentes resultados de las prácticas docentes internacionales con sus metodologías ortodoxas de enseñanza, aportaría de manera significativa al campo de la didáctica.

A futuro, implementar en la Didáctica del Diseño de Software la evaluación siguiendo el modelo pedagógico socialista, dará luces en el proceso de avance en la didáctica aquí propuesta.

Aplicar la Didáctica del Diseño de Software propuesta en la universidad y estudiar los resultados obtenidos puede convertirse en un trabajo interesante de investigación y contribuiría a mejorarla.

La Didáctica del Diseño de Software es conveniente adoptarla porque, en primer lugar, fue construida a partir de referentes de prácticas docentes a nivel nacional e internacional y contiene temáticas globalmente establecidas en el campo disciplinar; por otra parte es una estrategia organizada que incluye un conjunto de técnicas tanto de trabajo presencial como de trabajo independiente que permiten al estudiante ser autónomo y autor regular su actividad, es decir ser activo y protagonista de su aprendizaje; y al docente ser creativo y satisfacer a muchos más estudiantes a través de la retroalimentación inmediata.

BIBLIOGRAFÍA

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.

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.

Ahamed, S. (2003). Experiences in Teaching an Object-Oriented Design and Data Structure Course. *International Conference on Information Technology: Computers and Communications*, 5.

Barrado, C., Bofill, P., Díaz de Cerio, L., Herrero, J. R., Morancho, E., Navarro, L., y otros. (2001). Siete Experiencias de Aprendizaje Activo. 20.

Barzilay, O., Hazzan, O., & Yehudai, A. (2009). A Multidimensional Software Engineering Course. *IEEE Transactions on Education*, 12.

Bolívar, A. (1995). *El conocimiento de la enseñanza.* Granada: FORCE.

Bonwell, C. C. (s.f.). Active Learning Workshops.

Brignall, T., & Ramaswamy, S. (2003). Impact of Different Teaching Paradigms on Student Learning in Technically Intensive Courses: Observations from a Software Analysis and Design Course. *International Conferenceon Information Technology: Computers and Communications*, 5.

Camilloni, A. (1995). *Reflexiones para la construcción de una didáctica en la educación superior.* Buenos Aires: Universidad Católica de Valparaíso.

Cowling, A. (2007). Stages in Teaching Software Design. *20th Conference on Software Engineering Education and Training*, 8.

de Camilloni, A., Davini, M. C., Edelstein, G., Litwin, E., Souto, M., & Barco, S. (2001). Corrientes didácticas contemporáneas. En M. Souto, *Corrientes didácticas contemporáneas* (pág. 167). Buenos Aires: Paidós.

Diaz Alcaraz, F. (2002). *Didáctica y curriculo. Un enfoque constructivista.* Ediciones de la Universidad de Castilla-La Mancha.

Díaz Barriga , A. (1997). *Didáctica y currículum.* Paidós.

Escribano Gonzales, A. (2004). *Aprender a enseñar. Fundamentos de didáctica general.* Ediciones de la Universidad de Castilla la Mancha .

Estebaranz Garcia, A. (1994). *Didáctica e innovación curricular.* Sevilla: Publicaciones de la Universidad de Sevilla.

Garg, K., & Varma, V. (2007). A Study of the Effectiveness of Case Study approach in Software Engineering Education. *20th Conference on Software Engineering Education and Training*, 8.

Halling, M., Zuser, W., Köhle, M., & Biffl, S. (2002). Teaching the Unified Process to Undergraduate Students. *15th Conference on Software Engineering Education and Training*, 12.

Huang, S., & Distante, D. (2006). On Practice-Oriented Software Engineering Education. *19th Conference on Software Engineering Education and Training - Workshops.*

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

Imídeo G. , N. (1990). *Hacia una didáctica general dinámica.* Buenos Aires: Kapelusz.

Jalote, P. (2009). Teaching and Introductory Software Engineering Course in a Computer Science Program. *22nd Conference on Software Engineering Education and Training*, 1.

Jarzabek, S., & Eng, P.-K. (2005). Teaching an Advanced Design, Team-oriented Software Project Course. *18th Conference on Software Engineering Education and Training*, 8.

Jia, Y., & Tao, Y. (2009). Teaching Software Design Using a Case Study on Model Transformation. *6th International Conference on Information Technology: New Generations*, 5.

Jiménez Toledo, R., Martínez Navarro, Á., & Insuasty, J. (2010). *Didáctica del Diseño de Software.* Pasto: Universidad de Nariño.

Karam, O., Qian, K., & Diaz-Herrera, J. (2004). A Model of SWE Course: Software Architecture and Design. *34th ASEE/IEEE Frontiers in Education Conference*, 5.

Katz, E. (2010). Software Engineering Practicum Course Experience. *23rd IEEE Conference of Software Engineering Education*, 4.

Lago, P., & Van Vliet, H. (2006). Teaching a Course on Software Architecture. *18th Conference on Software Engineering Education and Training*, 8.

Mallart, J. N. (2000). *Capítulo 1. Didáctica: Concepto, objeto y finalidades.* Recuperado el 7 de Abril de 2010.

Männistö, T., Savolainen, J., & Myllärniemi, V. (2008). Teaching Software Architecture Design. *7th Working IEEE/IFIP Conference on Software Architecture*, 8.

Matos, A. (1963). *Compendio de Didáctica General.* Buenos Aires: Kapelusz.

Medina Rivilla, A., & Salvador Mata, F. (2002). *Didáctica general.* Madrid España: Pearson education, S.A.

Mishra, A., Cagiltay, N., & Kilic, O. (2007). Software Engineering Education: Some Important Dimensions. *European Journal of Engineeing Education (Vol 32, No 3)*, 15.

Naveda, J. F. (1999). Teaching Architectural Design in an Undergraduate Software Engineering Curriculum. *29th ASEE/IEEE Frontiers in Education Conference*, 4.

Oscar Sáenz , B. (1992). *Didáctica general. Un enfoque curricular.* Alcoy: Marfil.

Picado , G. F. (2006). *Didáctica general. Una perspectiva integradora.* EUNED: San José, Costa Rica.

Qui, M., & Chen, L. (2010). A Problem-based Learning Approach to Teaching an Advanced Software Engineering Course. *2nd International Workshop on Education Technology and Computer Science*, 4.

Restrepo Mesa, M. C. (2008). *Producción de textos educativos.* Bogotá.

Ross Sowell, C. G. (s.f.). The Active-Learning Transformation: A Case Study In Software Development And Systems Software Courses.

Roy, G. G., & Veraart, V. E. (1996). Software Engineering Education: From an Engineering Perspective. *Murdoch University Journal*, 7.

Ruiz Ruiz, J. M. (1996). *Teoría del currículum: diseño y desarrollo curricular.* Madrid: Universitas.

Silberman, M. (1998). *Aprendizaje activo, 101 estrategias para enseñar cualquier tema.* Argentina: Troquel S.A.

Sins-Knight, J., & Upchurch, R. (2001). Teaching software design: a new approach to computer science. *National Science Foundation*, 9.

Steiman, J. (2008). *Más didáctica. En la educación superior.* Buenos Aires · Argentina: UNSAMedita de Universidad Nacional de General San Martín.

UCIENCIA. (2006). II Simposio de formación del Ingeniero Informático y uso de las TICS en la educación. *Universidad de las Ciencias Informáticas*, 25.

Usman, M., Khan, J., Hardas, M., & Ikram, N. (2010). Pedagogical and Structural Analysis of SE Course. *23rd IEEE Conference of Software Engineering Education and Training*, 8.

vera, M. C. (2000). *La didáctica en cuestión, Investigación y enseñanza.* Rio de Janeiro: Narcea, S. A. de ediciones madrid.

Villalobos, J., & Vela, M. (2007). CUPI2-An Active Learning and Problem Based Learning Approach to Teaching Programming. *8th ALE International Workshop*, 8.

Wals, S. (2010). *Conocimientos didácticos para docentes no pedagogos.* México: Instituto Politécnico Nacional .

Wei, C.-S., Chen, Y., & Doong, J.-G. (2009). A 3D Virtual World Teaching and Learning Platform for Computer Science Courses in Second Life. *China University Journal*, 4.

Yang, C., & Liu, Y. (2009). Teaching Reform and Practice on the Software Design Engineering Course. *The 1st International Conference on Information Science and Engineering (ICISE2009)*, 4.

Zabalza, M. (1990). *"Fundamentación de la Didáctica y del conocimiento.* Madrid: UNED.

Zhenrong, D., Wenming, H., & Rongsheng, D. (2009). Discussion of ability cultivation of computational thinking in course teaching. *International Conference on Eduaction Technology and Computer*, 4.

Zhenrong, D., Wenming, H., & Rongsheng, D. (2009). Programming Courses Teaching Method for Ability Enhancement of Computational Thinking. *International Association of Computer Science and Information Technology - Spring Conference*, 4.

ANEXOS

Anexo A.

Lista de las 80 universidades internacionales convocadas.

Posición	Institución	País	Puntaje
1	Instituto Tecnológico de Massachusetts		100.0
2	Universidad de Stanford		92.5
3	Universidad de Illinois at Urbana-Champaign		85.8
4	Universidad de California - Berkeley		83.6
5	Universidad Carnegie Mellon		81.1
6	Universidad de Michigan - Ann Arbor		81.0
7	Universidad de Texas - Austin		79.8
8	Instituto Tecnológico de Georgia		79.7
9	Universidad de California - San Diego		78.9
10	Universidad Estatal de Pennsylvania		78.8
11	Universidad del Sur de California		77.4
12	Instituto Tecnológico de California		75.9
13	Universidad de California - Santa Bárbara		75.6
14	Universidad de Maryland		75.3
15	Instituto Tecnológico Federal Suizo - Lausanne		73.8
15	Universidad de Cambridge		73.8
17	Universidad de Purdue - West Lafayette		73.7
18	Universidad de Cornell		73.3
19	Universidad de Toronto		71.2
20	Universidad de Tohoku		70.7
21	Universidad Nacional de Singapur		70.4
22	Universidad del Noroeste		70.2
22	Universidad Estatal de Ohio - Columbus		70.2
24	Universidad de Florida		69.8
25	Universidad de Wisconsin - Madison		69.1
26	Universidad de Kyoto		69.0
26	Universidad de Princeton		69.0
26	Universidad de California - Los Ángeles		69.0
26	Universidad de Washington		69.0
30	Colegio Imperial de Ciencia, Tecnología y Medicina		68.7
31	Universidad de Texas A&M		68.6
32	Universidad de Manchester		68.0
33	Instituto Tecnológico de Tokyo		67.9
34	Universidad Estatal de Calorina del Norte - Raleigh		67.8

Posición	Institución	País	Puntaje
35	Universidad de Minnesota - Twin Cities		67.4
36	Universidad de Ciencia y Tecnología de Hong Kong		66.3
37	Universidad Nacional de Cheng Kung		65.3
38	Universidad de Harvard		64.9
38	Universidad de Pennsylvania		64.9
40	Instituto Tecnológico de Israel		64.4
41	Instituto Tecnológico Federal de Zurich		63.9
42	Universidad de California - Irvine		63.6
43	Instituto Tecnológico de Virginia		63.3
44	Universidad de Tsinghua		63.2
45	Universidad Nacional de Chiao Tung		63.0
46	Universidad de Oxford		62.8
47	Universidad de California - Davis		62.7
48	Universidad Estatal de New Jersey		62.4
49	Universidad McGill		61.9
50	Universidad de Massachusetts - Amherst		61.8
51	Universidad Estatal de Arizona - Temple		
52	Universidad Católica de Leuven		
53	Universidad Tecnológica de Chalmers		
54	Universidad de Hong Kong		
55	Universidad de Columbia		
56	Universidad Duke		
57	Instituto de Ciencias de la India		
58	Instituto Tecnológico de Kharagpur		
59	Universidad Nacional de Taiwán		
60	Universidad de Osaka		
61	Universidad de Pierre y Marie Curie - Paris 6		
62	Universidad Politécnica de Turín		
63	Instituto Politécnico de Rensselaer		
64	Universidad Nacional de Seúl		
65	Universidad de Shanghái - Jiao Tong		
66	Universidad Tecnológica de Dinamarca		
67	Universidad China de Hong Kong		
68	Universidad de Sheffield		
69	Universidad de British Columbia		
70	Universidad de Colorado - Boulder		
71	Universidad de Melbourne		
72	Universidad de New South Wales		

Posición	Institución	País	Puntaje
73	Universidad de Ciencia y Tecnología de China		
74	Universidad de Sídney		
75	Universidad de Virginia		
76	Universidad de Waterloo		
77	Universidad de Zhejiang		
78	Universidad Católica de Louvain		
79	Universidad Tecnológica de Delft		
80	Instituto de Ciencias Avanzadas y Tecnología de Corea		

Fuente: Tomado de (Academic Ranking of World Universities ARWU, 2009)

Anexo B.

Lista de los profesores internacionales titulares de cátedra convocados

Universidad	Nombre	Correo Electrónico
Instituto Tecnológico de Massachusetts	Daniel Jackson	dnj@mit.edu
Universidad de Illinois - Urbana Champaign	Ralph Johnson	johnson@cs.uiuc.edu
Universidad de California - Berkeley	Koushik Sen	ksen@cs.berkeley.edu
Universidad de Michigan - Ann Arbor	Chandrasekhar Boyapati	bchandra@eecs.umich.edu
Universidad de Texas - Austin	Dewayne Perry	perry@mail.utexas.edu
Universidad Carnegie Mellon	Johnatan Aldich	jonathan.aldrich@cs.cmu.edu
Instituto Tecnológico de Georgia	Richard DeMillo	richard.demillo@cc.gatech.edu
Universidad de California - San Diego	William G. Griswold	wgg@cs.ucsd.edu
Universidad del Sur de California	Barry Boehm	boehm@sunset.usc.edu
Universidad de California - Santa Bárbara	Tevfik Bultan	bultan@cs.ucsb.edu
Universidad de Maryland - College Park	Vibha Sazawal	vibha@cs.umd.edu
Universidad de Cornell	William Y. Arms	wya@cs.cornell.edu
Universidad de Purdue - West Lafayette	H. E. Dunsmore	dunsmore@cs.pardue.edu
Instituto Tecnológico Federal Suizo - Lausanne	George Candea	george.candea@epfl.ch
Universidad de Toronto	Steve Easterbrook	sme@cs.toronto.edu
Universidad Estatal de Ohio - Columbus	Atanas Rountev	rountev@cse.ohio-state.edu
Colegio Imperial de Londres	Alexander L. Wolf	a.wolf@imperial.ac.uk
Universidad de Washington - Seattle	David Notkin	notkin@cs.washington.edu
Universidad Nacional de Singapur	Stan Jarzabek	stan@comp.nus.edu.sg
Instituto Tecnológico de Tokyo	Katsuhiko Gondow	gondow@cs.titech.ac.jp
Universidad de Minnesota - Twin Cities	Mats Heimdahl	heimdahl@cs.umn.edu
Universidad de California - Los Angeles	Alfonso Cardenas	cardenas@cs.ucla.edu
Universidad de Florida	Peter J. Dobbins	pjd@cise.ufl.edu
Universidad de Texas A&M - College Station	William M. Lively	lively@cs.tamu.edu
Universidad de Harvard	Greg Morrisett	greg@eecs.harvard.edu
Universidad Estatal de Reutgers	Alex Borgida	borgida@cs.rutgers.edu
Universidad de Ciencia y Tecnología de Hong Kong	S.C. Cheung	sccheung@cse.ust.hk

Universidad	Nombre	Correo Electrónico
Universidad de California – Irvine	David Redmiles	redmiles@ics.uci.edu
Universidad de Osaka	Katsuro Inoue	inoue@ist.osaka-u.ac.jp
Universidad de Columbia	Alfred V. Aho	aho@cs.columbia.edu
Universidad de California – Davis	Premkumar T. Devanbu	devanbu@cs.ucdavis.edu
Universidad de Massachusetts - Amherst	Lori A. Clarke	clarke@cs.umass.edu
Instituto Tecnológico de Virginia	James D. Arthur	arthur@vt.edu
Universidad de Oxford	Jim Davies	jim.davies@comlab.ox.ac.uk
Universidad de Colorado - Boulder	Kenneth M. Anderson	kena@cs.colorado.edu
Universidad McGill	Joseph Vybihal	jvybihal@cs.mcgill.ca
Universidad de Tsinghua	ZHI. Zhixiong	chichihung@tsinghua.edu.cn
Universidad Tecnológica de Chalmers	Reiner Hähnle	reiner@chalmers.se
Universidad de Hong Kong	Y T Yu	csytyu@cityu.edu.hk
Universidad de Duke	Robert Duvall	rcd@cs.duke.edu
Universidad Nacional de Chiao Tung	Feng-Jian Wang	fjwang@csie.nctu.edu.tw
Instituto Politécnico de Turín	Maurizio Morisio	maurizio.morisio@polito.it
Instituto Politécnico Rensselaer	Cheng .K Hsu	hsuc@rpi.edu
Universidad Rice	Dung Nguyen	dxnguyen@rice.edu
Universidad Nacional de Seúl	Chisu Wu	wuchisu@selab.snu.ac.kr
Instituto Tecnológico Federal - Zurich	Bertrand Meyer	bertrand.meyer@inf.ethz.ch
Universidad Tecnológica de Dinamarca	Michael Reichhardt Hansen	mrh@imm.dtu.dk
Universidad de Tel Aviv	Amiram Yehudai	amiramy@tau.ac.il
Universidad de Tokyo	Shinichi Honiden	honiden@nii.ac.jp
Universidad de British Columbia	Gail C. Murphy	murphy@cs.ubc.ca
Universidad de Delaware	Lori L. Pollock	pollock@cis.udel.edu
Universidad de Leuven	Eric Steegmans	Eric.Steegmans@cs.kuleuven.be
Universidad de Melbourne	Zoltan Somogyi	zs@cs.mu.oz.au
Universidad de Ciencia y Tecnología de China	Zhao Baohua	bhzhao@ustc.edu.cn
Universidad de Sídney	Peter Eades	peter@it.usyd.edu.au
Universidad de Twente	Albert Van Den Berg	a.vandenberg@utwente.nl
Universidad de Virginia	Thomas Horton	horton@virginia.edu
Universidad Estatal de Arizona – Temple	James Collofello	james.collofello@asu.edu
Universidad de Boston	John Keklak	jkeklak@cs.bu.edu
Universidad de la Reserva del Oeste	Andy Podgurski	andy@eecs.case.edu
Universidad de Hong Kong	Elisa Baniassad	elisa@cse.cuhk.edu.hk
Universidad Tecnológica de Delft	P.G.Kluit	P.G.Kluit@tudelft.nl
Instituto de Ciencia de la India	K.V. Raghavan	raghavan@csa.iisc.ernet.in

Universidad	Nombre	Correo Electrónico
Instituto de Tecnología de Kharagpur	Rajib Mall	rajib@cse.iitkgp.ernet.in
Universidad Estatal de Iowa	Samik Basu	sbasu@cs.iastate.edu
Universidad Johns Hopkins	Peter H. Fröhlich	phf@acm.org
Instituto de Ciencias Avanzadas y Tecnología de Corea	Doo-Hwan Bae	bae@salmosa.kaist.ac.kr
Universidad de Kyushu	Keijiro Araki	araki@csce.kyushu-u.ac.jp
Universidad de Lund	Per Runeson	per.runeson@cs.lth.se
Universidad Estatal de Michigan	Laura Dillon	ldillon@cse.msu.edu
Universidad de Ciencia y Tecnología de Pohang	Kyo-Chul Kang	kck@postech.ac.kr
Universidad Tecnológica de Munich	Manfred Broy	broy@in.tum.de
Universidad de Alberta	Petr Musilek	Petr.Musilek@ualberta.ca
Universidad de Bristol	Kirsten Cater	cater@cs.bristol.ac.uk
Universidad de California – Riverside	Iulian Neamtiu	neamtiu@cs.ucr.edu
Universidad de Louvain	Kim Mens	Kim.Mens@uclouvain.be
Universidad de Nápoles - Federico II	Anna Rita Fasolino	anna.fasolino@unina.it
Universidad de New South Wales	David Ross Jeffery	rossj@cse.unsw.edu.au
Universidad de Roma - La Sapienza	Paolo Bottoni	bottoni@di.uniroma1.it
Universidad de Utah	Robert R. Kessler	kessler@cs.utah.edu

Fuente: Esta investigación.

Anexo C.

Lista de profesores internacionales titulares de cátedra que participaron en la investigación

Lista de profesores internacionales titulares de cátedra que participaron en la investigación y registro documental (Datos de contacto, profesionales, investigaciones y publicaciones)

1. Prof. RALPH E. JOHNSON (Universidad de Illinois - Urbana Champaign)

INFORMACION DE CONTACTO	
Departamento de Ciencias de la Computación Thomas M. Siebel Centro para las Ciencias de la Computación 201 N Goodwin Avenue Urbana, IL 61801 **Miembro de** ACM e IEEE Computer Society. **Miembro de** Hillside Group **PhD**, **MS**. - Cornell **BA.** - Knox College. ***Correo Electrónico*** johnson@cs.uiuc.edu http://st-www.cs.illinois.edu/users/johnson/ Profesor del Departamento de Ciencias de la Computación en la Universidad de Illinois, Líder del Grupo de patrones y arquitecturas de software UIUC y la coordinación de proyectos del departamento. Intereses personales en Diseño orientado a objetos, especialmente marcos de trabajo, patrones, objetos de negocios, y lenguajes Smalltalk. Java, C, y Fortran.	

2. Prof. PREMKUMAR T. DEVANBU (Universidad de California - Davis)

INFORMACION DE CONTACTO	
Universidad de California - Davis devanbu@cs.ucdavis.edu Sitio Web: http://www.cs.ucdavis.edu/~devanbu/ Desarrollador industrial para Pekín-Elmer; y los Laboratorios Bell Labs, investigador en Murray Hill, NJ; y los Laboratorios de Investigación AT&T en Florham Park. Profesor de UC Davis desde 1997. Pregrado realizado en IIT, Madras en la India; Doctorado en Rutgers University	

CURSOS IMPARTIDOS
✓ Introducción a la Programación y Solución de Problemas ✓ Ingeniería de Software – Pregrado ✓ Ingeniería de Software - Postgrado ✓ Seminario de grado en ingeniería de software con fuentes abiertas

3. Prof. TEVFIK BULTAN (Universidad de California - Santa Bárbara)

INFORMACION DE CONTACTO	
Universidad de California - Santa Bárbara *Correo Electrónico:* bultan@cs.ucsb.edu Sitio Web: http://www.cs.ucsb.edu/~bultan/ *Profesor* : Departamento de Ciencias de la Computación Universidad de California, Santa Bárbara	

INFORMACION PROFESIONAL

Actividades Profesionales Actuales

Miembro de los comités MEMOCODE 2010, ISSTA 2010, SOCA .

Actividades Académicas

Ingeniería de Software

4. Prof. DAVID NOTKIN (Universidad de Washington - Seattle)

INFORMACION DE CONTACTO	
Universidad de Washington – Seattle Correo Electrónico: notkin@cs.washington.edu Sitio Web: ww.cs.washington.edu/homes/notkin/	

INFORMACION PROFESIONAL

- ✓ Licenciado en Ingeniería de Software
- ✓ Coautor SWEBOK: Documento de IEEE-CS
- ✓ Editor de *ACM Transactions on Software Engineering and Methodology* Associate editor of Profesor titular de Ingeniería de Software en la Universidad de Washington

5. Prof. DANIEL JACKSON (Instituto Tecnológico de Massachusetts)

INFORMACION DE CONTACTO	
Instituto Tecnológico de Massachusetts *Correo Electrónico*: dnj@mit.edu *Sitio Web*: http://people.csail.mit.edu/dnj/ Profesor, Ciencias de la Computación; Becario MacVicar Grupo de Diseño de Software Laboratorios de Ciencias de la Computación e Inteligencia Artificial Depto. de Ingeniería Eléctrica y Ciencias de la Computación	
INFORMACION PROFESIONAL	
Docencia. Ingeniería de Sistemas de Computación, Elementos de Construcción de Software Principios de Sistemas de Computación Métodos formales livianos Laboratorio de Ingeniería de Software Tópicos Avanzados en Diseño de Software	

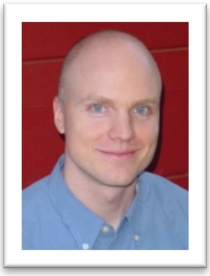
6. Prof. JOSEPH VYBIHAL (Universidad McGill)

INFORMACION DE CONTACTO	
***Correo Electrónico*:** jvybihal@cs.mcgill.ca **Sitio Web:** http://www.cs.mcgill.ca/~jvybihal ***Oficina*:** MC323 ***Tel*:** +1-514-398-7071 (ext. 00092) ***Fax*:** +1-514-398-3883 Dirección Escuela de Ciencias de la Computación Universidad McGill Edificio de Ingeniería McConnell 3480 University Street # 318 Montreal, QC, Canadá H3A 2A7	
INFORMACION PROFESIONAL	
Docencia: Programación, Sistemas de Software, Sistemas de Computación, Laboratorios de Sistemas de Computación, Sistemas Operativos, Ingeniería de Software	

7. Prof. GEORGE CANDEA (Instituto Tecnológico Federal Suizo- Lausanne)

INFORMACION DE CONTACTO

Profesor Asistente
PhD (Universidad de Stanford, 2005)
george.candea@epfl.ch
Oficina(s): INN330
Tel(s): [+41 21 69] 34648,31432
fax: 021 693-8130



INFORMACION PROFESIONAL

Profesor Asistente de Ciencias de la Computación en Lausanne, Director de los Laboratorios de Sistemas. , fundador del equipo de computación orientada a la recuperación. Trabajo con Oracle Corp. En la división de tecnologías de servidor y en los laboratorios de investigación de IBM y Microsoft Research.

Docencia

Ingeniería de Software (BS) , Principios de Sistemas (MS), Tópicos Avanzados en Sistemas de Software (PhD), Tópicos Avanzados en Sistemas Operativos (PhD), Tópicos Avanzados en Sistemas de Computación Aplicada (PhD)

8. Prof. JOHN KEKLAK (Universidad de Boston)

INFORMACION DE CONTACTO

64 Cummington St, PSY-234
Boston, MA 02215, USA
Tel: (978) 502-3051 , Fax: (617) 353-6457
Correo Electrónico: jkeklak@bu.edu

INFORMACION PROFESIONAL

Catedrático de Ciencias de la Computación en la Universidad de Boston University y es investigador y consultor de Ingeniería de Software. Imparte el curso de Ingeniería de Software,
Es Ingeniero Mecánico del MIT en 1981 y se especializo en Ingeniería de Software a nivel de máster y PhD

Reconocimientos

2007 - Aplicación de Patente Internacional 60/900,542 para técnicas en ingeniería de software.
2007 - Patente 7,184,949 por la lógica interna de Ray Kurzweil
2006 - Patente 6,941,262 por la interfaz de usuario Ray Kurzweil
2004 - Patente 6,754,823 por la técnica de sistemas distribuidos
2003 - Patente 6,647,395 por algoritmos de procesamiento de lenguaje natural
1989 - Patente 4,809,201 por algoritmos de procesamiento de regiones graficas

9. Prof. PAOLO BOTTONI (Universidad de Roma - La Sapienza)

INFORMACION DE CONTACTO	
Profesor Asociado ***Correo Electrónico:*** bottonidi.uniroma1.it ***teléfono***: +39.06.4991.8426 ***Oficina***: Departamento de Informática, st. 312	

INFORMACION PROFESIONAL

Obtuvo su título de Físico en 1988 de la Universidad de Milán. Obtuvo su PhD. en Informática de la Universidad de Milán en 1995. Desde 1994 ha trabajado para el departamento de informática de la Universidad de Roma, Actualmente imparte los curso de Informática e Ingeniería de Software

10. Prof. DEWAYNE PERRY (Universidad de Texas - Austin)

INFORMACION DE CONTACTO	
Universidad de Texas – Austin Profesor Titular Perry@ece.utexas.edu Laboratorio de Ingeniería de Software Empírico Ciencias de la Computación Universidad de Texas - Austin	

INFORMACION PROFESIONAL

Miembro de ACM , IEEE CS – Profesor de los Cursos de Ingeniería de Software

11. Prof. P. G. KLUIT (Universidad Tecnológica de Delft)

INFORMACION DE CONTACTO	
Universidad Tecnológica de Delft: Dr. P.G. Kluit EWI/ST/SERG/ Mekelweg 4 2628 CD Delft Oficina: 8.030 Tel: +31-15-2787181 Correo Electronico: P.G.Kluit@tudelft.nl	

INFORMACION PROFESIONAL

Docencia

Programación Orientada a Objetos, Algoritmos y Programación, Ingeniería de Software I, Diseño Orientado a Objetos.

12 Prof. JONATHAN ALDRICH (Universidad Carnegie Mellon)

INFORMACION DE CONTACTO

Profesor Asociado
Instituto de Investigación en Software,
Escuela de Ciencias de la Computación
Universidad Carnegie Mellon,
5000 Forbes Avenue
Pittsburgh, PA 15213-3891,
jonathan.aldrich@cs.cmu.edu,
http://www.cs.cmu.edu/~aldrich/ , +1-412-268-7278

INFORMACION PROFESIONAL

Cursos

- Análisis de Programas,
- Prácticas de Ingeniería de Software,
- Fundamentos de Ingeniería de Software,
- Análisis de Sistemas de Software (Cursos de MS y PhD.)
- Paradigmas Emergentes de Programación

13 Prof. ALEXANDER L. WOLF (Colegio Imperial de Londres)

INFORMACION DE CONTACTO

+44 (0) 207 594 8211 (teléfono)
+44 (0) 207 594 8932 (fax)
Departamento de Computación
180 Queen's Gate
Colegio Imperial de Londres
Londres
SW7 2AZ
Reino Unido

INFORMACION PROFESIONAL

Profesor en el Departamento de Computación. Recibió pregrado en Queens College de Nueva York, en Ciencias de la Computación. Recibió el M.S. y PhD. del Departamento de Ciencias de la Computación en la Universidad de Massachusetts - Amherst. Fue miembro del equipo técnico de los Laboratorios Bell de AT&T en Murray Hill, Nueva Jersey. Es miembro de ACM Council, y actualmente imparte las cátedras de Diseño de Software.

14. Prof. Kenneth M. Anderson (Universidad de Colorado - Boulden)

INFORMACION DE CONTACTO

Universidad de Colorado
Departamento de Ciencias de la Computación
430 UCB
Boulder CO 80309-0430
Tel: +1 (303) 492-6003
Fax: +1 (303) 492-2844
Correo Electrónico: kena@cs.colorado.edu
Oficina: ECCS 111A

INFORMACION PROFESIONAL

Ganador de la beca de NSF, Es miembro activo de ICSOFT 2010

Formación PhD., en Informática y Ciencias de la Computación (1997) Universidad de California, Irvine M.S., en Informática y Ciencias de la Computación (1992) Universidad de California, Irvine B.S., en Informática y Ciencias de la Computación (1990) Universidad de California, Irvine	Imparte las cátedras de Diseño de Software e Ingeniería de Software a nivel de Pregrado, Máster y PhD.

15. Prof. Robert Duvall (Duke University)

INFORMACION DE CONTACTO	
✓ Profesor Universidad Duke Departamento de Ciencias de la Computacion D228 LSRC Box 90129 Durham, NC 27708-0129 rcd@cs.duke.edu	
INFORMACION PROFESIONAL	
Docencia: Introducción a las Ciencias de la Computación e Ingeniería de Software	

Anexo D.

Lista de universidades y profesores nacionales que participaron en la investigación.

Universidad:	Escuela de Ingeniería de Antioquia
Ubicación:	Medellín, Colombia
Depto., Colegio o Escuela:	Ingeniería Informática
Profesor:	Diego Hernán Montoya Bedoya
Contenidos del Curso:	Ingeniería de Software

Universidad:	Universidad de Nariño
Ubicación:	Pasto, Colombia
Depto., Colegio o Escuela:	Departamento de Sistemas
Profesor:	Alexander Barón
Contenidos del Curso:	Introducción a la Ingeniería de Software

Universidad:	Universidad del Cauca
Ubicación:	Popayán, Colombia
Depto., Colegio o Escuela:	Departamento de Sistemas
Profesor:	Carlos Cobos
Contenidos del Curso:	Ingeniería de Software I

Universidad:	Fundación Universitaria San Martin
Ubicación:	Bogotá, Colombia
Depto., Colegio o Escuela:	Ingeniería de Sistemas
Profesor:	Armando Rafael Acuña Martínez
Contenidos del Curso:	Fundamentos de Construcción de Software I

Anexo E

Formato de encuesta de opinión vía correo electrónico

VERSION ORIGINAL

Teaching of Software Design Survey

Regards. Universidad de Nariño (Colombia) develops a research about the teaching practices of Professors in courses related to Software Design at the international context. Given your excellent professional profile, we are pleased to invite you to participate in this study because of your experience in the field of education, as you are part of a selected group of 82 teachers from the list of best universities in the world in the field of Engineering and Computer Science. The information about this survey (3 minutes aprox.) helps to our research project, this information is confidential and it will be used for academic purposes only. Sincerely, RÓBINSON JIMÉNEZ, ÁLVARO MARTÍNEZ and JESÚS INSUASTY Professors / Universidad de Nariño Pasto / Colombia

---

* Required

Type your institution's name:

Do you use UML in the teaching of the course related to Software Design? *

- Yes
- No

Select the following Classroom Activities that you develop into the course related to Software Design: Activities in your classroom, with your presence

- Lectures
- Questions session
- Brainstorming
- Complementary readings
- Role-playing game
- Exercises development
- Classroom project
- Fieldwork
- Forum
- Seminar
- Other:

Select the following educational resources that you use in the course related to Software Design:

- Board
- Projector
- Printed documentation
- Digital documentation
- Audiovisual resources

- ☐ Pro/Specialized programs for Software Design
- ☐ Educational software
- ☐ Software to meet your students (software for communications)
- ☐ Other:

Select the following Independent Activities (out-of-classroom) that you propose into the course related to Software Design:
Activities out-of-class, whithout your presence

- ☐ Homework
- ☐ Complementary readings
- ☐ Exercises development
- ☐ Essay
- ☐ Research project
- ☐ Paper
- ☐ Fieldwork diary
- ☐ Blog
- ☐ Other:

Select the physical spaces that you use for the course related to Software Design:

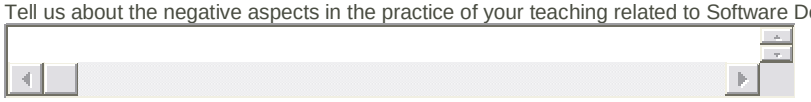
- ☐ Classroom
- ☐ Reading room
- ☐ Audiovisual room
- ☐ Audithorium/Conferences room
- ☐ Computer laboratory
- ☐ Virtual environments
- ☐ Other:

Select the different ways that you use to evaluate your students:

- ☐ Quiz
- ☐ Test
- ☐ Workshop
- ☐ Paper
- ☐ Report
- ☐ Project
- ☐ Portfolio
- ☐ Other:

Tell us about the positive aspects in the practice of your teaching related to Software Design:

Tell us about the negative aspects in the practice of your teaching related to Software Design:



Enviar

Available at: www.udenar.edu.co/dfsd-uml

VERSION TRADUCIDA

Encuesta de Didáctica del Diseño de Software

Saludos. La Universidad de Nariño (Colombia) desarrolla una investigación acerca de las prácticas de enseñanza de los profesores en los cursos relacionados con el Diseño del Software en contextos internacionales. Dado su excelente perfil profesional, nos complace invitarlo a participar en este estudio dada s experiencia en el campo disciplinar y educativo. Es Usted parte del selecto grupo de 82 profesores de las mejores universidades del mundo en el campo de la Ingeniería y las Ciencias de la Computación. La información de esta encuesta (de 3 minutos de duración aproximada) ayudara a nuestra investigación, de tal forma que la información consignada es estrictamente confidencial y se usara para los propósitos del proyecto únicamente. Atentamente, RÓBINSON JIMÉNEZ, ÁLVARO MARTÍNEZ y JESÚS INSUASTY; Profesores / Universidad de Nariño y Universidad Mariana; Pasto / Colombia

---

* Requerido

Escriba el nombre de la institución:

Usa UML en la enseñanza del curso relacionado con diseño de software? *

- Si
- No

Seleccione las actividades de clase que desarrolla en el curso relacionado con diseño de software: Es decir, las actividades dentro del salón de clase/laboratorio que se desarrollan con su presencia:

- Cátedras
- Sesión de Preguntas
- Lluvia de Ideas
- Lecturas Complementarias
- Juegos de Rol
- Desarrollo de Ejercicios
- Proyecto de Aula
- Trabajo de Campo
- Foro
- Seminario
- Otro:

Seleccione los siguientes recursos educativos que usa en el curso relacionado con Diseño de Software:

- Tablero
- Proyector
- Documentación Impresa

- ☐ Documentación Digital
- ☐ Recursos Audiovisuales
- ☐ Programas Profesionales/Especializados para el Diseño de Software
- ☐ Software Educativo
- ☐ Software para comunicaciones
- ☐ Otro:

Seleccione las siguientes actividades independientes (actividades fuera del aula) que Usted propone en el curso relacionado con el diseño de software: Es decir, actividades fuera de clase donde Usted no está presente

- ☐ Tareas
- ☐ Lecturas Complementarias
- ☐ Desarrollo de Ejercicios
- ☐ Ensayo
- ☐ Proyecto de Investigación
- ☐ Artículos
- ☐ Diario de campo
- ☐ Bitácora
- ☐ Otro:

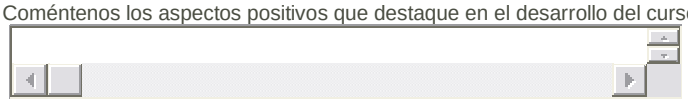
Seleccione los espacios físicos que Usted usa para el curso relacionado con diseño de software:

- ☐ Aula de clase
- ☐ Sala de lectura
- ☐ Sala audiovisual
- ☐ Auditorio/Salón de conferencias
- ☐ Laboratorio de Computación
- ☐ Ambientes Virtuales
- ☐ Otros:

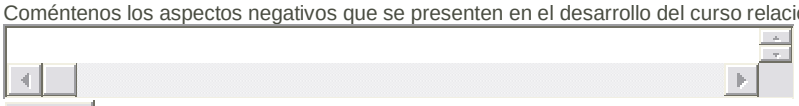
Seleccione las diferentes formas que Usted usa para evaluar a los estudiantes:

- ☐ Prueba Corta
- ☐ Examen
- ☐ Taller
- ☐ Articulo
- ☐ Informes
- ☐ Proyectos
- ☐ Portafolio
- ☐ Otro:

Coméntenos los aspectos positivos que destaque en el desarrollo del curso relacionado con Diseño de Software:



Coméntenos los aspectos negativos que se presenten en el desarrollo del curso relacionado con Diseño de Software:



Enviar

Disponible en: www.udenar.edu.co/dfsd-uml