Revista EIA

UNIVERSIDAD EIA
Ser, Saber y Servir

Metodología lúdica para la enseñanza de la ingeniería de requisitos basada en esquemas preconceptuales

HÉCTOR JOSÉ HERNÁNDEZ-REINOZA[1]
CAMILO VILLOTA-IBARRA[1]
JOVANI ALBERTO JIMÉNEZ-BUILES[1]

1. Departamento de Ciencias de la Computación y de la Decisión Universidad Nacional de Colombia, Sede Medellín. Medellín, Colombia

Revista EIA
ISSN 1794-1237
e-ISSN 2463-0950
Año XVIII/ Volumen 18/ Edición N.35
Enero-Junio de 2021
Reia35001 pp. 1-15

Publicación científica semestral
Universidad EIA, Envigado, Colombia

PARA CITAR ESTE ARTÍCULO / TO REFERENCE THIS ARTICLE /
Hernández-Reinoza, H.J.; Villota-Ibarra, C.; Jiménez-Builes, J.A. (2021). Metodología lúdica para la enseñanza de la ingeniería de requisitos basada en esquemas preconceptuales. Revista EIA, 18(35), Reia35001. pp. 1-15.
https://doi.org/10.24050/reia.v18i35.1394

Autor de correspondencia:

Jiménez-Builes, J. A. (Jovani Alberto): Departamento de Ciencias de la Computación y de la Decisión Universidad Nacional de Colombia, Sede Medellín.
Correo electrónico: jajimen1@unal.edu.co

Recibido: 24-01-2020
Aceptado: 12-10-2020
Disponible online: 01-01-2021

Resumen

Los requisitos software son parte fundamental del proceso de ingeniería de software. Algunas veces, estas especificaciones de requisitos pueden ser inconsistentes, incompletas o simplemente incorrectas y no son detectadas a tiempo en la fase de ingeniería de requisitos. Estos requisitos son obtenidos a partir de la comunicación con los usuarios. Y esta comunicación, presentada en lenguaje natural, no siempre es interpretada adecuadamente. Este problema puede minimizarse con una mejor enseñanza del proceso de ingeniería de requisitos mediante el uso de una metodología diferente. Las metodologías tradicionales y los proyectos de clases por sí solos han demostrado no ser adecuados para la enseñanza del proceso de software, por lo que se propone una metodología lúdica para la enseñanza y aprendizaje de la construcción de requisitos software basada en Esquemas Preconceptuales.

Palabras Claves: Ingeniería de software, ingeniería de requisitos, esquemas preconceptuales, didácticas, lúdicas, enseñanza y aprendizaje.

Playful methodology for teaching requirements engineering based on preconceptual schemes

Abstract

Software requirements are an essential part of the software engineering process. Sometimes, such requirements specifications may be inconsistent, incomplete or simply incorrect and are not detected in time in the requirements engineering phase. These requirements are obtained from communication with users; because of the natural language in which this communication usually takes place, it sometimes could be incorrectly interpreted. This problem can be minimized with a better teaching of the requirements engineering process by using a different methodology. Traditional methodologies and class projects alone have proven not to be

suitable for teaching the software process; that is why a playful methodology for teaching and learning the construction of software requirements based on Pre-conceptual Schemes is proposed here.

Key Words: Software Engineering, Requirements Engineering, Preconceptual Schemes, didactic, playful, teaching and learning.

I. Introducción

La ingeniería de requisitos es la primera fase del proceso de ingeniería de software, en la cual los requisitos de usuario son recolectados, entendidos, y especificados (Wahono, 2003). Es en esta fase donde a través de un proceso de análisis iterativo se consiguen los requisitos, se validan y se especifican para definir con precisión qué se debe construir y documentar los resultados del análisis.

Aunque se resalta la importancia de una identificación efectiva de los requisitos, existe un número de dificultades para lograr este objetivo. Siendo la mala identificación de los requisitos una de las fuentes más significantes de no satisfacción de los clientes con los sistemas entregados (Macaulay, 1996), junto con las especificaciones de requisitos incompletas y cambio de requisitos durante el proceso de desarrollo (Herlea, 1999). Al mismo tiempo, el no crear y/o mantener apropiadamente la documentación de requisitos tiene un impacto en la calidad y resultado del proceso. Por lo tanto, la gestión del proceso de requisitos afecta no solo la fase de requisitos, sino también todo el desarrollo del proyecto (Chatzoglu y Macaulay, 1995).

La ingeniería de requisitos es sobre personas comunicándose con personas, por lo que las dificultades de comunicación, la falta de conocimiento apropiado y mutuo entendimiento se traduce directamente en malos entendidos para el diseño y desarrollo de las aplicaciones software (Herlea, 1999).

La ingeniería de software es una disciplina que aborda soluciones valiosas a problemas complejos, los cuales requieren de una buena combinación de información teórica y práctica (Sommerville, 2004). Sin embargo, la educación tradicional de la ingeniería de software con formación teórica no puede resaltar los problemas potenciales que los ingenieros de software deben afrontar de forma práctica.

Si bien la ingeniería de software ha evolucionado, su enseñanza ha presentado pocas modificaciones desde sus inicios. Se ha empleado para la formación de los ingenieros de software una combinación de clases expositivas y pequeños proyectos de aprendizaje, con los cuales se ha pretendido que los futuros ingenieros de software adquieran las destrezas necesarias para su ejercicio profesional. Este tipo de enseñanza, ampliamente centrada en el docente como fuente del conocimiento y como asesor o guía para los proyectos prácticos, viene siendo reevaluado debido a los problemas de participación e iniciativa que puede generar en los estudiantes, y estas dos características son ampliamente deseables en la formación de nuevos ingenieros de software (Zapata, 2007).

Como una forma de solución a estos problemas, se propone una metodología que posibilita la enseñanza de ciertos aspectos de la fase de ingeniería de requisitos como parte de la ingeniería de software, mediante el uso de un juego didáctico basado en el levantamiento de requisitos usando Esquemas Preconceptuales.

En el artículo se presenta en la Sección II, el marco teórico que fundamenta esta propuesta; en la III, la construcción de la metodología del juego y la Sección IV corresponde a las conclusiones.

https://doi.org/10.24050/reia.v18i35.1394

II. Materiales y Métodos

Ingeniería de Requisitos

La ingeniería de requisitos es la rama de la ingeniería del software que se encarga de la realización de actividades en el intento de entender las necesidades exactas de los usuarios de un sistema, y por consecuente traducir éstas a precisas funciones y/o acciones que serán usadas en el desarrollo del sistema, o bien, serán realizadas por el sistema al terminarse satisfactoriamente.

Esta área de la Ingeniería de Software implica todas las actividades dedicadas (como se ha mencionado anteriormente) a la extracción de datos de los entornos posibles, por lo que se lleva a cabo el proceso de la "elicitación", más conocido como "la educción", que no es más que la intuición o deducción de los recursos que se deberán sustituir por acciones de la solución informática, así como también, para precisar, se realiza el análisis y negociación de requisitos para derivar requisitos adicionales para dictaminar lo que posiblemente se necesite aun considerando que sea de menor importancia. Se crea la documentación de los requisitos como especificación, se validan los requisitos documentados enfrentándolos con las necesidades de usuario (González, 2010).

Suele tener varias fases de implementación, puesto que no es algo sencillo de realizar debido a su amplia cantidad de elementos a analizar. Por ello, se suele decir que las actividades que van de manera secuencial a la Ingeniería de requisitos son:

- Elicitación de requisitos
- Especificación de requisitos
- Verificación y validación de los requisitos

Estas fases del proceso de buscan cumplir la principal tarea de la Ingeniería de Requisitos, la cual consiste en la definición del proceso a seguir en la construcción de un software, y de facilitar la comprensión de lo que el cliente requiera. La obtención correcta de los requerimientos puede llegar a describir con claridad, sin ambigüedades, en forma consistente y compacta, el comportamiento de un sistema.

Dificultades de la ingeniería de requisitos

Muchos de los problemas comunes y de mayor seriedad asociados con el desarrollo software están relacionados con los requisitos y estos pueden identificarse en cada una de las fases del proceso de Ingeniería de Requisitos.

Los problemas de la elicitación de requisitos pueden ser agrupados y clasificados en tres categorías (Christel y Kang, 1992): problemas de alcance, problemas de entendimiento, y problemas de volatilidad. También conocidos como problemas en el "análisis del problema" y "entendimiento de las necesidades del usuario".

Siendo la fase de la especificación de requisitos donde se determinan y se definen las entradas, salidas, funciones, atributos del entorno del sistema, el problema es hacer los requisitos lo suficientemente detallados para ser bien entendidos sin la restricción excesiva del sistema y sin definir otros aspectos que es mejor dejar en otras fases del proceso de la ingeniería de software. En otras palabras, es buscar la cantidad justa y necesaria de especificidad, es decir, el nivel de ambigüedad correcto para ser resuelto más adelante. La técnica más común para documentación de requisitos es usar el lenguaje natural y escribir todo de forma estéticamente

organizada para que cumpla los ocho principios que un gran número de estándares recomiendan (Institute of Electrical and Electronics Engineers, 1998): correcto, sin ambigüedad, completo, consistente, clasificado según importancia o estabilidad, verificable, modificable y rastreable. Al mismo tiempo, si la descripción del requisito es demasiado compleja para el lenguaje natural se debe considerar escribir esa porción de requisitos con una metodología técnica.

De la tercera fase depende la construcción adecuada del sistema, debido a que esta se encarga de la confirmación continua de que el desarrollo va por el camino correcto y que los resultados son correctos, así como poder lidiar con los cambios durante el desarrollo. Esto implica que lo que se realice en cada una de las fases el proceso de Ingeniería de Requisitos debe estar tan bien estructurado que se puedan hacer verificaciones y validaciones lógicas y tener la suficiente flexibilidad para adaptarse al cambio.

Para agregar mayor complejidad a la situación, se reconoce que los problemas de la Ingeniería de Requisitos no pueden ser resueltos de una forma netamente tecnológica; el contexto social es más crucial que en las fases de diseño y programación. La información necesitada para el diseño del sistema está embebida en el mundo social de los usuarios y administradores, y tiende a ser informal y altamente dependiente de su contexto social para la interpretación. Una reconciliación de los aspectos socio-técnicos de la información es considerada como la esencia misma de la ingeniería de requisitos (Goguen, 1994).

Enseñanza de la ingeniería de software

Desde sus primeros pasos a finales de los años sesenta, la Ingeniería de Software ha venido despertando un creciente interés en la industria del software, dado que los problemas ligados con el desarrollo de estos productos intangibles se han vuelto cada vez más complejos; esta complejidad se debe a la gran demanda de este tipo de productos y al alto impacto que pueden alcanzar en el medio (Zapata, 2007).

Desde entonces, la enseñanza de la Ingeniería de Software se ha acometido por lo general con dos estrategias principales: las clases expositivas y la elaboración de proyectos prácticos en el área, generalmente en pequeña escala, y que sirven como pequeños laboratorios en ambientes excesivamente simulados para enseñar el desarrollo metodológico de software. Sin embargo, la industria y la academia son conscientes de las necesidades de modificación a la enseñanza tradicional de la Ingeniería de Software, ya que en el momento no se está promoviendo la formación en ciertas habilidades tan complejas y necesarias como la resolución de conflictos, la comunicación y el manejo de los recursos durante la ejecución de un proyecto de software, las cuales son necesarias para los ingenieros de software (Zapata, 2007).

Boehm (2006) sugiere, 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. Esta sugerencia coincide con la apreciación de Felder et al. (2000) 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 imparten, 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), debido a razones como la motivación, representatividad, interactividad y dinamismo, competencia

y seguridad (Zapata, 2007). Además, Klassen y Willoughby (2003) presentan otras razones para emplear los juegos como estrategias educativas:

- El conocimiento se devela al participar en el juego. Los participantes experimentan una sensación de conocimiento "obvio" una vez han participado en el juego, y esa experiencia permite un aprendizaje significativo mucho mayor.
- Entre mayor participación se da en el juego, mayor es el conocimiento adquirido. Sin importar si los participantes ganan o pierden el juego, el nivel de intensidad de su participación se refleja en los conocimientos que se logran en el mismo.
- Reduce los niveles de tensión de los participantes. En el caso de las clases expositivas, el temor de la evaluación actúa como una fuente de tensión para los estudiantes. Cuando se trata de participar en un juego, lo que menos importa en un momento dado es el resultado y la participación se logra de manera mucho más desinteresada que en una clase expositiva.
- Se pueden emplear materiales sencillos. No siempre es necesario un material tecnológicamente avanzado para realizar una clase mediante juegos. Con un tablero convencional y unos dados o unos cartones de juego se puede realizar este tipo de actividades. No es necesario disponer de hardware o software especializado

Esquemas Preconceptuales en la Elicitación de Requisitos

El ciclo de vida del desarrollo de software parte de una cuidadosa recolección de los requisitos de los interesados. Dentro de esa recopilación cobran importancia ciertos diagramas que compendian de manera gráfica la información que se puede capturar de los interesados. Los Esquemas Preconceptuales surgen como formas de representación del conocimiento y se pueden usar en esa recopilación de requisitos.

Un Esquema Preconceptual es una representación del conocimiento, relativa a un dominio particular, que emplea una simbología que los conocedores de ese dominio pueden validar. Además, es un diagrama que posibilita la comunicación entre personas con orientación técnica (como son los analistas en el proceso de desarrollo de software) y los conocedores del dominio (que, en términos del desarrollo de software se denominarán, en adelante, "interesados") (Zapata, 2007).

Según Zapata (2007), se prefiere el nombre "Esquema Preconceptual" pues representa un dominio de aplicación, expresando el conocimiento previo y definiendo la terminología propia del área, en lugar de establecer el conjunto de características previas de diseño de una solución informática. Así, un Esquema Preconceptual se constituye en un facilitador de la traducción del discurso inicial sobre el dominio y los diferentes esquemas conceptuales que se elaboran en las fases de análisis y diseño de una aplicación.

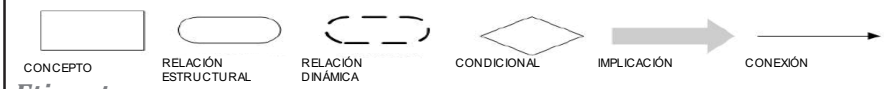
Sintaxis y Semántica

Los Esquemas Preconceptuales (EP) usan una notación similar a los Gráficos Conceptuales, con algunos símbolos que representan las propiedades dinámicas. Un EP es un diágrafo etiquetado (no necesariamente conectado) sin bucles ni múltiples arcos, compuesto por nodos de cuatro tipos conectados por arcos de dos tipos como se observa en la Imagen 1, con las siguientes restricciones:

Topología

- Un arco de *conexión* conecta un concepto a una relación o vice versa
- Un arco de *implicación* conecta una relación dinámica o condicional a una relación dinámica
- Cada *concepto* tiene un arco *incidental* (el cual es de tipo conexión que va hacia o desde una relación)
- Una *relación dinámica* tiene exactamente un arco de conexión entrante y uno saliente (incidental para los conceptos; puede tener cualquier cantidad de arcos de implicación incidental)
- Una *relación estructural* tiene exactamente un arco de entrada y uno o más arcos de salida (de tipo conexión, incidental para los conceptos)
- Un *condicional* no tiene arcos entrantes y uno o más arcos de salida (de tipo implicación, que van hacia relaciones dinámicas).

Imagen 1. Elementos sintácticos de los Esquemas Preconceptuales



Etiquetas

- Un arco de conexión no tiene
- Un arco de implicación tiene una etiqueta de si o no (si se omite, se asume un sí)
- Un concepto está etiquetado con un sustantivo que representa una entidad del mundo modelado. Diferentes nodos de concepto tienen diferentes etiquetas.
- Una relación dinámica está etiquetada con un verbo de acción, ej., pagar, registrar. Diferentes nodos dinámicos de relación pueden tener la misma etiqueta.
- Una relación estructural está etiquetada con el verbo es o tiene.
- Un condicional está etiquetado con una condición lógica para valores de cierto concepto, ej., puntuación > 3.

Semántica

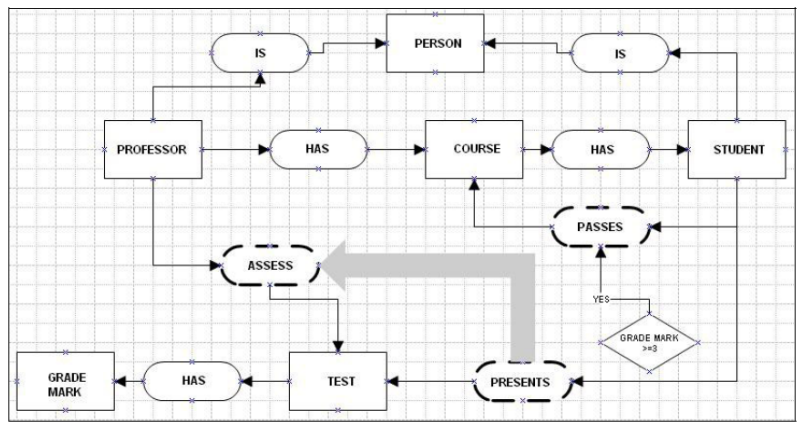
- Las conexiones expresan argumentos estructurales o relaciones: a groso modo, el sujeto y el objeto del verbo correspondiente. Un concepto puede participar en varias relaciones (secretaria -> imprime -> reporte, secretaria -> llama -> cliente, director -> emplea -> secretaria; aquí secretaria es el mismo nodo).
- Los conceptos representan personas (empleado, secretaria), cosas (documento, factura), y propiedades (dirección, teléfono). En la elicitación de requisitos, se puede imaginar que después se podrían convertir en cajas de

Revista EIA https://doi.org/10.24050/reia.v18i35.1394

diálogo mostradas al usuario según la categoría dada o representar alguna cosa dada, como campos en estas cajas de diálogo.

- Las relaciones estructurales expresan jerarquía de clases (secretaria es un empleado), propiedades (empleado tiene teléfono), relaciones de partes (carro tiene motor), etc. Un nodo de relación puede tener solo un sujeto y un objeto (secretaria -> imprime->reporte, contador->imprime->factura; estos son dos nodos de imprimir diferentes). En la elicitación de requisitos, se puede imaginar las propiedades como campos de texto o enlaces a cajas de diálogos correspondientes a sus propietarios.
- Las relaciones dinámicas expresan acciones que las personas pueden realizar (secretaria puede registrar la factura). En la elicitación de requisitos, se les puede imaginar cómo botones que los usuarios del software pueden presionar para realizar acciones correspondientes.
- Los condicionales representan prerrequisitos para realizar una acción. En la elicitación de requisitos, se pueden imaginar cómo inhabilitar o habilitar los botones correspondiente, dependiendo de si alguna condición es verdadera (contador puede pagar una factura después que la factura ha sido registrada) o si alguna otra acción ha sido realizada (contador puede pagar una factura solo si la secretaria ha registrado la factura).
- Los arcos de implicación tienen una etiqueta de si o no (si se omite, se asume un sí). Representa una implicación lógica entre eventos.

Imagen 2. Esquema Preconceptual de ejemplo del discurso



A partir de los Esquemas Preconceptuales se pueden obtener diferentes diagramas UML, como lo son los diagramas de clases, diagramas de comunicación, diagramas de máquinas de estado (Zapata y Arango, 2007), y diagramas de casos de uso para el proceso de especificación de requisitos (Chaverra, 2011). En la Imagen 2,

III. Resultados y discusión de la metodología lúdica

Imagen 3. Diagrama de Clases obtenido a partir del Esquema Preconceptual de ejemplo

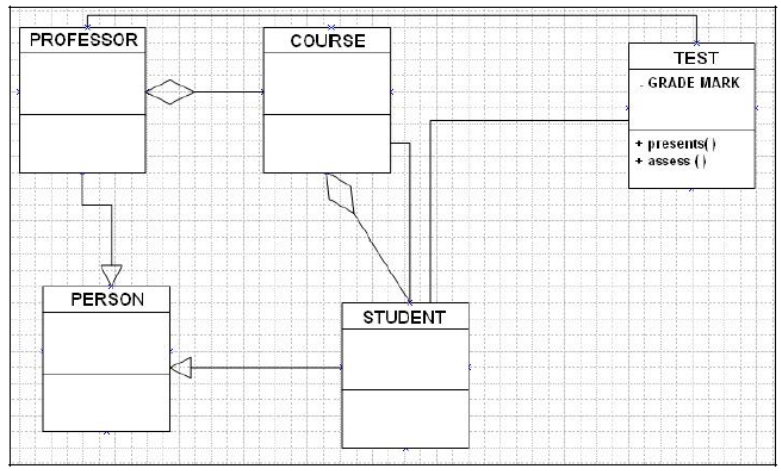


Imagen 4. Diagrama de Comunicación obtenido a partir del Esquema Preconceptual de ejemplo

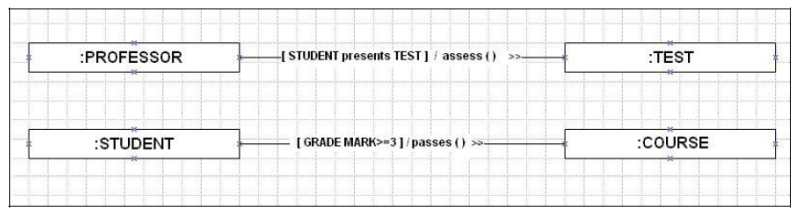


Imagen 5. Diagrama de Máquina de Estados obtenido a partir del Esquema Preconceptual de ejemplo

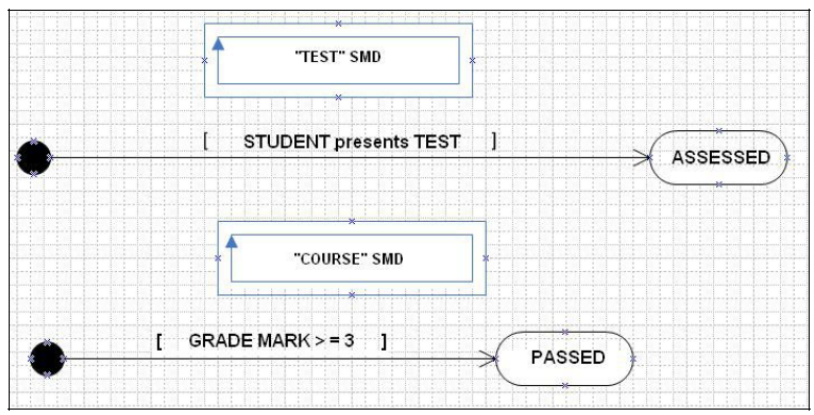


Imagen 3, i, e Imagen 5 obtenemos un ejemplo de los diagramas obtenidos a partir de un ejemplo básico de Esquema Preconceptual (Zapata, Gelbukh y Arango, 2006).

Revista EIA https://doi.org/10.24050/reia.v18i35.1394

Con el marco teórico de la Ingeniería de Software en mente, es posible definir el juego lúdico que se propone para complementar la enseñanza tradicional de la Ingeniería de Requisitos.

El juego de Elicitación de Requisitos Software con Esquemas Preconceptuales, Elipcon, es una simulación de la fase de ingeniería de requisitos. En esta fase, se procura que los jugadores conviertan un texto de requisitos, que está en lenguaje natural, a una forma de representación gráfica basada en los Esquemas Preconceptuales. El Esquema Preconceptual que se está desarrollando puede ser evaluado en cualquier momento para garantizar que lo que se está construyendo cumpla con las reglas de esta representación gráfica.

El propósito del juego es desarrollar en los jugadores la capacidad de relacionar el discurso del lenguaje natural con los requisitos de desarrollo software. De esta forma podrá desarrollar habilidades que le permitan identificar el tipo de elementos de un discurso que se transformará en un componente del software, al mismo tiempo, podrá mejorar sus habilidades para transformar el discurso hablado del cliente en un discurso escrito estructurado que le facilitará realizar las representaciones respectivas de los requisitos. También, aprenderá el uso de una herramienta de representación gráfica de ideas cómo lo son los Esquemas Preconceptuales, y conocer su aplicación en el proceso de ingeniería de requisitos.

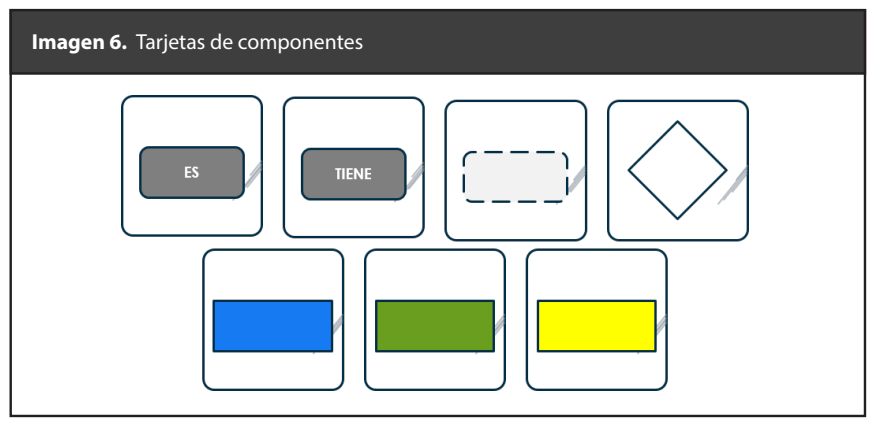
Objetivo de Elipcon

Obtener el mayor número posible de componentes, construir los componentes siguiendo las reglas de Esquemas Preconceptuales, y asignar a cada componente un valor obtenido del lenguaje natural de los requisitos.

Materiales

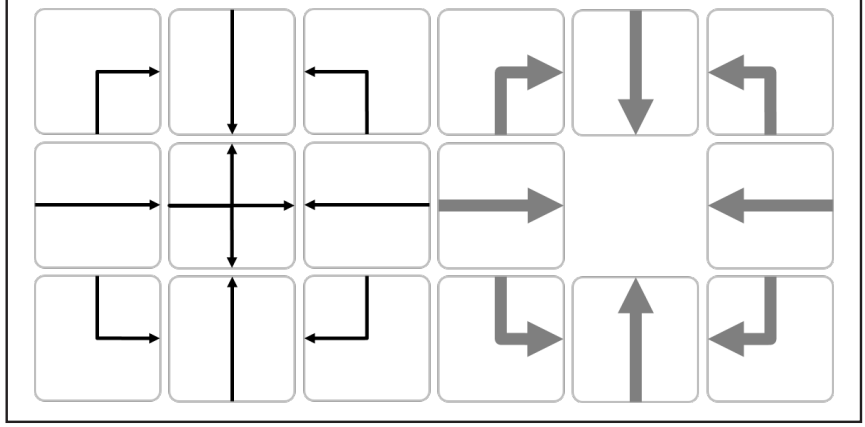
- Hoja de reglas básicas de Esquemas Preconceptuales: es una hoja que contiene las reglas descritas en el marco teórico para los Esquemas Preconceptuales. Establecen la relación entre los componentes y cómo deben ser usados.
- Tarjetas de componentes (conceptos, relaciones, y condicionales): son las tarjetas que se utilizan para construir los Esquemas Preconceptuales, y a las cuales se les debe asignar un valor proveniente de los requisitos en lenguaje natural. Estas se identifican por tener un borde oscuro. Ver Imagen 6.

Imagen 6. Tarjetas de componentes



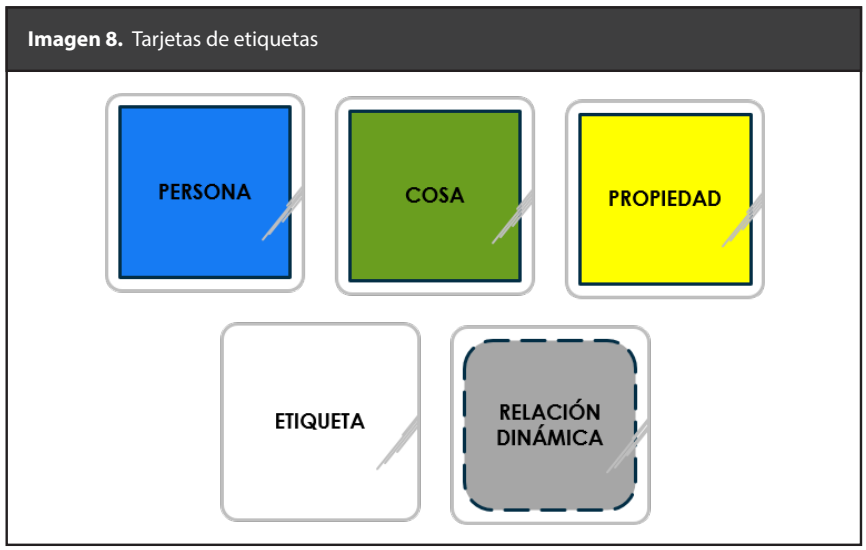
- Tarjetas de arcos (conexiones e implicaciones): son las tarjetas que se utilizan para conectar los componentes del Esquema Preconceptual. Estas pueden utilizarse seguida de otra tarjeta de arco para extender el alcance del mismo entre los diferentes componentes. Ver Imagen 7.

Imagen 7. Tarjetas de arcos



- Tarjetas de etiquetas (valores para etiquetas de conceptos y relaciones): son las tarjetas en las cuáles se escribirá el valor de los elementos que se identifican en la hoja de requisitos que se encuentra en lenguaje natural. Según el tipo de elemento se deberá usar una tarjeta respectiva para esto. Estas se identifican por tener un borde claro y no tener arcos. Ver Imagen 8.

Imagen 8. Tarjetas de etiquetas



Revista EIA https://doi.org/10.24050/reia.v18i35.1394

- Hoja de requisitos en lenguaje natural: contiene el texto en lenguaje natural de los requisitos obtenidos para el desarrollo de un software dado por un cliente

- Lápiz

Distribución y Proporción de Tarjetas

El número de conceptos y relaciones que se encuentran presentes en las diferentes hojas de requisitos en lenguaje natural no es constante, es decir, varía de acuerdo a aspectos como el nivel de detalle, tamaño del proyecto y aproximación deseada. Por lo tanto, no es posible definir un número fijo para la cantidad de cada una de las tarjetas que componen el juego. Sin embargo, se debe garantizar la disponibilidad de cada uno de los elementos de forma que toda la hoja de requisitos utilizada para cada partida pueda ser representada sin mayores problemas. Esto implica que debe mantenerse una proporción entre los diferentes elementos con el fin de que haya suficientes tarjetas para cada elemento a representar y que su cantidad no evite que se pueda obtener un elemento diferente.

Para establecer la proporción de elementos para el juego, se definen tres grupos diferentes formados por los siguientes elementos y sus cantidades:

- Grupo de Componentes: 3 tarjetas de Persona, 3 tarjetas de Cosa, 8 tarjetas de Propiedad, 3 tarjetas de Es, 3 tarjetas de Tiene, 2 tarjetas de Relación Dinámica, 1 tarjeta de Condicional. Para un total de 23 tarjetas para este grupo de elementos de la imagen 6.

- Grupo de Conexiones: 4 tarjetas de línea recta, 2 tarjetas de línea curva, 2 tarjetas de línea curva invertida, 1 tarjeta de encrucijada. Para un total de 9 tarjetas para este grupo de elementos de la imagen 7a.

- Grupo de Implicaciones: 4 tarjetas de línea recta, 2 tarjetas de línea curva y 2 tarjetas de línea curva invertida. Para un total de 8 tarjetas para este grupo de elementos de la imagen 7b.

La proporción propuesta es que por cada 10 conceptos que se identifiquen en la hoja de requisitos a utilizar se deberá utilizar para cada jugador un (1) grupo de componentes, ocho (8) grupo de conexiones, tres (3) grupo de implicaciones y 10 tarjetas de etiqueta para los 10 conceptos identificados. A esta proporción se le conocerá como un *Conjunto*.

Reglas del Juego

1. El guía del juego seleccionará una hoja de requisitos en lenguaje natural, identificará la cantidad de conceptos presentes y por cada 10 conceptos encontrados preparará un Conjunto para cada jugador.

2. Del total de conjuntos seleccionados se formarán dos pilas de tarjetas comunes para todos los jugadores, la pila de componentes y la pila de arcos. A cada jugador se le darán sus tarjetas de etiqueta y la hoja de requisitos.

3. El jugador dispondrá de 5 minutos para identificar cada elemento de la hoja de requisitos y escribirlo en las tarjetas de etiqueta. Después de llenadas las tarjetas de etiqueta, deberá revolverlas y crear su pila individual de tarjetas de etiqueta.

4. Los conceptos tienen un color según su tipo. Azul para personas, verde para cosas, y amarillo para propiedades. Las tarjetas de componentes y tarjetas de etiquetas para componentes se identifican según estos colores.

5. La primera tarjeta jugada por cualquier jugador deberá ser una tarjeta azul o verde.

6. Las tarjetas de etiqueta solo podrán ser utilizadas sobre una tarjeta de componente o de arco de implicación según el color que esta tenga.

7. Una tarjeta de arco puede ser utilizada al final de otra tarjeta de arco para extender la longitud del arco.

8. Cada jugador podrá tener hasta un máximo de seis tarjetas en su mano. En caso de necesitar más tarjetas deberá descartar permanentemente alguna tarjeta de su mano.

9. Cada jugador podrá realizar hasta tres acciones limitadas y tantas acciones ilimitadas como quiera durante su turno

10. Las acciones limitadas son: tomar una tarjeta de una pila, recoger una tarjeta jugada o acusar una tarjeta. Podrá tomar hasta dos tarjetas de arcos en una misma acción.

11. Las acciones ilimitadas son: descartar una tarjeta de la mano y jugar una tarjeta.

12. Una tarjeta ubicada en el área de juego será considerada una tarjeta jugada, y solo podrá ser recogida mediante una acción limitada

13. Si un jugador acusa una tarjeta de otro jugador, esta deberá ser revisada para verificar y validar que sea correcta. Si la tarjeta es correcta el acusador perderá la acción. Si la tarjeta es incorrecta quedará en posesión del jugador acusador y sumará como parte de sus puntos al final de la partida. Si la tarjeta incorrecta está ubicada sobre un componente, el jugador acusador también tomará posesión de esta.

14. Cada componente jugado correctamente equivale a un punto (tarjetas de componentes jugadas).

15. Cada valor asignado correctamente equivale a dos puntos (tarjetas de etiqueta jugadas).

16. Cada conexión realizada satisfactoriamente equivale a un punto (tarjetas de arco), es decir, que tengan un sujeto y objeto establecido. Solo la tarjeta de arco inicial en una secuencia continua de arcos se considerará como punto.

17. Cada tarjeta obtenida de otro jugador por acusación, equivale a dos puntos.

18. El juego finaliza cuando la pila de componentes se termina, o al finalizar la ronda en la que cualquier jugador haya llamado a verificación y validación (para lo cual su pila de etiquetas deberá estar vacía y no haber descartado una tarjeta de etiqueta). Para llamar a verificación y validación el jugador deberá tener una acción limitada disponible.

Revista EIA https://doi.org/10.24050/reia.v18i35.1394

19. Al finalizar la partida se totalizan los resultados y se determina el ganador.

Imagen 9. Ejemplo de tarjetas jugadas basado en el Esquema Preconceptual de Imagen 2. Esquema Preconceptual de ejemplo del discurso. Construcción parcial del discurso con algunas tarjetas de etiqueta sobre los componentes

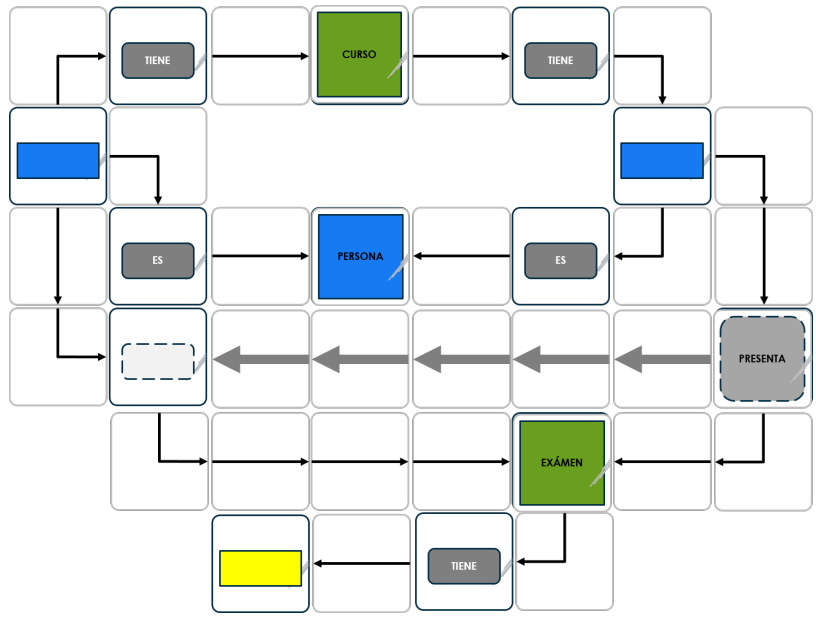
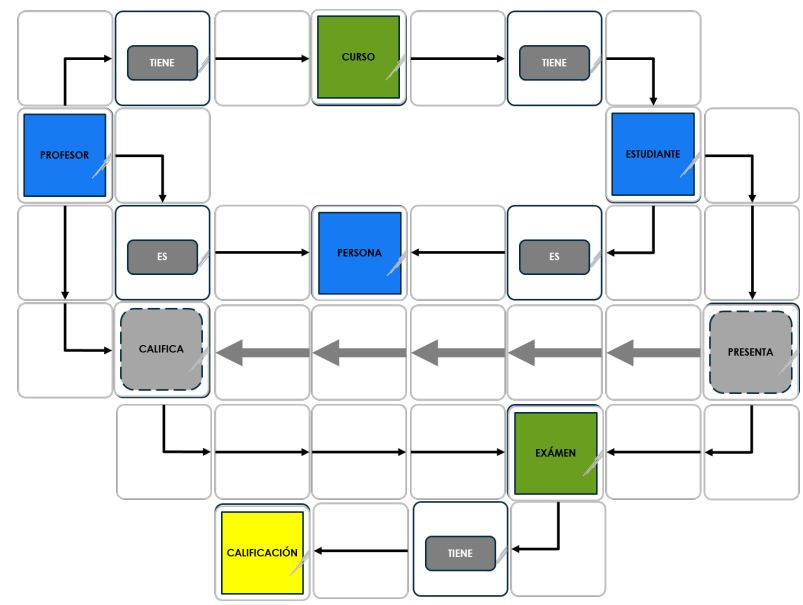


Imagen 10. Ejemplo de tarjetas jugadas basado en el Esquema Preconceptual de la Imagen 2. Esquema Preconceptual de ejemplo del discurso. Construcción parcial del discurso con todas las tarjetas de etiqueta sobre los componentes



IV. Conclusiones

La ingeniería de requisitos es una compleja disciplina que trata de formalizar las actividades relacionadas con obtener la especificación de requisitos formales del sistema a desarrollar a base de interactuar y negociar con el cliente. Realizar esta fase de la ingeniería de software es esencial para el éxito de un proyecto de desarrollo de software, en especial en las metodologías tradicionales de desarrollo de software, por lo que es crucial contar con un conjunto de requisitos muy estables sobre los que construir el resto del proyecto.

Sin embargo, la obtención de los requisitos depende de la información que se obtenga de parte de los interesados y de la formalidad con la que se represente esta información. Lo que resulta en una tarea que está ligada al proceso de comunicación entre personas y que presenta los problemas básicos de la comunicación como la pobre comunicación entre las partes y la falta de conocimiento apropiado y entendimiento mutuo. Adicionalmente, el lenguaje cotidiano no posee los formalismos apropiados para los procesos sistematizados por lo que es importante implementar metodologías de representación que faciliten el proceso de transformación del discurso cotidiano natural a una forma apta para el levantamiento de los requisitos para la fase de ingeniería de requisitos.

Como forma de ayudar a mejorar los problemas que se presentan en la fase de requisitos, y en la enseñanza y aprendizaje de la ingeniería de software en general, existen propuestas orientadas al uso de diferentes metodologías lúdicas, entre ellas los juegos, las cuales han demostrado ser complementos importantes de la enseñanza tradicional sin llegar a reemplazarla.

El juego Elipcon, es una propuesta lúdica para la enseñanza de ingeniería de requisitos y está basado en una herramienta de representación gráfica del conocimiento, los Esquemas Preconceptuales. Al ser una forma de representación fácil de entender para todas las partes interesadas del proceso, mejora la comunicación y aporta un formalismo que facilita la conversión de los elementos del discurso a elementos que pasarían a ser parte del software a desarrollar.

El juego Elipcon, permite mejorar y desarrollar competencias de comunicación al orientar a los jugadores a pensar en cómo guiar y transformar el discurso del cliente a componentes de Esquemas Preconceptuales, a detectar problemas en la representación del mismo, y a utilizar la representación para una automatización de las tareas de la fase de requisitos.

Referencias

Chatzoglu, P; Macaulay, L.A. (1995). Requirements capture and analysis: the project manager's dilemma, International Journal of Computer Applications in Technology 8.

Chaverra, J. (2011). Generación automática de prototipos funcionales a partir de esquemas preconceptuales, tesis (Título Magister en Ingeniería –Ingeniería de Sistemas), Colombia, Universidad Nacional de Colombia, Sede Medellín, 155 pp.

Christel, M., Kang, K. (1992). Issues in Requirements Elicitation, Technical Report CMU/SEI-92-TR-12, ESC-TR-92-012.

Crespo, A. (2017). Ingeniería de requisitos, Facultad de Ingeniería, Universidad de Cartagena. 23 pp.

https://doi.org/10.24050/reia.v18i35.1394

Goguen, J.A. (1994). Requirements Engineering as the reconciliation of social and technical issues, in Jirotka, M. and Goguen, J. (Eds.), Requirements Engineering: Social and Technical Issues, Academic Press, 165-200

Herlea, D.E. (1999). Challenges in Requirements Engineering.

Institute of Electrical and Electronics Engineers (1998), IEEE Recommended Practice for Software Requirements Specifications, IEEE Std 830-1998, Institute of Electrical and Electronics Engineers, New York, Unite States.

Macaulay, L. (1996). Requirements Engineering, 2nd Edition. Springer-Verlag London Limited, pp. 1-21.

Sommerville, I. (2004). Software Engineering. Pearson Editorial

Wahono, R. (2003). Analyzing Requirements Engineering Problems. IECI Japan Workshop 2003, pp. 55-58. Japan

Zapata, C., Gelbukh, A., Arango, F. (2006). Pre-conceptual Schema: a Conceptual-Graph-like Knowledge Representation for Requirements Elicitation. MICAI 2006: Advances in Artificial Intelligence: 5th Mexican International Conference on Artificial Intelligence, Apizaco, Mexico, November 13-17, 2006. Proceedings (pp.27-37)

Zapata, C. (2007). Los juegos de clase no tecnológicos como una estrategia didáctica para la enseñanza de la ingeniería de software. Trabajo de promoción presentado como requisito parcial para la promoción a la Categoría de Profesor Asociado. Universidad Nacional de Colombia, Medellín.

Zapata, C. y Arango, F. (2007). Construcción automática de esquemas preconceptuales a partir de lenguaje natural. Informe Final del Proyecto de Investigación. C.M. Zapata (Ed.), Medellín, Colombia.