Lámpsakos | Nº 10 | pp. 76-91 | julio-diciembre | 2013 | ISSN: 2145-4086 | Medellín - Colombia

ENSEÑANZA EN LA INGENIERÍA DE SOFTWARE: APROXIMACIÓN A UN ESTADO DEL ARTE

SOFTWARE ENGINEERING TEACHING: A STATE-OF-ART APPROACH

Lina María Montoya-Suárez, MSc.
Grupo de Investigación INGECO.
Universidad Autónoma Latinoamericana,
Medellín, Colombia
linamaria.montoya@unaula.edu.co

Elizabeth Pulgarín-Mejía, Esp.
Facultad de Educación
Universidad Nacional Abierta y a Distancia
Medellín, Colombia
epulgarin@hotmail.com

(Recibido el 20-05-2013 Aprobado el 21-06-2013)

Resumen. En este artículo se presenta una revisión de estado del arte sobre la formación en torno a la Ingeniería de Software, con una aproximación a aspectos relacionados con las estrategias de enseñanza aprendizaje basado en lúdica, haciendo énfasis en la enseñanza de la ingeniería de requisitos.

El artículo presenta una novedosa aplicación fundamentada en juegos, como estrategias de enseñanza con el objetivo de desarrollar habilidades, espacios de trabajo colaborativo, que permitan apoyar la investigación y la formación de maestros, entre otros. También se presenta un análisis sobre los diferentes estrategias lúdicas para la enseñanza en temáticas de la ingeniería del software al interior de un aula de clase.

Palabras clave: Educación informática, ingeniería de requisitos, ingeniería de software, lúdica, estrategias didácticas, técnicas de requisitos.

Abstract. This article presents an overview of state of the art teaching software engineering, with an approach to aspects of teaching strategies based on playful learning, emphasizing teaching requirements engineering.

The text presents different gaming applications as teaching strategies: aimed at developing skills, collaborative workspaces, and support research and training of teachers, among others, and performed at the end, an analysis of the different strategies for recreational teaching inside a classroom.

Keywords: Technology Education, software engineering, recreational, educational strategies, requirements techniques.

Citación de artículo, estilo IEEE:
L.M. Montoya-Suárez, E. Pulgarín-Mejía, "Enseñanza en la Ingeniería de Software: Aproximación a un Estado del Arte", Lámpsakos, N° 10, pp. 76-91, 2013.

1. INTRODUCCIÓN

Enseñar ingeniería de requisitos es fundamental para los proyectos de desarrollo de software, debido a que los requisitos marca el punto de inicio y fin de las actividades como la planeación [1]. Para esto se necesitan técnicas que se incorporen en el proceso de enseñanza en el área de Ingeniería de software, con el propósito de crear métodos de enseñanza que permitan abordar la complejidad inherente a los sistemas de desarrollo [2], para esto es fundamental comprender varias temáticas en especial el área de requisitos.

Las autoras involucradas en la realización de este trabajo de investigación pretenden presentar una revisión del estado del arte para la enseñanza de ingeniería de software [3]. El trabajo hace parte de una experiencia documentada en educación para cursos importantes dentro de la ingeniería informática.

La necesidad de dar respuesta al desarrollo de las habilidades en los estudiantes obliga a realizar una revisión sobre la literatura disponible acerca de las estrategias de enseñanza con componente lúdico, estrategias didácticas y su aplicación en aula de clase [3], a continuación se presenta un marco teórico.

Esta revisión del estado del arte consiste en dos secciones. Se presenta el marco teórico en torno a las definiciones dentro de la ingeniería de requisitos en la primera parte del artículo. En la segunda parte del artículo se presentan algunas estrategias consideradas para la enseñanza de la ingeniería de software, lo cual conduce a la presentación de algunas conclusiones al final del artículo.

2. MARCO TEÓRICO

La construcción de la revisión literaria, en torno a la enseñanza Ingeniería de Software, las cuales incluye las siguientes definiciones mínimas:

- Terminología utilizada en la enseñanza en la ingeniería, la elicitación de requisitos y metodologías lúdicas.
- Conceptualización asociada a enseñanza, ingeniería de software, ingeniería de requisitos, elicitación de requisitos.
- Factores que influyen en la práctica de enseñanza y aprendizaje en lúdicas.
- Metodologías y lúdicas relacionadas con ingeniería de software.

2.1 Definiciones en Ingeniería de software.

La ingeniería de software se define como el estudio de los principios y metodologías para desarrollo y mantenimiento de sistemas de software [4] [5].

Comprende la aplicación práctica del conocimiento científico en el diseño y construcción de programas de computadora y la documentación asociada requerida para desarrollar, operar (funcionar) y mantenerlos. Se conoce también como desarrollo de software o producción de software [6][7].

Trata del establecimiento de los principios y métodos de la ingeniería a fin de obtener software de modo rentable que sea fiable y trabaje en máquinas reales [7]. También se entiende como la aplicación de un enfoque sistemático, disciplinado, y cuantificable al desarrollo, operación, y mantenimiento del software; es decir la aplicación de Ingeniería de Software [8]. Lo anterior, implica la mención a una disciplina que integra el proceso, los métodos, y las herramientas para el desarrollo de software de computadora [9] [10].

En resumen, es una disciplina de la ingeniería que comprende todos los aspectos de la producción de software desde las etapas iniciales de la especificación del sistema hasta el mantenimiento de este después que se utiliza [11].

2.2 Definiciones en Ingeniería de Requisitos.

Esta disciplina ayuda a los ingenieros de software a entender mejor el problema en cuya solución trabajarán. Incluye el conjunto de tareas que conducen a comprender cuál será el impacto del software sobre el negocio, qué es lo que el cliente quiere y cómo interactuarán los usuarios finales con el software [10].

Según Sommerville en [11], se define la ingeniería de requisitos como el proceso de desarrollar una especificación de software. Las especificaciones pretenden comunicar las necesidades del software del cliente a los desarrolladores del sistema [11].

Existen otras definiciones de ingeniería de requisitos, que la consideran como un conjunto de actividades en las cuales, utilizando técnicas y herramientas, se analiza un problema y se concluye con la especificación de una solución [29].

2.2.1 Requisitos.

Los requisitos se refieren a una condición o necesidad de un usuario que permita resolver un problema o alcanzar un objetivo [12]. Lo anterior, simplemente significa una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste [11].

También se entienden como una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal [12][13].

2.2.2 Captura de los Requisitos.

La captura de los requisitos es la primera etapa para la construcción y comprensión del problema que se pretende solucionar a través de una aplicación de software [14]. Es fundamental una actividad humana y es allí donde se identifican los stakeholders (Los interesados del sistema); así como las relaciones se establecen entre el equipo del desarrollo y el cliente logrando "descubrir los requisitos," y la "adquisición de los requisitos" [15].

Uno de los aspectos fundamentales de la buena ingeniería de software es que exista buena comunicación entre los usuarios del aplicativo de software y los ingenieros [16]. Antes que comience el desarrollo, los expertos en requisitos deben formar el conducto para ser mediador entre el dominio de los usuarios del software y el mundo técnico del ingeniero de software hace la tarea de traducir la necesidad de los clientes utilizando diversas técnicas [17]

2.2.3 Técnicas de Captura de requisitos.

El proceso de definición en la ingeniería de requisitos, requiere una concentración en las técnicas. En este punto, los ingenieros informáticos necesitan sensibilizarse dado que los usuarios pueden tener dificultad para describir sus tareas [18]. En muchos casos se puede dejar información importante sin especificar, o los usuarios pueden estar poco dispuestos a cooperar, ni tampoco contarán con el tiempo suficiente que permita describir lo que esta definición de requisitos [19]. En este contexto los stakeholders y los ingenieros de requisitos trabajan en forma conjunta para identificar "qué", "dónde" y "cómo" [15].

Es particularmente importante entender que la captura de requisitos no es una actividad pasiva y que ingenieros de requisitos tienen que trabajar arduamente para extraer la información adecuada.

2.3 Definiciones en el proceso Enseñanza–Aprendizaje.

2.3.1 Enseñanzas basadas en juegos.

Los juegos, la enseñanza y el aprendizaje pretenden la adquisición de conocimientos, hábitos, capacidades partiendo de la persona como entidad global. Las experiencias e intereses le permiten al ser humano aprender y generar conocimiento. También le permiten motivarse, lo que provoca cambios a nivel educativo [20].

A pesar de las discrepancias existentes en las teorías de la motivación, la mayoría de los especialistas coinciden en la definición de motivación como el conjunto de procesos implicados en la activación, dirección y persistencia de la conducta [21].

2.3.2 Categorías de los resultados del aprendizaje.

Las categorías de aprendizaje, según la estructuración propuesta por Rodas [22], consisten en:

Habilidades intelectuales: Son las capacidades que hacen al estudiante competente, y le permite responder a las conceptualizaciones de su entorno social. Forma la estructura fundamental y al mismo tiempo una cobertura en la educación formal, incluyendo las habilidades más simples del lenguaje como hacer un escrito básico o frase hasta las habilidades avanzadas de la ciencia o la ingeniería [22].

Estrategias Cognoscitivas: Son las capacidades que administran el proceso de aprendizaje del individuo, su retentiva y su conducta al pensar [21] [22] [23].

Información verbal. La mayoría de seres humanos reciben cantidad relevante de información o contenido verbal. Existe mucha información almacenada en la memoria de las personas, en forma de palabras (nombres de pueblos y ciudades, nombres de los meses, de las personas, de las cosas, etc.). Los seres humanos están en capacidad de guardar información en forma organizada, cuando se relaciona

con hechos históricos, formas de gobierno, noticias de éxitos de las ciencias, etc [21] [22] [23].

Habilidades Motrices. Se refiere a la capacidad humana para caminar, manejar un vehículo, escribir, dibujar, pintar, manejar una serie de herramientas, entre otros [21] [22] [23].

Actitudes. Con base al dominio afectivo de la persona, se pueden identificar algunas capacidades aprendidas o "actitudes". Los seres humanos tienen actitudes hacia personas, cosas o situaciones [21]. El efecto de tales actitudes es magnificar las reacciones positivas o negativas, lo cual se denomina como "estados internos de la persona", los cuales pueden observarse indirectamente en su conducta [22].

3. FASES DE APRENDIZAJE

La literatura en general permite evidenciar los métodos de enseñanza enfocados en tres aspectos fundamentales para la ingeniería informática, con lo que se pretende identificar algunas características o elementos relevantes para ser incorporados en el diseño de métodos para la enseñanza.

3.1 Enseñanza general en la Ingeniería Informática

En esta fase se consideran las estrategias de aprendizaje como procesos de toma de decisiones (conscientes e intencionales) en los cuales el estudiante elige y recupera, de manera coordinada, los conocimientos que necesita para asistir una determinada demanda u objetivo, dependiendo de las características de la situación educativa en que se produce la acción [24].

Wankat & Oreovicz en [25] presentaron un conjunto de estrategias didácticas para ingeniería en general, en la enseñanza incluyen las clases magistrales y los proyectos prácticos. Otras estrategias, como los juegos, los estudios de casos y la educación personalizada, poco se han aplicado en la enseñanza de la ingeniería de software. Estos autores, además, afirman que la clase magistral como estrategia didáctica requiere ser complementada por otras estrategias para alcanzar objetivos cognitivos de más alto nivel [25].

La enseñanza tradicional de la ingeniería se ha complementado con nuevos métodos que ofrecen posibilidades en el conocimiento. Los juegos establecen estrategias que pueden complementar la enseñanza tradicional, pues su objetivo no es reemplazar las clases magistrales y los proyectos prácticos, sino suministrar y apoyar los espacios para afianzar los conceptos que se ofrecen en los tipos tradicionales de enseñanza [25].

Godoy en [26] presenta una revisión de las contribuciones de Roger C. Schank en [27] y propone desarrollar actividades en un ambiente virtual. Este autor sostiene que solo se aprende a través de intentar hacer cosas que requieran conocimiento y no escuchando a un docente narrar reglas del oficio. Schank en [27] resalta la importancia de tener presente las fallas de quien aprende, y que la falla conduce a reconstruir la estructura del conocimiento para explicarse por qué las predicciones no resultaron correctas, para esto presenta la estructura del razonamiento basada en casos y en reglas; el modelo de explicación basado en casos consta de cinco arquitecturas de enseñanza [28], las cuales se denominan:

1. Aprendizaje Activo.
2. Aprendizaje incidental.
3. Aprendizaje por reflexión.
4. Enseñanza basada en casos.
5. Aprendizaje por exploración.

3.2 Enseñanza Específica en la Ingeniería de Software.

Los ingenieros de software deben tener aptitudes de tipo administrativo, que poco se cultivan en la enseñanza tradicional [29]. Hoy en día, la enseñanza no se basa sólo en los conceptos impartidos por el maestro, sino que se inclina hacia el estudiante como elemento central de la clase. Es por ello que ahora se emplean los juegos como herramienta pedagógica [29] [30].

El desarrollo de software ha sido desde sus inicios una actividad caótica, complicada de entender y poco ingenieril [31]. En efecto, si se observa años atrás, se encuentran más problemas que aciertos. Es común que los proyectos de software se atrasen, no terminen a tiempo, se liquiden y que los presu-

puestos destinados a ellos se desborden sin límite; es común que los esfuerzos en la gestión de esos proyectos caiga en terrenos estériles y que en ocasiones parezca que esos proyectos van a la deriva [32].

Algunas reflexiones personales de parte de Ruiz en [33], sobre el papel que juega la ingeniería del software en la informática, argumenta la necesidad de una reforma en los contenidos en la manera de enseñar y aprender apostándole a la incorporación transversal en toda la carrera de informática una perspectiva de ingeniería [33].

Anaya en [34] propone una visión de la enseñanza en la ingeniería de software como apoyo al mejoramiento de las empresas con un fin: la integración de factores técnicos, gerenciales y organizacionales permitiendo mejorar la práctica del desarrollo en las organizaciones como principios y estrategias. Su trabajo se resume en dos aspectos básicos: uno es que los profesores deben tener una vista unificada acerca del cuerpo de conocimiento que soporte esta área y lo segundo, la universidad debe tener una percepción de la realidad de las prácticas en las organizaciones de software y los problemas que enfrentan, por la falta de aplicación de las buenas prácticas de ingeniería de software [34].

Marqués en [35] plantea un ciclo de desarrollo para software educativo de programas en diez fases y hace una descripción detallada de las actividades y recursos fundamentales para cada una de las etapas, el eje de la construcción de los programas educativos parte de una idea inicial (semilla) con el propósito de favorecer los procesos de enseñanza / aprendizaje y que va tomando forma poco a poco para el pre-diseño o diseño funcional; una idea que conforma unas actividades atrayentes para el alumno que potencialmente pueden facilitar la consecución de unos determinados objetivos educativos [35].

Gayo Lanvin & Salvador en [36] describe una experiencia, desarrollada en una asignatura utilizado herramientas colaborativas en el desarrollo de software libre utilizando la herramientas Sourceforgo, creado un proyecto común entre todos los estudiantes, con el objetivo de facilitar un aprendizaje basado en proyectos.

Guitart et al. [37] hace una elección del modelo de evaluación como caso práctico para asignaturas de ingeniería del software, con base al planteamiento inicial del proceso de evaluación, que debe de ser coherente con los objetivos de aprendizaje. Para esto tuvieron en cuenta las características fundamentales de la asignatura y como influía en el modelo de evaluación, generando los siguientes descriptores: objetivos generales de aprendizaje, secuencialidad, semestre en que se sitúa la asignatura, metodología de aprendizaje a seguir, tipología de la asignatura y número de aulas por asignatura. En resumen, para los objetivos generales de aprendizaje, se han distinguido tres elementos [37]:

- **Conceptual**: Relacionado con el incremento del conocimiento teórico del saber de un área.
- **Procedimental**: Se basa en la ampliación de conocimiento práctico del saber de un área.
- **Integrador**: Relacionado con el crecimiento de conocimiento sobre las destrezas, aptitudes y actitudes propias del ejercicio de una profesión.

A partir de los descriptores mencionados en [37], ha sido posible construir un resumen de la discusión del modelo de evaluación, según se presenta en la Tabla 1:

Vale la pena mencionar en esta revisión, el trabajo de Mariño et al. en [38], en el que se presenta un software interactivo orientado a la enseñanza de un método de programación por camino crítico. Para el desarrollo de la aplicación se utilizaron varias técnicas y tecnologías de comunicación. A partir de estas técnicas, se priorizaron aquellas relacionadas con el diseño de aplicaciones interactivas hipermediales y se adoptó la técnica basada en el desarrollo de prototipos informáticos de Kendall [39] y de Whitten [40] orientándola a la producción de material didáctico interactivo. Para ello siguieron los siguientes pasos:

1. Evaluación y estudio de las herramientas orientadas a la construcción de software multimedial educativo.
2. Selección de los contenidos temáticos específicos.
3. Definición de dos etapas para la elaboración del producto.
4. Diseño de las interfaces, la arquitectura de las pantallas y la secuencia del recorrido.

Tabla 1: Modelos de evaluación para Ingeniería de Software

Tipología	Objetivos generales de aprendizajes	Modelo de Evaluación
Secuenciales Terminales	Conceptual	Examen o prueba de Validación
	Procedimental	Prueba de Validación o Evaluación continuada
	Integradora	Evaluación continuada
Secuenciales No Terminales	Conceptual	Examen
	Procedimental	Prueba de Validación o Evaluación continuada
	Integradora	**Evaluación continuada**

5. Creación del prototipo educativo.

6. Ejecución de pruebas y validaciones.

3.3 Enseñanza en la Ingeniería de Requisitos para el Desarrollo de Software

En la actualidad se encuentran herramientas en entornos virtuales que apoyan el proceso de enseñanza con fines educativos [41] En el ámbito multimedial, las principales tendencias en el diseño de juegos, se basa en juegos de computador [42].

Díaz et al. [43] presenta elementos teórico-metodológicos con el fin de guiar la elaboración de software con fines educativos, haciendo referencia a una guía metodológica, con la que se persigue propiciar un enfoque crítico, reflexivo, interdisciplinario e integrador, de los conceptos clave en el desarrollo de software educativo. Se hace énfasis en la relación que presenta la pedagogía en la ingeniería del software, acerca de los principios subyacentes en los dos paradigmas educativos, instructivismo y constructivismo, de forma implícita o explícita.

4. ENSEÑANZA BASADA EN JUEGOS EN LA INGENIERÍA DE SOFTWARE.

El juego de la consistencia es una herramienta didáctica para usar en las aulas de clase, que le permite al estudiante afianzar, estructurar, analizar los conocimientos sobre modelado, métodos de desarrollo de software, trabajo en equipo y comunicación en la ingeniería de software [44].

Para la enseñanza futura de la ingeniería de software, estrategias de autoaprendizaje para los estudiantes, y refiere específicamente entre estas estrategias la participación en investigación desde el pregrado y la realización de juegos instructivos en clase [45]. Esta sugerencia coincide con la apreciación de Fairley et al [46], para la Ingeniería en general, cuando afirman que se requiere que el aprendizaje en clase sea activo y cooperativo, de forma que los estudiantes tengan una actitud más activa en relación con los conocimientos que se ofrecen, a diferencia de las tradicionales clases expositivas que se suelen impartir. Estrategias como los juegos en clase no han sido comúnmente empleadas en la enseñanza tradicional de la ingeniería (o, más específicamente, en la Ingeniería de Software) para esto existen algunas experiencias

Zapata & Duarte en [44], proponen en su investigación el juego de la consistencia: una estrategia didáctica para la ingeniería de software como una manera de lograr el aprendizaje de los alumnos desde una experiencia lúdica; el juego permite trabajar el concepto de consistencia entre diagramas de UML (Unified Modeling Language), el estándar más aceptado en la actualidad para el modelado de sistemas informáticos en ingeniería de software [44].

4.1 Juego UNC Method

Zapata et al. en [47] propone un Juego para la Enseñanza de Métodos de Desarrollo de Software, específicamente "UNC-Method". Este método de desarrollo de software, se caracteriza por poseer cuatro fases: contexto del software, análisis del problema, propuestas de solución y esquema conceptual, en el interior de cada fase, el equipo de desarrollo debe elaborar un conjunto de artefactos que conforman un entregable no importa el orden de elaboración, pero sí se debe respetar el orden de secuencia entre las fases, con el fin de lograr el refinamiento de los artefactos, principalmente el diagrama de casos de uso, las interfaces gráficas de usuario y el diagrama de clases.

En la Figura 1 se observa el tablero de juego que posee un conjunto de zonas.

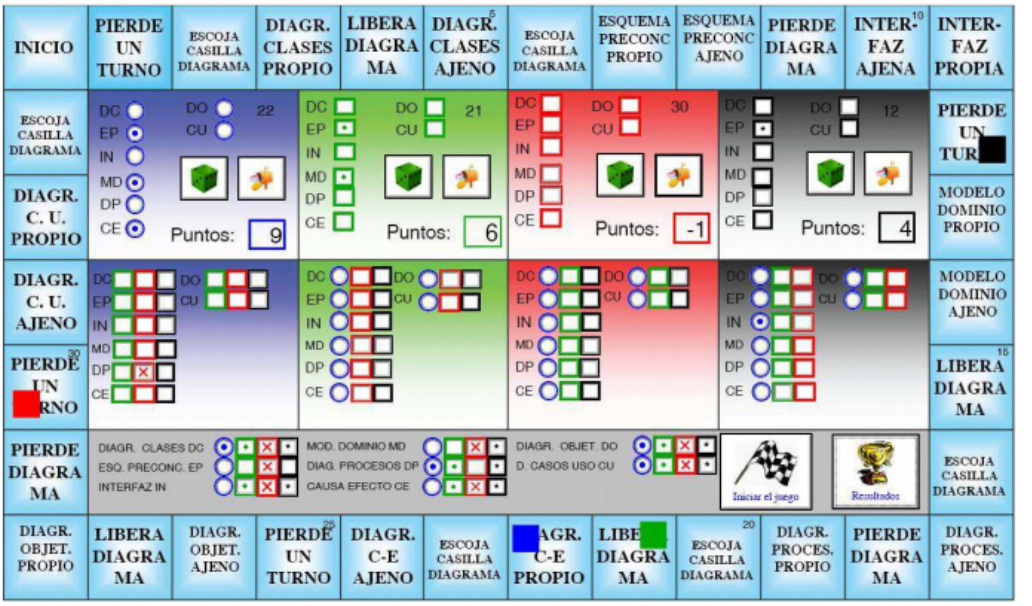

Figura 1: Tablero del juego del UNC-Method [47].

4.2 Tarjetas de Riesgos

En la Universidad de Carnegie Mellon se desarrolló un juego para la enseñanza de los conceptos básicos de la gestión de riesgos en el proceso de desarrollo de software, para el programa de Ingeniería, con el objetivo de afianzar la toma de decisiones [48]. Consiste en la simulación de un proyecto de desarrollo de software presenta aspectos importantes a tener en cuenta al diseñar un juego con propósito educativo [48].

El juego consiste en que un jugador asume el rol de director de proyecto y todos compiten unos contra otros, para lograr desarrollar un producto, venderlo en el mercado. El juego está estructurado en 5 fases: la planificación, los requisitos, la arquitectura y el diseño, la ejecución y el ensayo. Se ofrece a los jugadores un conjunto determinado de opciones para elaborar, estructurar y opinar sobre cómo y qué hacer en el proyecto.

Los participantes inician con un conjunto de recursos ya sea personal o dinero, se presentan cinco tarjetas de riesgo, cada una representada en la Figura 2

1. Proyecto de tarjetas.

2. Tarjetas de sorpresa.

3. Las tarjetas de perdón.

4. El riesgo

5. La mitigación de tarjeta.

Los jugadores en cada turno pueden hacer una de dos cosas: Realizar un proyecto de paso o Mitigar una situación de riesgo.

El objetivo del juego es mejorar el aprendizaje y la toma de decisiones con la simulación de un proyecto de desarrollo de software propuesto [48].

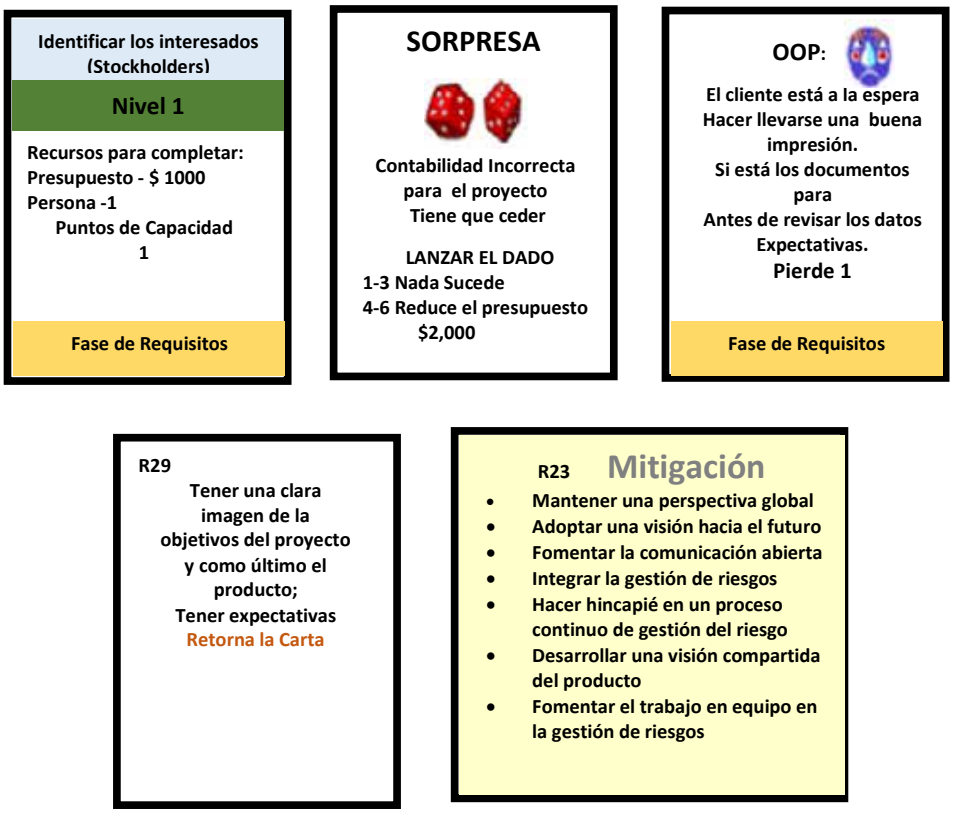

Figura 2: Ejemplo de las 5 Tarjetas de Riesgos [48].

4.3 Juego PoV

PoV-Game [30] es una aplicación de juego, mediante cartas, que busca concientizar a los participantes en la importancia de lograr acuerdos entre los diferentes puntos de vista de los actores participantes de una aplicación de software, además busca resaltar el resultado que se genera al unir las funciones de cada uno de los actores de manera consistente, planteando infinitas soluciones del sistema, que tiene un problema en el desarrollo de un software; el fin es procurar reconocer el alcance de una aplicación y seleccionar la información apropiada con el fin de lograr la satisfacción de los interesados del sistema.

La estructura del juego consiste en que 129 fichas de juego, están distribuidas de la siguiente manera:

- 18 actores (3 de cada uno).
- 54 procesos (2 de cada una).
- 51 objetos (6 de cada uno) identificadas con el color amarillo.
- Disposición de comodines (2 de cada tipo) identificadas con el color rosado.

Cada carta tiene un valor asignado el cual se usará para determinar el jugador que inicie la partida y en algunos casos, el ganador del juego. Los valores asignados son los siguientes: verbos: 1 punto, objetos: 2 puntos, actores: 3 puntos y comodines: 4 puntos.

Para lograr una diferenciación visual de las cartas y su contenido se utilizan los colores de la Figura 3. El tablero permite que todos los jugadores puedan leer fácilmente las palabras indicadas, además las señalizaciones permiten observar el orden temporal correcto en que deben realizarse las acciones para lograr el pago de la nómina, la estructura de juego consiste en que los participantes armen correctamente nueve frases que componen algunos de los procesos que se deben llevar a cabo a la hora de realizar el pago de la nómina.

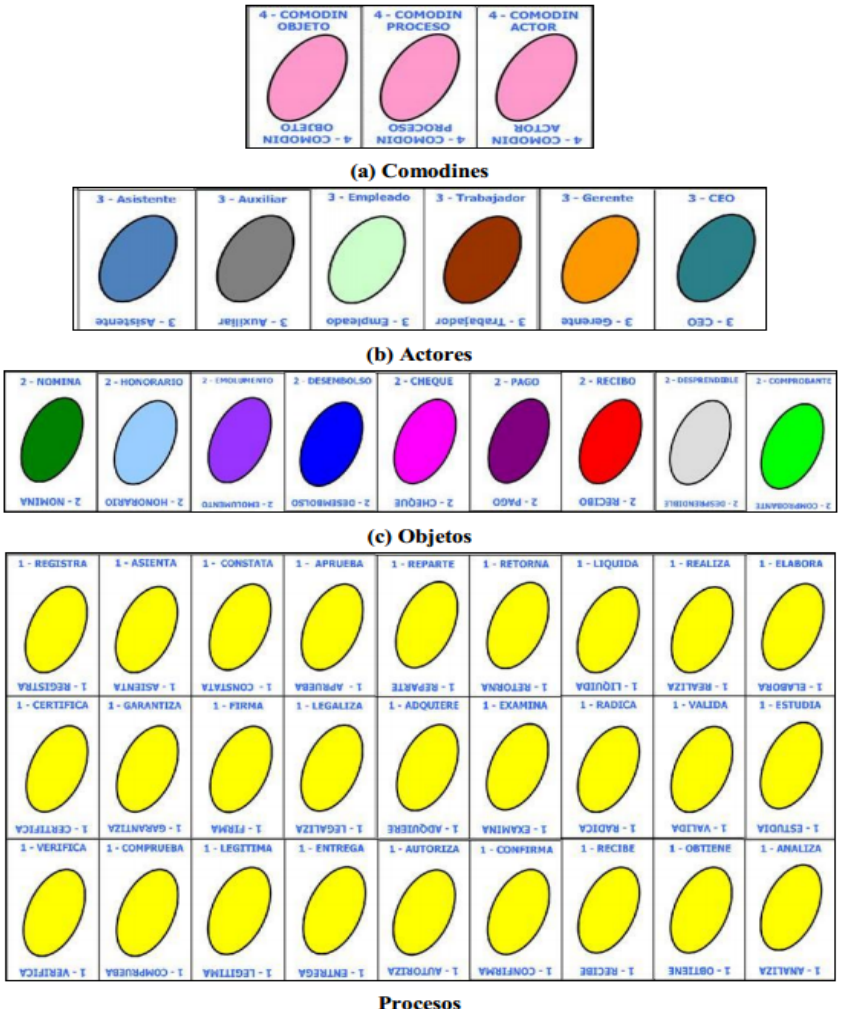

Figura 3: Cartas de juego en PoV GAME [30].

4.4 Origami para Desarrollo de Software

"El juego del desarrollo de software", está estructurado y elaborado con el objetivo es construir cajas de origami que estén marcadas con uno de los siguientes grupos de letras: SO, FT, WA o RE, cada una de las letras están representa en un módulo, como se observa en la Figura 4. Cada módulo debe cumplir con un conjunto de reglas definidas al principio del juego.

A cada grupo se le entrega una hoja con instrucciones representadas gráficamente, con el propósito de armar las cajas (ver Figura 5), esta hoja representa la materia prima junto con los cuatro grupos de letras descritos en la figura anterior y una hoja de control que tiene la posibilidad de recopilar la información del juego para determinar el ganador.

El director del juego, verifica el cumplimiento de las normas y solo él puede responder preguntas a sólo uno de los integrantes de cada grupo, el director termina la construcción y realiza el conteo de los módulos elaborados correctamente y las "compra", al final recopila la información de cada empresa y obtiene el balance en términos de pérdidas o utilidades. El grupo con mayores utilidades o menores pérdidas será el ganador.

4.5 Especificación de requisitos con HELER

Cuervo et al. en [51] presentan la realización de herramienta HELER (Herramienta Libre para la Especificación de Requisitos) una aplicación de carácter académico, libre y monousuario, para ambiente Windows y licencia GPL además desarrollada en java. Su arquitectura está dada por el patrón de diseño

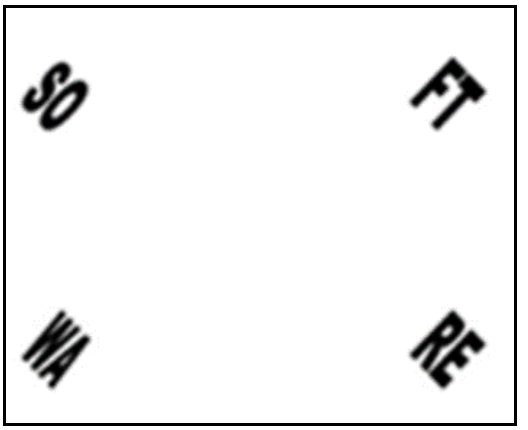

Figura 4: Hoja de materia Prima [49].

Modelo Vista Controlador, ofrece soporte a las actividades de Ingeniería de Requisitos y entendimiento del problema, enmarcada dentro del Proceso Unificado con base a 5 módulos: proyecto, stakeholder, actores, casos de uso, y requisitos.

4.6 Experiencia de SimulES

El juego del diálogo de deducción de "requisitos" simula de manera práctica y dinámica las condiciones de desarrollo de software en un entorno competitivo; aproximadamente en dos horas, los participantes están a cargo del desarrollo de una aplicación de software, con requisitos específicos establecidos previamente por el cliente, este juego consta de dos fases y cada rol tiene definido un conjunto de normas y especificaciones sobre lo que se debe realizar en cada etapa del juego; además se tiene un tiempo asignado para la realización de cada ciclo del juego y finalmente se determina el ganador, que será aquel equipo que genere mayores utilidades en el desarrollo de la aplicación, incluyendo la documentación presentada en [29].

Para la deducción de requisitos Zapata & Carmona en [52], proponen un modelo de diálogo para la ingeniería de software basado en requisitos, esta es una técnica para recolección, procesamiento y especificación de los requisitos de los interesados en el desarrollo de una aplicación de software con base a preguntas durante una entrevista, aplicado a un caso de estudio [52].

Monsalve et al. en [53] muestran cómo la elección y la aplicación oportuna de estrategias pueden llevar a la evolución y mejora de los sistemas, cuyo objetivo

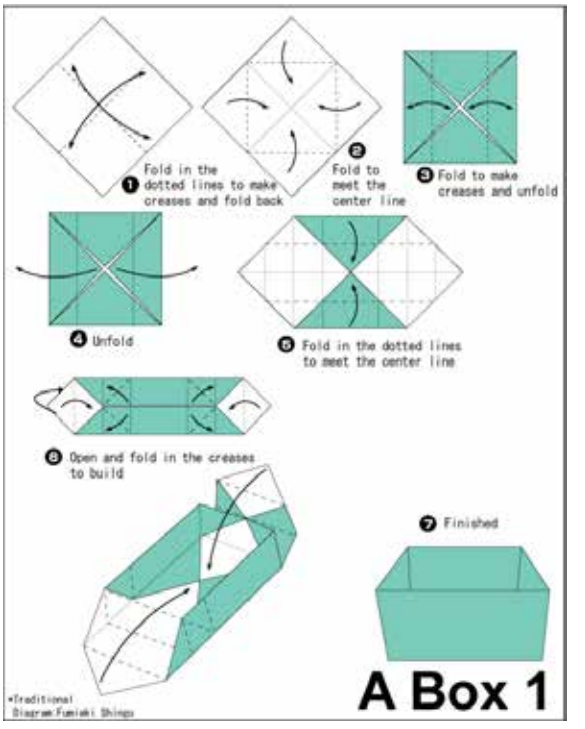

Figura 5: Instrucciones para el armado de origami de cubo [50].

es expresar a través de las técnicas de elicitación de requisitos como puede ser concebida su evolución por medio de un juego de cartas educacional SimulES. Este juego cuenta con un modelo situacional como fuente de información y lo deriva para un modelo intencional, adiciona conceptos de evolución de software e interacción entre jugadores al ser un juego multiusuario [53] .

En SimulES el estudiante asume el rol de gerente de proyecto y debe lidiar con el presupuesto del proyecto, contratar ingenieros, despedir ingenieros, construir los diferentes artefactos requeridos para el proyecto, entre muchas otras actividades.

En la Figura 6 se puede observar el proceso que se surge durante el juego, al inicio de la jugada se elige el proyecto que deberá ser construido durante el juego, además se debe de elegir las acciones de los artefactos propios del producto software, en cada jugada se tratan los conceptos y tratamiento de problemas usados y relacionados a los conceptos y problemas típicos en la ingeniería de software. Las estrategias abordadas para encontrar elementos de evolución en el juego SimulES fueron observación y cuestionarios.

Tabla 2: Aprendizaje en cuatro fases [57] .

Paso	Descripción
Atención	Aprender algo requiere atención, déficit en ella perjudican el aprendizaje.
Retención	Revisión de la información, codificándola y relacionándola con la información ya almacenada.
Reproducción	Comparación con la representación conceptual del individuo y reproducción del comportamiento.
Motivación	Las consecuencias de la observación informan a los observadores de su valor y estos a su vez tienen razones valederas para observar el modelo.

4.7 Otras Experiencias Exitosas

Los desarrollos actuales han mostrado la importancia de los procesos de captura, definición, validación e implementación en la ingeniería de requisitos [54]. En este sentido, se han efectuado estudios y se ha encontrado que una de las debilidades del desarrollo de software se encuentra comúnmente en el análisis de un problema; para esto se necesita buscar aproximaciones concretas aplicando los modelos que permitan mejorar la consistencia, la completitud, viabilidad en la comprensión del sistema y la gestión de los cambios [55].

Existen varios problemas para la enseñanza en las aulas de las técnicas de entrevistas para la definición de requisitos. De acuerdo a Oliveros [56], una de ellas es la dificultad de ejecutar una práctica real: las técnicas son meramente descriptas y en la mejor de las situaciones practicadas en un caso simplificado; otra dificultad por parte de personas de formación tecnológica es la subestimación de las técnicas "blandas" que requiere la elicitación de requerimientos, esta subestimación en muchos casos no se limita a los alumnos: se especula que una sólida formación técnica es suficiente para el alumno asegurar el éxito de un proyecto de software .

Par abordar el problema, tuvo en cuenta cada uno de los pasos identificados por Bandura [57] en el proceso de aprendizaje por observación (ver Tabla 3), quien desarrolló la experiencia basado en las cuatro fases de aprendizaje usando la técnica de requisitos: entrevista a dos grupos de estudiantes y obtuvo muy buenos resultados.

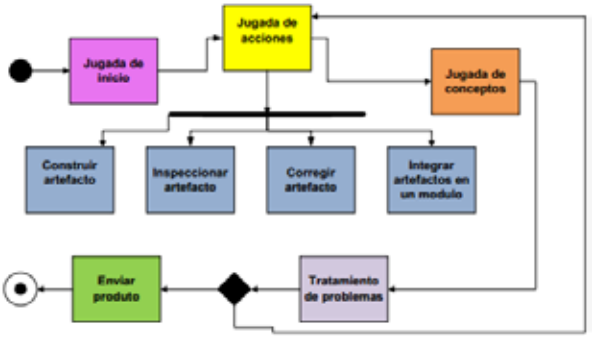

Figura 6: Dinámica de SimulES [53].

4.8 Juegos Aplicables a la Enseñanza de la Ingeniería de Requisitos.

A continuación se estructura la aplicación de juegos en la ingenieria de requisitos:

Zapata & Awad [29] proponen un juego para la ingeniería de requisitos a partir de roles y orientado al desarrollo de una aplicación de software basados en proyectos. Su fin esencial es simular aspectos en el cumplimiento de requisitos, la especialización de funciones, la completitud de la documentación, el trabajo en equipo, etc.

González et al. en [58] formulan el diseño y aplicación de juegos para la enseñanza de ingeniería de software a nivel de pregrado en la universidad de Medellín, presentando un conjunto de lúdicas implementadas y diseñadas en la asignatura "Ingeniería de Información" realizado en tres fases:

- Video casero sobre catástrofes de software.
- Los empresarios en el aula.
- Concurso para la asimilación de conceptos básicos de la Ingeniería de Requisitos.

Los resultados obtenidos le permitieron concluir que las estrategias basadas en lúdicas son útiles en el proceso de enseñanza-aprendizaje, debido a que el estudiante alcanza ser un sujeto activo, logrando generar mayor recordación de conceptos en el tiempo y el desarrollo de capacidades adicionales, lo que realmente ayuda al alumno aprender a partir de juegos [58].

Baker presenta [59] una simulación para la especificación de los requisitos en la ingeniería de software

con el fin de entregar una aplicación de software con Juego de cartas, usado para ejemplificar el proceso de desarrollo de software. Hace énfasis en la fase de implementación, donde los programadores se encuentran con inconvenientes de tipo humano, técnico o de cambios en el alcance inicial de un proyecto

Los jugadores tienen el reto de seguir las prácticas adecuadas de ingeniería de software a fin de evitar consecuencias adversas que podrían provocar su caída por detrás de su rival durante la competencia.

Al completar su proyecto, los jugadores deberán poner las cartas sobre la base del ciclo de vida del modelo en cascada, que se muestra en la Figura 7.

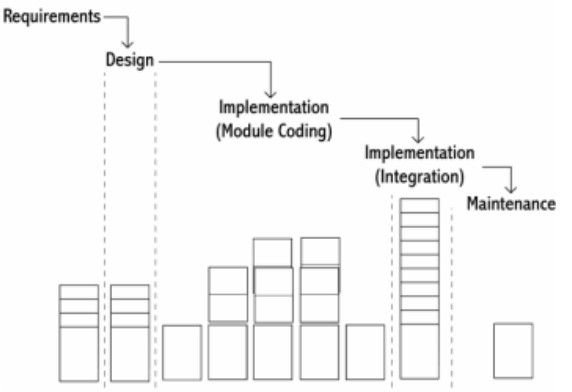

Figura 7: Fases y zonas en el juego problemas y programadores [59].

5. ANÁLISIS DE LAS EXPERIENCIAS

En la tabla 4 se muestra una síntesis de los estudios presentados hasta el momento y se realiza una clasificación por autor, año de publicación y Propuesta con un énfasis especial en el aspecto enseñanza en la ingeniería, enseñanza en la ingeniería de software, enseñanza en la ingeniería de requisitos en el desarrollo de software.

Tabla 4: Síntesis estudios presentados.

Autor y año	Propuesta
Wankat & Oreovicz, [25].	Presentaron un conjunto de estrategias didácticas para ingeniería en general, en la enseñanza incluyen las clases magistrales y los proyectos prácticos, basados en juegos.
Marqués [35].	Plantea un ciclo de desarrollo para software educativo de programas en diez fases y hace una descripción detallada de las actividades y recursos fundamentales para cada una de las etapas.
Mariño et al, [38].	Desarrollaron un software interactivo orientado a la enseñanza de un método de programación por camino crítico.
Baker [59].	Presenta una simulación para la especificación de los requisitos en la ingeniería de software con el fin de entregar una aplicación de software con Juego de cartas, usado para ejemplificar el proceso de desarrollo de software.
Anaya [34].	Propone una visión de la enseñanza en la ingeniería de software como apoyo al mejoramiento de las empresas con un fin: la integración de factores técnicos, gerenciales y organizacionales permitiendo mejorar la práctica del desarrollo en las organizaciones como principios y estrategias.
Gayo et al [36].	Describen una experiencia, desarrollada en una asignatura utilizado herramientas colaborativas en el desarrollo de software libre utilizando la herramientas Sourceforgo, creado un proyecto común entre todos los estudiantes, con el objetivo de facilitar un aprendizaje basado en proyectos.
Guitart et al, [37].	Estructuran y hacen una elección del modelo de evaluación como caso práctico para asignaturas de ingeniería del software, partiendo de un planteamiento inicial del proceso de evaluación, que debe de ser coherente con los objetivos de aprendizaje.
Díaz et al, [43].	• Estructuraron elementos teórico-metodológicos con el fin de guiar la elaboración de software con fines educativos, haciendo referencia a una guía metodológica, con la que se persigue propiciar un enfoque crítico, reflexivo, interdisciplinario e integrador, de los conceptos clave en el desarrollo de software educativo.
Taran [48].	La Universidad de Carnegie Mellon desarrolló un juego para la enseñanza de los conceptos básicos de la gestión de riesgos en el proceso de desarrollo de software, para el programa de Ingeniería, con el objetivo de afianzar la toma de decisiones.

Autor y año	Propuesta
Ruiz [33].	Afirma la importancia y el papel que juega la ingeniería del software en la informática y argumenta que es necesaria una reforma en los contenidos en la manera de enseñar y aprender.
Zapata& Awad [29].	Proponen un juego para la ingeniería de requisitos a partir de roles y orientado al desarrollo de una aplicación de software basados en proyectos. Simular aspectos en el cumplimiento de requisitos, en un entorno competitivo.
Zapata & Duarte [44].	Crearon un juego de la consistencia como herramienta didáctica para usar en las aulas de clase, que le permite al estudiante afianzar, estructurar, analizar los conocimientos sobre modelado, métodos de desarrollo de software, trabajo en equipo y comunicación en la ingeniería de software.
Zapata & Carmona [60].	Estructuran un modelo de diálogo para la ingeniería de software basado en requisitos, esta es una técnica para recolección, procesamiento y especificación de los requisitos de los interesados en el desarrollo de una aplicación de software con base a preguntas durante una entrevista, aplicado a un caso de estudio.
Cuervo et al. [51].	Presentan la realización de herramienta HELER (Herramienta Libre para la Especificación de Requisitos) 5 módulos: proyecto, stakeholder, actores, casos de uso, y requisitos.
Zapata, Calderón & Rivera, [61].	Crearon un juego PoV-Game, es una aplicación de juego, mediante cartas, que busca concientizar a los participantes en la importancia de lograr acuerdos entre los diferentes puntos de vista de los actores participantes de una aplicación de software.
Godoy [26].	Hace una revisión de las contribuciones de Roger C. Schank en [97] y propone desarrollar actividades en un ambiente virtual. Este autor afirma que el estudiante solo aprende a través de intentar hacer cosas que requieran conocimiento y no escuchando a un docente narrar reglas del oficio.
González et al. [58].	Proponen el diseño y aplicación para la enseñanza de ingeniería de software. “Ingeniería de Información” realizado en tres fases:
	1. Video casero sobre catástrofes de software.
	2. Los empresarios en el aula.
	3. Concurso para la asimilación de conceptos básicos de la Ingeniería de Requisitos.

Las investigaciones encontradas hasta la fecha se han realizado sobre las diferentes estrategias de enseñanza en la ingeniería de software, existe caso de éxito aplicado en aula de clase como valor añadido al proceso de enseñanza- aprendizaje, haciendo una búsqueda más profunda se encontró que hay caso de éxito en investigaciones donde evidencias resultados y análisis realizados sobre la importancia de utilizar estrategias didactas para enseñar Ingeniería de Software donde permite responder a necesidades de formación.

Es importante destacar la tendencia y significado que ha venido dando los docentes de universidades en Colombia para enseñar ingeniería de software con estrategias de enseñanza aprendizaje que sea didáctico con el objetivo de que los estudiantes puedan identificar y entender el problema, entender las causas que originan el problema, identificar a los involucrados en el sistema, definir los límites del sistema, el mantenimiento, la calidad y la documentación de diseño entre otros [15]

5. CONCLUSIONES

En este artículo se han presentado las investigaciones más significativas en torno a la enseñanza y aprendizaje en ingeniería de software. Algunas de estas investigaciones presentan y demuestran la importancia de incorporar en el aula de clase como apoyo a los cursos, la aplicación de estrategias didácticas para la enseñanza en Ingeniería de Software.

Se ha validado la hipótesis según la cual es fundamental para el desarrollo de nuevas metodologías identificar que estrategias de aprendizaje se pueden aplicar y que temáticas presentan los estudiantes dificultad para aprender, con base a esto se puedan abordar y estructurar como estrategias enseñanza-aprendizaje dentro del aula de clase, para garantizar una formación efectiva a estudiantes con habilidades y tipos de aprendizaje diferentes.

REFERENCIAS

[1] M. Arias, “La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software.” InterSedes: Revista de las Sedes Regionales, 6 (10)., 2005.

[2] E. Monsalve, V. Werneck, & J. Leite, “Evolución de un Juego Educacional de Ingeniería de Software a través de Técnicas de Elicitación de Requisitos,” in Proceedings of XIII Workshop on Requirements Engineering (WER’2010), Cuenca, Ecuador, 2010, pp. 12–23.

[3] L. M. Montoya, “Definición de un método para el proceso de enseñanza de elicitación de requisitos, basado en lúdicas.,”, Tesis Magistral en Ingeniería de Software, Universidad de Medellín, Medellín (Colombia), 2013.

[4] M. V Zelkowitz, A. C. Shaw, & J. D. Gannon, Principles of software engineering and design. Prentice-Hall Englewood Cliffs, 1979.

[5] T. Gilb & S. Finzi, Principles of software engineering management, vol. 4. Addison-Wesley Reading, MA, 1988.

[6] B. W. Boehm, “Software engineering economics,” IEEE Transactions on Pioneers and Their Contributions to Software Engineering, pp. 99–150, 2001,.

[7] R. S. Pressman & W. S. Jawadekar, “Software engineering,” New York 1992, vol. McGraw-Hil, New York, NY (USA), ISBN:0070507902, 1987.

[8] F. J. Zarazaga-Soria, M. I. Alonso-Galipienso, “La Ingeniería del Software en el currículo del Ingeniero en Informática,” Novática Rev. la Asoc. Técnicos Informática, N° 161, pp. 43–50, 2003.

[9] R. S. Pressman, Ingeniería del Software: Un enfoque práctico, McGraw-Hil. Mikel Angoar, 1997.

[10] R. S. Pressman, “Ingeniería del Software.” McGraw-Hill Interamericana-México. Disponible en: http://awl.com/cseng/otseries , 670p, 2006.

[11] I. Sommerville, Ingeniería del software, Pearson Ed. Madrid: Pearson Educación S.A., 2005, pp. 1–677.

[12] IEEE Computer Society Committee, IEEE Computer Society model program in computer science and engineering. IEEE Computer Society, 1983.

[13] A. Geraci, F. Katki, L. McMonegal, B. Meyer, J. Lane, P. Wilson, J. Radatz, M. Yee, H. Porteous, & F. Springsteel, IEEE standard computer dictionary: Compilation of IEEE standard computer glossaries. IEEE Press, 1991.

[14] I. Villanueva, J. Sánchez, & Ó. Pastor, “Elicitación de requisitos en sistemas de gestión orientados a procesos.,” in WER, 2005, pp. 38–49.

[15] I. Sommerville & G. Kotonya, Requirements engineering: processes and techniques, Pearson Ed. John Wiley & Sons, Inc., 1998, Disponible en: http://books.google.com.co/books/about/Ingenier%C3

[16] L. Jiang & A. Eberlein, “Selecting Requirements Engineering Techniques Based on Project Attributes--A Case Study,” 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS’07), 2007, pp. 269–278.

[17] K. Pohl, Requirements engineering: fundamentals, principles, and techniques. Springer Publishing Company, Incorporated, 2010.

[18] A. M. Davis, Software requirements: objects, functions, and states, Prentice-Hall, Upper Saddle River, NJ (USA), 1993. ISBN:0-13-805763-X

[19] P. Loucopoulos & V. Karakostas, System requirements engineering. McGraw-Hill, Inc., 1995.

[20] J. Beltrán Llera, “Procesos, estrategias y técnicas de aprendizaje,” Editor. Síntesis, SA Madrid, 1993.

[21] J. Beltrán, Procesos, estrategias y técnicas de aprendizaje. Síntesis, Madrid (España), 1996.

[22] J. M. Rodas, “Investigación sobre métodos de enseñanza-aprendizaje,” Boletín Electrónico. Univ. Rafael Landívar. Disponible en: http://www.academia.edu/1646623/INVESTIGACION_SOBRE_METODOS_DE_ENSENANZA-APRENDIZAJE

[23] C. López, "'Aprendizaje Organizacional,'" 2003. Disponible en: http://www.gestiopolis.com .

[24] C. Monereo, M. Castelló, M. Clariana, M. Palma, & M. L. Pérez, "Estrategias de enseñanza y aprendizaje," Editor. Grao, Barcelona, Vol 1, pp. 1–185, 1997.

[25] P. C. Wankat & F. S. Oreovicz, Teaching engineering. McGraw-Hill New York, 1993.

[26] L. A. Godoy, "Una revisión del programa de investigación sobre aprendizaje activo en un ambiente simulado desde la perspectiva de la educación en ingeniería," LatinAmerican and Caribbean Journal on Engineering Education, vol. 3 (2), 2013.

[27] R. Schank, "Aprendizaje Virtual: Un enfoque revolucionario para formar equipos de trabajo altamente capacitados." México DF, McGraw-Hill, 1997.

[28] R. C. Schank & C. Cleary, Engines for education. Lawrence Erlbaum Associates, Inc, 1995, p. Hillsdale, NJ, England: Lawrence Erlbaum Associate.

[29] C.M. Zapata-Jaramillo, G. Awad-Aubad, "Requirements Game: Teaching Software Project Management," CLEI Electronic Journal, vol. 10 (1), 2007.

[30] C. M. Zapata-Jaramillo, S. M. Villegas, & F. Arango, "Reglas de consistencia entre modelos de requisitos de UN-Metodo," Revista Universidad Eafit, Vol. 42 (141), pp. 40–59, 2012.

[31] C. M. Zapata-Jaramillo, A. Gelbukh, F. Arango, A. Hernández, & J. L. Zechinelli, "UN-Lencep: Obtención automática de diagramas UML a partir de un lenguaje controlado," Avances en la Ciencia de la Computación, pp. 254–259, 2006.

[32] C. M. Zapata-Jaramillo & F. Arango, "Alineación entre metas organizacionales y elicitación de requisitos del software," Dyna, vol. 143, pp. 101–110, 2004.

[33] F. La Ruiz, "Enseñanza de la Ingeniería del Software en el marco del Espacio Europeo de Educación Superior," Ponencia en el II Congreso Español de Informática (CEDI), 2007, pp. 1–20.

[34] R. Anaya, "Una visión de la enseñanza de la ingeniería de software como apoyo al mejoramiento de las empresas de software," Revista Universidad Eafit, vol. 42 (141), pp. 60–76, 2012.

[35] P. Marqués, "Metodología para la elaboración de software educativo," Barcelona (España). Editor. Estel, 1995.

[36] J. E. Labra-Gayo, D. Fernández-Lanvin, J. Calvo-Salvador, & A. Cernuda-del-Río, "Una experiencia de aprendizaje basado en proyectos utilizando herramientas colaborativas de desarrollo de software libre." XII Jornadas de Enseñanza Universitaria de la Informática, Disponible en: http://di002.edv.uniovi.es/~labra/FTP/Papers/LabraJenui06.pdf ,2006.

[37] I. Guitart, M. E. Rodríguez, J. Cabot, & M. Serra, "Elección del modelo de evaluación: caso práctico para asignaturas de ingeniería del software," Actas las XII Jornadas Enseñanza Univ. Informática, Jenui, pp. 191–198, 2006. Disponible en: http://bioinfo.uib.es/~joemiro/aenui/procJenui/Jen2006/prDef0035_96a3be3cf2.pdf

[38] S. I. Mariño, M. V. López, & M. F. Golobisky, "Un software interactivo orientado a la enseñanza del Método de Programación por Camino Crítico," VII Congreso Argentino de Ciencias de la Computación, 2001.

[39] K. E. Kendall & J. E. Kendall, Análisis y diseño de sistemas, Prentice Hall & Pearson Educacion, México DF, 2005, 714p.

[40] J. L. Whitten, V. M. Barlow & L. Bentley, Systems analysis and design methods. McGraw-Hill Professional, 1997.

[41] D. Buckingham, "Educación en medios," Alfabetización, Aprendizaje y Cultura, vol. 1, pp. 1–331, 2005.

[42] S. W. M. Overmars & K. Poels, "Singlemedium-versus multimedia campaigns: exposure effects on implicit versus explicit memory, brand attitude and purchase intention," Tijdschr. VOOR Commun., vol. 41 (2), 104p. 2013.

[43] M. G. Díaz-Antón, M. Pérez, A. Grimmán, & L. Mendoza, "Propuesta de una metodología de desarrollo de software educativo bajo un enfoque de calidad sistémica," Univ. Simón Bolívar, Caracas, Venez., vol. 1, 2006.

[44] C. M. Zapata-Jaramillo & M. Duarte, "El juego de la consistencia: una estrategia didáctica para la Ingeniería de Software," Revista Técnica Ingeniería de la Universidad del Zulia, Vol. 31 (1), pp. 1–10, 2008.

[45] B. Boehm, "A view of 20th and 21st century software engineering," in Proceedings of the 28th international conference on Software engineering, 2006, pp. 12–29.

[46] R. E. Fairley, "Educational issues in software engineering," in Proceedings of the 1978 annual conference, 1978, pp. 58–62.

[47] G. González-Calderón, C. M. Zapata-Jaramillo & R. A. Manjarres-Betancur, "Un juego para la enseñanza de métodos de desarrollo de software" Undécima Conferencia Iberoamericana en Sistemas, Cibernética e Informática: CISCI 2012 Sistemas, Cibernética E Informática. Memorias Volumen I, International Institute Of Informatics And Systemics , p.67 - 72 , 2012.

[48] G. Taran, "Using games in software engineering education to teach risk management," in Software Engineering Education & Training, 2007. CSEET'07. 20th Conference on, 2007, pp. 211–220.

[49] R. Cano, D. Astrid, et al., "Una propuesta de juego no tecnológico para la enseñanza de puntos de vista en el desarrollo de software," Tesis de Grado en Ingeniería de Sistemas, Universidad Nacional de Colombia, Medellín (Colombia), 2009.

[50] Anónimo, "Origami," 2013. Disponible en: http://en.origami-club.com/

[51] M. Callejas-Cuervo, L. Y. Castillo-Estupiñán, & R. M. Fernández-Álvarez, "Heler: Una herramienta para la ingeniería de requisitos automatizada", Entramado (ISSN 1900-3803), Vol. 6 (2), pp. 184-200, 2010

[52] C. M. Zapata & N. Carmona, "Un modelo de diálogo para la educción de requisitos de software," Dyna, vol. 164, pp. 209–219, 2010.

[53] C. M. Zapata & G. Giraldo, "El juego del diálogo de educción de requisitos", Revista de Avances en Sistemas e Informática, vol. 6 (1), pp. 105–113, 2009.

[54] E. Monsalve, V. Werneck, & J. Leite, "Evolución de un Juego Educacional de Ingeniería de Software a través de Técnicas de Elicitación de Requisitos," in Proceedings of XIII Workshop on Requirements Engineering (WER'2010), Cuenca, Ecuador, 2010, pp. 12–23.

[55] E. Monsalve, V. Werneck, & J. Leite, "Evolución de un Juego Educacional de Ingeniería de Software a través de Técnicas de Elicitación de Requisitos," Proceedings of XIII Workshop on Requirements Engineering (WER'2010), Cuenca, Ecuador, 2010, pp. 12–23.

[56] M. S. Tabares, "UNA REVISIÓN DE MODELOS Y SEMÁNTICAS PARA LA TRAZABILIDAD DE REQUISITOS," Revista EIA (ISSN 1794–1237), Número 6, p. 33–42, 2006.

[57] A. Oliveros, J. A. Zuñiga, R. Wehbe, S. del V. Rojo, & F. Sardi, "Enseñanza de elicitación de requerimientos," XVIII Congreso Argentino de Ciencias de la Computación, 2012.

[58] A. Bandura & D. C. McClelland, "Social learning theory," Gen. Learn. Corp. Libr. Congr. Cat. Card Number 75-170398, vol. 1, 1977.

[59] Á. M. C. González, P L. González, P M. Gómez, "'Diseño y aplicación de juegos para la enseñanza de ingeniería de software a nivel de pregrado en la Universidad de Medellín,'" " WEEF 2013 - Innovación en Investig. y Educ. en Ing. factores claves para la Compet. Glob., 2013.

[60] A. Baker, E. Oh Navarro, & A. Van Der Hoek, "An experimental card game for teaching software engineering processes," Journal of System Software, vol. 75 (1), pp. 3–16, 2005.

[61] C. M. Zapata & N. Carmona, "A Dialog Model For Software Requirements Elicitation," DYNA, vol. 77 (164), pp. 209–219, 2010.

[62] C. M. Zapata, G. G. Calderón, & D. Rivera, "PoV-game: puntos de vista mediante juegos," Revista de Ingeniería Universidad de Medellín, vol. 11, no. 20, pp. 115–126, 2012.