

ISSN: 2007-5448



Red Temática Mexicana de Ingeniería de Software

Número Especial

Editado por:

Dr. Raúl Antonio Aguilar Vera

Dr. Reyes Juárez Ramírez

Dr. Juan Pablo Ucán Pech



RECIBE

Revista electrónica
DE COMPUTACIÓN, INFORMÁTICA, BIOMÉDICA Y ELECTRÓNICA

Vol. 4, No. 4

Índice

Editorial

Presentación I

Computación e Informática

Estructurando la forma de trabajo de Células de desarrollo de Software
usando Quali-Beh II

- Emmanuel Arroyo-López
- Teresa Ríos-Silva
- Alejandro Rico-Martínez
- Miguel Ehécatl Morales-Trujillo
- Hanna Oktaba

Un modelo para la solución de requerimientos no alineados: El caso del
Software lúdico para la divulgación III

- Héctor G. Pérez-González
- Rosa M. Martínez-García
- Francisco E. Martínez-Perez
- Sandra Nava-Muñoz
- Alberto S. Nuñez-Varela

Reducciones temporales para convertir la sintaxis abstracta del diagrama de flujo de tareas no estructurado al álgebra de tareas IV

Carlos Alberto Fernández-y-Fernández

José Angel Quintanar Morales

Impacto del aprendizaje basado en proyectos implementado en una empresa escolar de Base Tecnológica dedicada al desarrollo de Software V

María Angélica Astorga Vargas

Brenda Leticia Flores Rios

Jorge Eduardo Ibarra Esquer

Josefina Mariscal Camacho

Luis Enrique Vizcarra Corral



Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 2.5 México.

Presentación

La Revista electrónica de Computación, Informática, Biomédica y Electrónica (ReCIBE) presenta en la edición de este número especial, versiones extendidas de trabajos presentados en el Congreso Internacional de Investigación e Innovación en Ingeniería de Software (CONISOFT '15), organizado por la Red Temática Mexicana de Ingeniería de Software (REDMIS).

En su edición 2015, realizado en la ciudad de San Luis Potosí del 27 al 29 de abril, se recibieron treinta y dos artículos provenientes de veinte instituciones de México, Perú y Colombia, de los cuales, un Comité Técnico integrado por miembros de siete países: México, España, Argentina, Colombia, Uruguay, Estados Unidos y Brasil, seleccionó a los veinte mejores trabajos.

En esta edición, se presentan versiones extendidas de trabajos seleccionados que fueron presentados en cada una de las cuatro mesas de trabajo: Procesos Software, Desarrollo de Software, Enseñanza/Aprendizaje de la Ingeniería de Software y Áreas afines.

Agradecemos a los autores el esfuerzo por generar versiones extendidas de los trabajos presentados y damos la enhorabuena por su publicación en este número especial de RECIBE; así mismo, agradecemos a la editora de la revista su confianza para este número especial, el cual permite aumentar la visibilidad de los trabajos realizados en esta disciplina.

Dr. Raúl Antonio Aguilar Vera

Dr. Reyes Juárez Ramírez

Dr. Juan Pablo Ucán Pech



Esta obra está bajo una licencia de Creative Commons
Reconocimiento-NoComercial-CompartirIgual 2.5 México.

Estructurando la forma de trabajo de Células de desarrollo de Software usando KUALI-BEH

Emmanuel Arroyo-López

Instituto Tecnológico Superior de San Luis Potosí

emmanuel.arroyo@tecsuperiorslp.edu.mx

Teresa Ríos-Silva

Instituto Tecnológico Superior de San Luis Potosí

teresa.rios@tecsuperiorslp.edu.mx

Alejandro Rico-Martínez

Instituto Tecnológico Superior de San Luis Potosí

alejandro.rico@tecsuperiorslp.edu.mx

Miguel Morales-Trujillo

Grupo de Investigación KUALI-KAANS

Universidad Nacional Autónoma de México

migmor@ciencias.unam.mx

Hanna Oktaba

Grupo de Investigación KUALI-KAANS

Universidad Nacional Autónoma de México

hanna.oktaba@ciencias.unam.mx

Resumen: La incorporación de estudiantes recién egresados al campo laboral puede resultar en un proceso complicado, principalmente por la poca experiencia con la que cuenta el estudiante. En busca de reducir esta brecha, el Instituto Tecnológico Superior de San Luis Potosí, Capital, ha desarrollado un programa de incubación de células de desarrollo de software, conformadas por recién egresados de la carrera de Ingeniería en Sistemas Computacionales. Estas células incursionan en el desarrollo de proyectos de software con clientes reales. En este artículo se describe la experiencia de la célula Tic-Tac-S y la aplicación de KUALI-BEH como marco de trabajo para la definición, mejora y ejecución de un método de desarrollo de software propio, acorde a sus necesidades y contexto. La colaboración fructífera entre los creadores de KUALI-BEH y la célula trajo beneficios mutuos, así como más confianza en el programa y consecuente inversión en su infraestructura.

Palabras Clave: Células de Desarrollo de Software, Incubación de Empresas, Método, KUALI-BEH.

Using KUALI-BEH to organize ways of working of Software development cells

Abstract: Incorporating newly graduated students to the industry may turn out to be complicated due to the little experience they possess. Seeking to overcome this lack of experience, the San Luis Potosi Superior Tech Institute has developed a program of incubation of software development cells. These cells enroll recent graduates from Computer Systems Engineering major into software development projects with real customers. This paper presents the experience of the Tic-Tac-S cell that used KUALI-BEH as a framework for defining, improving and enacting their own software development method tailored according to their needs and context. This fruitful collaboration brought about improvement and recognition for both parties, as well as more confidence in the program and consequent investment in its infrastructure.

Keywords: Software Development Cell, Business Incubation, Method, KUALI-BEH.

1. Introducción

La ejecución de proyectos de desarrollo de software requiere que los integrantes del equipo de trabajo involucrado apliquen sus conocimientos y habilidades durante un periodo de tiempo establecido, de tal manera que el trabajo realizado permita el cumplimiento de los objetivos establecidos. Para lograr lo anterior, los equipos de trabajo siguen procesos de desarrollo de software apoyados en diversas metodologías, las cuales proporcionan una estructura adecuada o recomendada de la forma de trabajo a los participantes del proyecto.

Sin embargo, la problemática principal que enfrentan los estudiantes recién egresados que se integran a un ámbito laboral, es la falta de experiencia, en particular en el desarrollo de software. Situación que no es particular de una zona o contexto específico.

El Instituto Tecnológico Superior de San Luis Potosí, Capital (ITSSLP-C) no es ajeno a este fenómeno, es por ello que en busca de generar una ventaja competitiva a sus egresados, decidió implementar un proyecto para la conformación de células de desarrollo de software, formadas por alumnos, para que éstos adquieran experiencia en proyectos reales. Esta iniciativa cuenta con el apoyo de los departamentos de Vinculación y de académicos de la carrera de Ingeniería en Sistemas Computacionales (ISC) del Tecnológico.

A partir del surgimiento de la iniciativa, se formó el primer grupo de desarrollo de software: Tic-Tac-S. Tic-Tac-S es una célula de desarrollo de software conformada por egresados de la carrera de ISC del ITSSLP-C. Dicha célula es guiada por profesores de la ISC en la realización de proyectos con clientes reales.

Sin embargo, conforme el primer proyecto asignado a Tic-Tac-S avanzaba, el equipo de trabajo comenzó a enfrentarse al hecho de que la metodología usada se presentaba como una forma de trabajo estricta y pesada, con poca

posibilidad de adaptarse al ritmo y hábitos del grupo, dando como resultado la imposibilidad de definir un proceso acorde a las características y necesidades de la célula.

Lo anterior conllevó a incurrir en el uso de malos hábitos como: no documentar, “saltarse” pasos para el análisis e ir directo a la codificación, no establecer roles ni funciones. Con el paso del tiempo se volvió cada vez más complicada la corrección de la forma de trabajo, provocando que se generaran productos con poca calidad y los tiempos de desarrollo se extendieran fuera de lo establecido.

De tal forma el grupo Tic-Tac-S, comenzó la búsqueda y exploración de alternativas que permitieran la mejora de la célula. En conjunto con profesores del ISC la célula tomó la decisión de definir un proceso propio utilizando al marco de trabajo KUALI-BEH (Object Management Group, 2014a). La elección de este marco de trabajo se basó en las características ofrecidas por KUALI-BEH ya que se adaptaban a las necesidades de Tic-Tac-S, un equipo pequeño y sin un proceso explícito. La flexibilidad para definir un método propio y la viabilidad para la inclusión y aplicación de buenas prácticas obtenidas de estándares y modelos fueron las razones fundamentales para la elección de KUALI-BEH.

Este artículo tiene como objetivo describir la experiencia de aplicar KUALI-BEH dentro de la célula de desarrollo de software Tic-Tac-S para la definición y mejora de un método propio. Esta experiencia de colaboración entre Tic-Tac-S y el grupo de investigación KUALI-KAANS, creadores de KUALI-BEH, se gestó entre 2013 y 2014.

El presente documento se organiza de la siguiente manera: en la sección 2 se presentan los antecedentes tanto de Tic-Tac-S como de KUALI-BEH, en la sección 3 se describe la implementación de KUALI-BEH en el grupo de desarrollo de software Tic-Tac-S, en la sección 4 se presentan los resultados

de esta experiencia. En la sección 5 se muestran las lecciones aprendidas y para finalizar en la sección 6 se concluye.

2. Antecedentes

En esta sección se describirán tanto el contexto de Tic-Tac-S dentro del ITSSLP-C, como el origen de KUALI-BEH y los elementos que resultaron más importantes para las necesidades de Tic-Tac-S.

2.1 El ITSSLP-C y Tic-Tac-S

El ITSSLP-C fue creado el 05 de julio del 2003 (ITSSLP-C, 2014). En la actualidad oferta las carreras de: Ingeniería en Administración; Ingeniería en Mecatrónica; Ingeniería Industrial; e Ingeniería en Sistemas Computacionales. La población estudiantil al año 2015 es de 1400 alumnos, de los cuales 250 son del área de sistemas computacionales con 7 generaciones de egresados (ITSSLP-C, 2014).

Son profesores de esta carrera los interesados en incubar células de desarrollo de software con estudiantes recién egresados. Con la intención de que los alumnos, que formen parte de estas células, participen de manera activa en proyectos con clientes reales. Este enfoque intenta proveer a los alumnos de un panorama más cercano a lo que se enfrentarán una vez fuera de la carrera.

Como se mencionó anteriormente, Tic-Tac-S es una de éstas células de desarrollo de software. Tic-Tac-S en su inicio estuvo conformada por 4 alumnos que cursaban 8° o 9° semestres. Los roles de mayor peso, como el líder técnico y administrador de proyectos fueron tomados por 2 profesores del ITSSLP-C.

La primera experiencia de esta célula fue el desarrollo del proyecto Sistema administrativo para una institución de educación superior. Este proyecto tenía

como objetivo desarrollar los módulos del departamento de control escolar de dicha institución y fue desarrollado del mes de Abril del 2012 al mes de Octubre del 2013. Si bien, al término de este proyecto se logró la implementación de los módulos requeridos, se dejó ver la clara necesidad de contar con un proceso propio que se ajustara al tamaño del equipo, la experiencia y conocimiento de sus integrantes y que fuera lo suficientemente completo para atender de manera adecuada los proyectos del cliente.

Es por estas razones que surge el interés del grupo de trabajo en buscar una alternativa para expresar las maneras de trabajo reales del equipo.

2.2 Trabajos relacionados

En la literatura existen algunas alternativas que permiten la definición y modelado de procesos de software. Por ejemplo, el Object Management Group (OMG) tiene definido desde 2008 la versión 2.0 del Software and Systems Process Engineering Meta-model (SPEM) (Object Management Group, 2008), que es tanto un marco de trabajo como un meta modelo de procesos de ingeniería. SPEM describe un lenguaje y un esquema de representación que expresa formalmente métodos y procesos reutilizando algunos aspectos del Lenguaje de Modelado Unificado (UML) (Object Management Group, 2011).

Sin embargo, con este enfoque sigue existiendo la necesidad de especificar los fundamentos, tanto teóricos como del contexto de uso, de esas prácticas, que en el contexto de la célula de desarrollo aún no estaban identificados.

Por otra parte Humphrey propuso en (Humphrey, 2010) un proceso para equipos de alto desempeño que desarrollan software, dividido en cuatro grupos de actividades: preparación de las tareas, tareas de desarrollo, tareas de administración y actividades de evaluación.

Sin embargo, dada la generalidad del modelo, es difícil saber qué prácticas serían las adecuadas para poder construir los métodos asociados al proceso y

llevarlo a cabo. Más aún, resulta complejo delimitar que puede ser incluido dentro de cada grupo y que siga considerándose una actividad válida.

Además de estas propuestas se pueden mencionar a modelos de referencia de procesos, tales como CMMI (Software Engineering Institute, 2010), o estándares ISO/IEC como lo son el 24744 (International Organization for Standardization, 2007), 12207 (International Organization for Standardization, 2008) y 29110 (International Organization for Standardization, 2011). No obstante la naturaleza de estos compendios de buenas prácticas de la industria no se ajustaba a la de la célula de desarrollo, razón por la cual se decidió investigar otras alternativas, búsqueda que desembocó en KUALI-BEH.

2.3 KUALI-BEH

KUALI-BEH es un marco de trabajo para la expresión, comparación y mejora de las formas de trabajo de los practicantes de Ingeniería de Software involucrados en proyectos de software. Este marco de trabajo forma parte del estándar ESSENCE – Kernel and Language for Software Engineering Methods (Object Management Group, 2014b) del OMG.

KUALI-BEH describe los conceptos comunes y relaciones presentes en cualquier proyecto de software (Object Management Group, 2014a). Los ingenieros de software, o practicantes, que están involucrados activamente en proyectos de software son el principal objetivo de este marco de trabajo. KUALI-BEH consta de 2 componentes principales:

- Vista Estática: A través de conceptos comunes presentes en cualquier proyecto de software, se definen los elementos básicos para construir prácticas que definan la forma de trabajo de un equipo de trabajo. Permitiendo a sus integrantes expresar un método organizado que cumpla con ciertas propiedades.
- Vista Operacional: Ésta abarca la ejecución de los métodos definidos mediante los elementos de la vista estática.

KUALI-BEH es una herramienta que permite la estandarización de términos y conceptos para la ejecución de proyectos de software. Además define una estructura de trabajo enfocada a los conocimientos y habilidades que posee cada participante dentro de un proyecto de software.

KUALI-BEH guía a los participantes en proyectos de software a un cambio estratégico sobre sus hábitos de desarrollo, para ser adaptados a una forma de trabajo estructurada, la cual permita sustentar con una base de conocimiento que pueda ser útil para los involucrados en el proyecto.

Al usar KUALI-BEH el grupo Tic-Tac-S buscó cubrir la necesidad de contar con un método propio que se adaptara a su forma de trabajo, la cual está definida directamente por sus integrantes y su característica principal es la de ser recién egresados con poca experiencia en el desarrollo de software. Para fines de este artículo, la percepción de Tic-Tac-S sobre este marco de trabajo es presentada en la Figura 1.

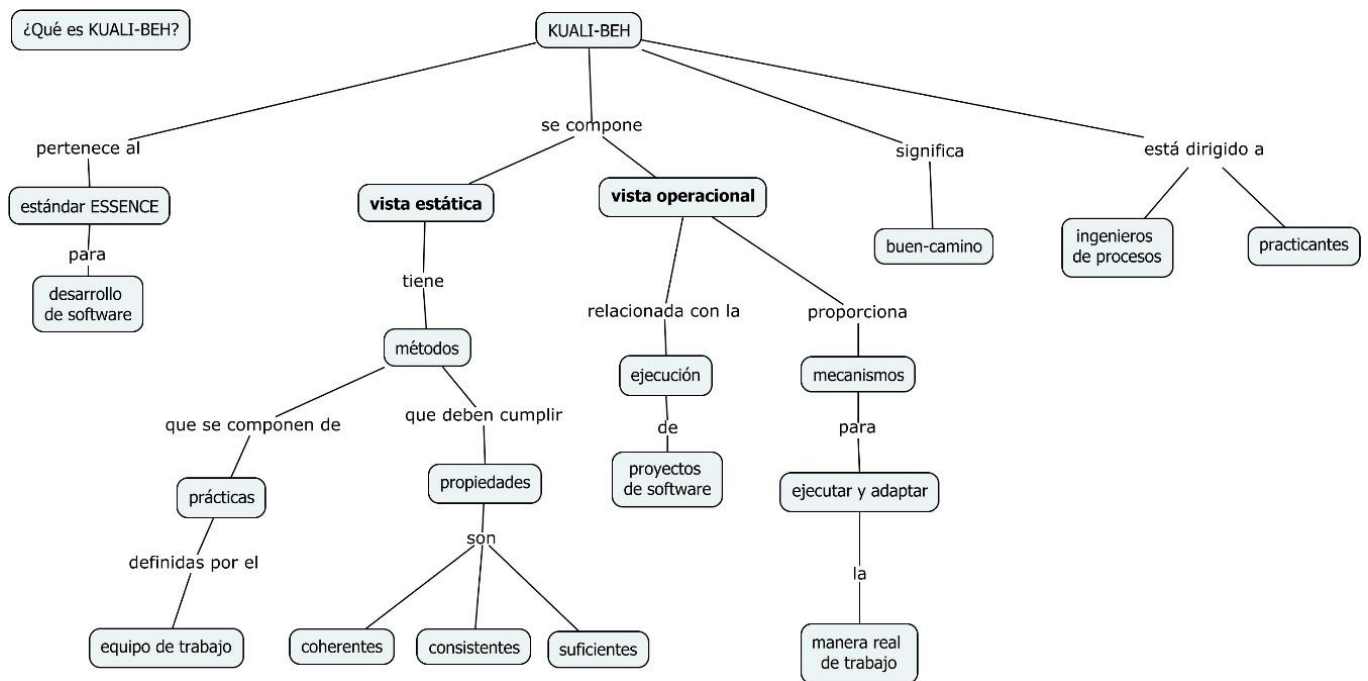


Figura 1.
 Mapa conceptual sobre KUALI-BEH.

3. Implementación de KUALI-BEH en una célula de desarrollo del ITSSLP

Un webinar ofrecido por Software Gurú fue el medio por el cual Tic-Tac-S tuvo el primer contacto con KUALI-BEH, posterior a ello, se dio un primer acercamiento entre el ITSSLP-C y KUALI-KAANS para formalizar una colaboración que permitiera a Tic-Tac-S implementar el marco de trabajo y así atender las necesidades identificadas.

La colaboración se planeó a través de un caso de estudio que constó de 3 fases que permitieran alcanzar los siguientes objetivos:

1. Demostrar la suficiencia de los elementos que conforman una Práctica para describir la manera de trabajo de los practicantes.
2. Medir la factibilidad para el practicante de expresar sus prácticas tácitas utilizando el concepto de Práctica.
3. Identificar el valor obtenido para la organización consecuencia de definir sus propias Prácticas.
4. Medir el esfuerzo requerido para la capacitación del equipo de trabajo.

En las siguientes subsecciones se describen cada una de las fases del caso de estudio.

3.1 Fase 1

La fase 1 del caso de estudio se realizó en el periodo que comprendió del 30 de Abril al 02 de Octubre del 2013 y tenía como objetivos:

- Identificar la manera de trabajo de los practicantes a ser modelada como Práctica(s)
- Expresar la manera de trabajo de los practicantes como Práctica(s)
- Acordar que la(s) Práctica(s) representa(n) la manera de trabajo de los practicantes

Una vez que el grupo de trabajo Tic-Tac-S conoció los objetivos del caso de estudio éstos se dieron a la tarea de estudiar KUALI-BEH para adquirir y comprender a mayor detalle sus elementos. Actividad que involucró una sesión-taller en línea entre Tic-Tac-S y KUALI-KAANS, en donde se clarificaron los conceptos contenidos en la documentación oficial y se dio respuesta a algunas dudas surgidas durante la primera actividad de trabajo.

Además se presentó la plantilla de Práctica de KUALI-BEH como ejemplo para comenzar con la definición de la manera de trabajo real de Tic-Tac-S. La plantilla de práctica fue, a partir de ese momento, el mecanismo principal para expresar las maneras de trabajo de Tic-Tac-S. Derivado de esta actividad se creó en conjunto con los investigadores la primera práctica de Tic-Tac-S, ver Figura 2.

Una vez entendida la forma en que se manejaría el llenado de las plantillas de prácticas, el grupo de trabajo generó un listado de prácticas identificadas que Tic-Tac-S realizaba de manera implícita durante sus proyectos de software.

Basados en ese listado se procedió a la generación de un diagrama que ilustrara la relación entre las prácticas definidas y las etapas en que se agruparían cada una de ellas para poder controlar de mejor manera el avance del método. Una vez que ya se tenía el diagrama de relación de las prácticas se fueron detallando una por una, en esta fase se encontraron diversos obstáculos tales como la confusión de conceptos para el llenado de la plantilla, específicamente sobre la definición de lo que es una actividad y lo que es una tarea, haciendo este proceso más lento de lo planeado, por lo cual se solicitó una comunicación más constante con el equipo de KUALI-BEH para aclarar estos puntos.

Aclarados los puntos anteriores, se continuó con el llenado de las plantillas de manera secuencial, sin embargo al continuar con este proceso, el grupo se dio cuenta de que era necesario agregar algunas prácticas que no se habían considerado, esta acción derivó en la modificación del diagrama de prácticas.

Basados en el listado de prácticas y el diagrama del método, se continuó con las revisiones sobre los objetivos, entradas y salidas, actividades y tareas de las prácticas para afinar cada práctica.

El proceso de revisión de las prácticas se daba de manera electrónica, siendo el equipo de KUALI-BEH quien hacía las observaciones pertinentes sobre las cuales trabajaría el grupo con el fin de ganar una nueva versión de cada práctica hasta lograr una versión definitiva.

Es importante mencionar, que las observaciones se centraban en el uso correcto de los conceptos y de la plantilla y no en la forma de trabajo que Tic-Tac-S expresaba.

Al concluir esta fase los integrantes del grupo de desarrollo de software definieron su propio método de trabajo, el cual cumplía con las propiedades de coherencia, consistencia y suficiencia establecidas en KUALI-BEH. Es decir, el método era coherente porque todas sus prácticas contribuían al logro del propósito del método, era consistente porque todas las entradas y salidas que requerían sus prácticas eran proveídas y/o consumidas por alguna otra práctica del método, finalmente era suficiente porque el conjunto de prácticas permitía cubrir totalmente el propósito para el que el método fue creado.

Las prácticas ayudaron a plasmar la forma de trabajo de Tic-Tac-S y en consecuencia permitió a sus integrantes tener una mejor visibilidad de su trabajo y así poder controlarlo. Además, un beneficio extra de definir el método, fue la creación de una lista maestra de productos de trabajo que se generaban a lo largo del proyecto. Esto permitió comenzar con la construcción de plantillas de apoyo específicas para cada producto de trabajo.


		Práctica	
Análisis de Requerimientos			
Objetivo			
Entender la funcionalidad de los requerimientos para poder proponer una solución.			
Entrada		Resultado	
Lista de Requerimientos [Generada]		Esquema del Sistema [Inicial]	
Criterios de Verificación			
El esquema del sistema que se generó cubre las necesidades esperadas por el cliente.			
Guía			
Actividad 1		Analizar la lista de requerimientos	
Input		Output	
Lista de requerimientos		Boceto del sistema	
Tareas (opcional)	Herramientas (opcional)	Conocimientos y Habilidades	Métricas
<ul style="list-style-type: none"> 1. Definir la navegación del sistema 2. Definir las pantallas del sistema 		<ul style="list-style-type: none"> • Análisis • Creatividad 	
Actividad 2		Diseñar Esquema del sistema	
Input		Output	
Boceto del sistema		Esquema del sistema [Inicial]	
Tareas (opcional)	Herramientas (opcional)	Conocimientos y Habilidades	Métricas
<ul style="list-style-type: none"> 1. Cotejar el esquema diseñado con la lista de requerimientos 	<ul style="list-style-type: none"> • Edraw 	<ul style="list-style-type: none"> • Manejo de Herramientas de diseño 	

Figura 2.
Práctica de Análisis de Requerimientos.

3.2 Fase 2

La Fase 2 del caso de estudio se dio del 03 de Octubre del 2013 al 01 de Abril del 2014. En esta fase se establecieron como objetivos los siguientes:

- Utilizar las Prácticas en proyectos de software por los practicantes como su forma de trabajo.
- Adaptar y mejorar las Prácticas en base a la experiencia, el conocimiento e influencia externa recabada por los practicantes.

En esta fase, Tic-Tac-S se enfrentó a un nuevo proyecto de desarrollo en el que se implementó y puso a prueba el método que se definió en la Fase 1. El proyecto elegido fue Ripos cuyo cliente fue el administrador de la cafetería del Tecnológico. Ripos consistió en la generación de una aplicación web que permitía administrar los ingresos, egresos e inventario de la cafetería. La aplicación resultante estuvo compuesta por ocho módulos que ayudaron a mejorar el control del negocio.

Durante esta fase el grupo comprendió que el método aún está inmaduro y era necesario realizar algunos ajustes de la versión inicial de las prácticas. En consecuencia la versión inicial del método sufrió cambios en las siguientes prácticas:

- Se consideró que la práctica Jerarquización de controles (Etapa 2), dada su baja complejidad podía ser considerada como una actividad de la práctica Definición de funcionalidad de módulos.
- Por otra parte, la práctica Identificación de fuente de datos (Etapa 2) podía concatenarse con la práctica Análisis del entorno del sistema.

Tic-Tac-S realizó ajustes a las prácticas anteriores utilizando las operaciones propias de KUALI-BEH, en particular se aplicaron las operaciones de fusión y concatenación. Estas operaciones, aplicadas correctamente, permiten mantener las propiedades originales del método.

Siempre que era necesario ejecutar alguna de éstas operaciones se registraban los factores causantes de la adaptación en una plantilla específica para este fin. Es de destacar la flexibilidad de las operaciones que permitieron llevar a cabo la adaptación.

Una vez terminada esta fase, el grupo obtuvo un método mejorado y probado durante el desarrollo de un proyecto de software. Este método pasó a formar parte de la base de conocimiento del grupo.

3.3 Fase 3

La última fase del caso de estudio se llevó a cabo del 02 de Abril al 24 de Octubre del 2014. El objetivo establecido fue:

- Adoptar las Prácticas como una manera rutinaria de trabajo.

En esta fase se inició un nuevo proyecto llamado Stand, que consistía en la generación de un sistema modular para el diseño y cotización de stands comerciales mediante herramientas web.

Durante el proyecto Stand se retomó el método desarrollado por el grupo Tic-Tac-S y se incorporaron nuevos miembros al equipo, a quienes en un primer momento se les capacitó en los conceptos básicos definidos por KUALI-BEH. Posterior a esta capacitación se comenzó con la ejecución del método y sus prácticas en el proyecto Stand, es así como se logró incorporar a los nuevos miembros a la forma de trabajo establecida en Tic-Tac-S.

En esta fase se logró que los nuevos integrantes a la célula comprendieran y adoptaran de manera adecuada el método de trabajo de Tic-Tac-S. Además, para fines del proyecto Stand se redujo el tiempo de desarrollo en un 10% al estimado, lo anterior derivó de dos razones principales: una ejecución cuidadosa y del método; y el involucramiento de la empresa Stand con el equipo de desarrollo conforme se avanzaba en el proyecto. Cabe resaltar que

las métricas recogidas durante este proyecto tuvieron que ver con el tiempo que tomó la ejecución de las prácticas.

Al término de la fase, la célula fue consciente de que el método y los hábitos de trabajo de sus integrantes habían madurado, sin embargo aún era necesario mejorar, situación que se planteó lograr mediante el desarrollo de más proyectos.

4. Resultados

Una vez concluidas las 3 fases del caso de estudio, se pudieron identificar los siguientes beneficios:

- El grupo de trabajo Tic-Tac-S cuenta con un método de desarrollo de software propio, ver Figura 3. Dicho método fue definido acorde a las necesidades y capacidades de los miembros que conforman la célula de desarrollo, permitiendo la consecución exitosa de los proyectos que ha enfrentado.
- Una vez que se identificó la necesidad de adaptar al método definido, a raíz de aplicarlo en el desarrollo de un nuevo proyecto, las operaciones de adaptación permitieron modificar de manera ordenada al método.
- Además permitió generar evidencia de los cambios realizados así como la justificación de los mismos. Dichas modificaciones dieron como resultado una nueva versión del método, ver Figura 4.
- Actualmente se cuenta con 23 prácticas documentadas para las cuales el número total de horas destinadas a su creación fue de 44, es decir se requirió en promedio de 115 minutos para documentar cada práctica.


		Método
Tic-Tac KUALI-BEH		
Propósito		
Desarrollar módulos completos para conformar sistemas de software.		
Entrada		Resultado
Solicitud del Cliente Datos Generales del Cliente Formatos del Equipo		Manuales de Usuario Manual Técnico Trámites Administrativos Sistema [Liberado] Contrato de Satisfacción [Firmado]
Conjunto de Prácticas		
<p><i>Etapa 1</i></p> <p>1.1 Recopilación de Información 1.2 Lista de Requerimientos 1.3 Análisis de Requerimientos 1.4 Modelo del Sistema 2.1 Definición de funcionalidad de módulos 2.2 Establecer Restricciones de los módulos 2.3 Análisis del Entorno del Sistema 2.4 Plan de Desarrollo</p> <p><i>Etapa 2</i></p> <p>3.1 Diseño de Plantillas 3.2 Generación de Pantallas 3.3 Validación de pantallas 4.1 Jerarquización de Controles 4.2 Identificación de Fuentes de Datos 4.3 Identificación de Código Reusable 4.4 Codificación de Pantallas 4.5 Verificación General de Controles</p> <p><i>Etapa 3</i></p> <p>5.1 Diseño de Pruebas 5.2 Ejecución de Pruebas 5.3 Análisis y Corrección de Defectos 6.1 Definir la Estrategia de Migración 6.2 Producir Estrategia de Migración</p> <p><i>Etapa 4</i></p> <p>7.1 Liberación del Sistema 7.2 Documento de liberación del sistema</p>		

Figura 3.

Método documentado.

- El método ha sido seccionado en 7 etapas, las cuales han permitido un mejor control sobre la ejecución del mismo, además de que esta nueva división hace más claro para el resto del equipo y el cliente qué avance se tiene.
- El método de desarrollo de software definido ha sido utilizado en dos nuevos proyectos con la integración de dos nuevos elementos al grupo, sin que esto

haya resultado en perjuicio ni para el equipo ni para el producto desarrollado.

- El tiempo destinado para la capacitación de los nuevos elementos fue de 7 horas.
- La participación de los integrantes del equipo en la definición del método logró elevar el compromiso del grupo en la realización de las actividades.

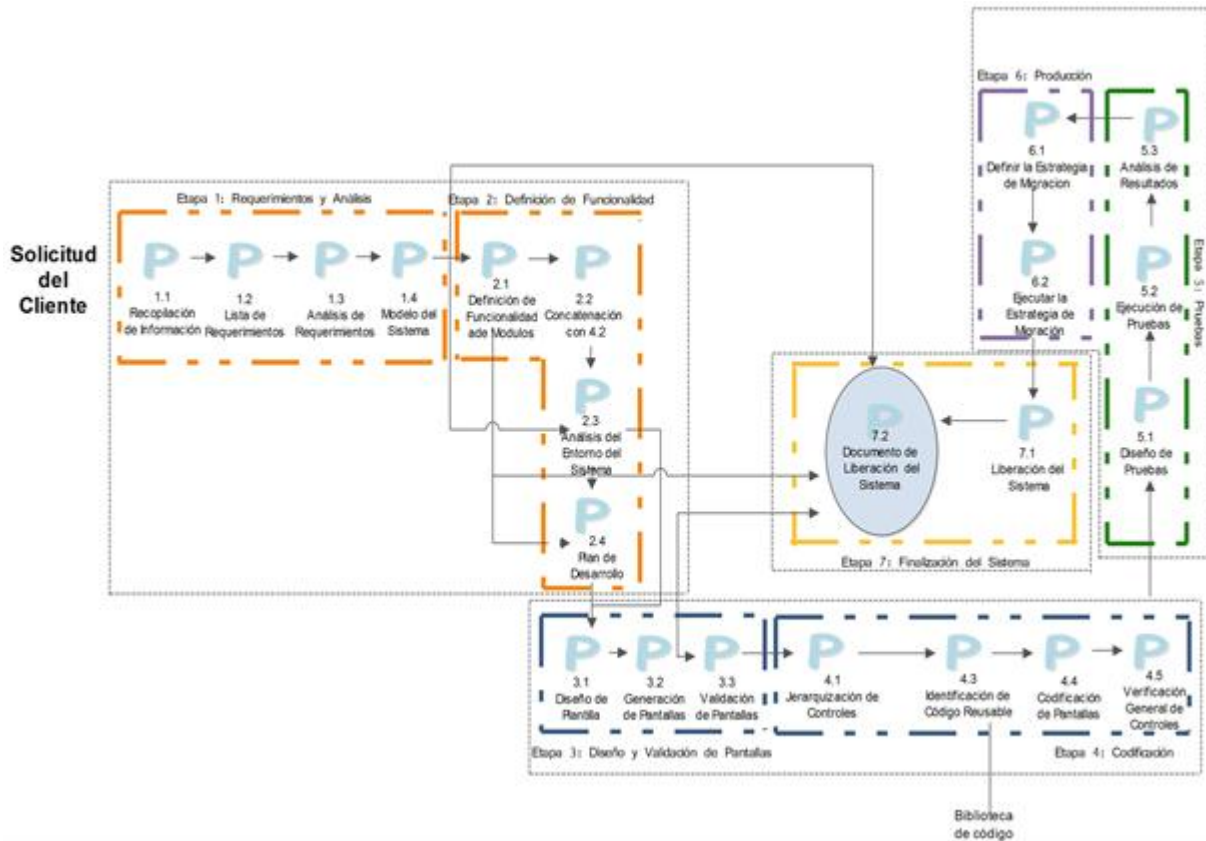


Figura 4.

Segunda versión del método Tic-Tac-S.

Entre las desventajas y oportunidades de mejora se pueden mencionar:

- Para los no iniciados en KUALI-BEH es necesaria una guía que ayude en la delimitación del alcance de la plantilla de prácticas.

- Es necesaria una herramienta electrónica que permita gestionar y compartir la base de conocimiento generada en las organizaciones que implementen KUALI-BEH.
- Sería deseable contar con una herramienta que permitiera llevar el control del método definido, así como apoyar en la adaptación de sus prácticas.

5. Lecciones aprendidas

Las lecciones aprendidas durante esta experiencia fueron las siguientes:

Como Tic-Tac-S, esta experiencia nos permitió darnos cuenta que el proceso utilizado, antes de KUALI-BEH, comenzaba directamente en la línea de programación, únicamente con los datos que proporcionaba el cliente en la primera o segunda entrevista, esto reflejaba la falta de una forma de trabajo estructurada y dirigida a generar productos de software con calidad, y en consecuencia carecía de un orden bien definido.

Al conocer y comprender KUALI-BEH resultó evidente que para generar una estructura de trabajo óptima es de suma importancia apearse a un orden y a un lenguaje común entre todo el equipo.

KUALI-BEH permitió canalizar el conocimiento de los integrantes en un proyecto. Dentro de las organizaciones debe existir una forma de trabajo significativa que identifique el conocimiento de sus miembros. Lo anterior permitirá definir los métodos adecuados a los diferentes tipos de proyectos que enfrenta la organización y sobre los cuales pueda sustentar la calidad de sus productos de software.

Para el contexto de Tic-Tac-S se observó que un equipo de trabajo debería incluir entre 4 y 6 integrantes para un mejor desempeño del método. Número que a lo largo de los proyectos realizados se ha visto como ideal de acuerdo a las características de los proyectos que atiende el ITSSLP-C. Los integrantes

de las células deberán contar con las habilidades y conocimientos necesarios para la aplicación y desarrollo de los productos de software.

Muchas de las ocasiones cuando se sigue una metodología de trabajo que está previamente definida, los equipos de trabajo enfrentan una cuesta arriba al momento de intentar adoptarlo y/o adaptarse. En este caso se logró definir una metodología propia que contempla necesidades particulares para el desarrollo de proyectos y productos de software.

Las prácticas que se diseñaron a través de KUALI-BEH, y que definen la forma de trabajo real de Tic-Tac-S, han permitido comenzar a conformar una base de conocimiento propia que, a la par, está siendo utilizada para los diferentes proyectos de software que enfrenta la célula de desarrollo.

Conforme se fueron completando las prácticas, los participantes pudieron tener una referencia clara del avance de los proyectos. El progreso de las prácticas se fue midiendo de forma cuantitativa y esto permitió ver reflejado el esfuerzo dedicado de cada uno de los participantes en el logro de las metas del proyecto. Las prácticas proveyeron de un panorama confiable a los participantes conforme se avanzaba.

Una limitante de esta experiencia tuvo que ver con la recolección y comparación de métricas. Al ser un proyecto realizado en una entidad de reciente creación y en proceso de maduración, muchos datos históricos no existían lo que imposibilitó realizar comparaciones cuantitativas, por lo que se recurrió a utilizar aspectos cualitativos. Ejemplo de esto, fue el incremento en el compromiso de los alumnos participantes o la facilidad de comprensión del “qué” y el “por qué” tenían que hacer alguna labor.

5.1 Incubación y grupos de desarrollo

Con la experiencia acumulada en la generación de la primera célula de desarrollo, se corroboró la utilidad de KUALI-BEH como marco de trabajo. De

manera colaborativa fue posible construir una “guía” para grupos que se están iniciando en área de desarrollo de software, sin importar la poca o nula experiencia de los mismos en un entorno de trabajo real. El programa de incubación de células de desarrollo fungió como intermediario para estudiantes que están próximos a pasar del ámbito académico al laboral. Conjugando el conocimiento adquirido en las aulas con la experiencia de los profesores se potenció el trabajo en equipo y sobre todo la adquisición de experiencia a través de la inmersión en proyectos con clientes reales.

Los integrantes de las células al ejecutar las prácticas definidas por ellos mismos, comenzaron a adquirir una estructura de trabajo más ordenada, misma que paulatinamente se convirtió en un hábito y permitió generar buenas prácticas propias del equipo.

La célula de desarrollo Tic-Tac-S avanzó en el desarrollo de proyectos con sólo cuatro integrantes, demostrando que con una manera de trabajo ordenada y estructurada los esfuerzos de sus integrantes eran mejor aprovechados. Además, la comunicación y toma de acuerdos se agilizaron. Estos factores nos permiten sugerir que el tamaño ideal de las células puede variar entre 2 y 5 integrantes.

A lo largo de este proyecto, los profesores involucrados en la dirección de la célula, han observado que:

- La comunicación dentro de la célula se da principalmente por correo electrónico y video llamadas.
- Las habilidades de administración y comunicación mejoran cuando se tiene un método bien definido y conocido por todo el equipo de trabajo.
- Las tareas de documentación y conceptualización eran las que mayor esfuerzo consumían, por tal razón se aconseja generar guías específicas sobre estos temas.

5.2 Beneficios y mejoras derivadas del proyecto de incubación

Derivado de los buenos resultados de Tic-Tac-S en las asignaturas relacionadas con la Ingeniería del Software se ha incluido la enseñanza del marco de trabajo KUALI-BEH como parte de un estándar para el desarrollo de proyectos de software.

Actualmente, la carrera de Ingeniería en Sistemas Computacionales en conjunto con el Departamento de Tecnologías de la Información desarrollan proyectos de software alineados con KUALI-BEH en los cuales participan estudiantes, profesores y responsables del departamento. Esta conjunción repercutirá en experiencia y aprendizaje para mejorar varios aspectos relacionados con la enseñanza y con la mejora del programa de incubación.

Al día de hoy, en el ITSSLP-C, en un esfuerzo por consolidar la propuesta de las células de desarrollo de software se adecuó el área del centro de cómputo de la institución para proporcionar 3 espacios de trabajo colaborativo, ver Figura 5. Dos de estos espacios, conocidos como salas de trabajo 1 y 2 tienen capacidad para 10 personas, con lo cual pueden trabajar comodamente dos células de desarrollo en cada una. Finalmente, un tercer espacio con capacidad para 15 personas puede albergar tres células simultáneamente. Para el año 2015 se planea generar dos células de desarrollo nuevas y para el 2016 se tiene como meta trabajar proyectos con al menos 4 células de desarrollo independientes.



Figura 5.

Nuevo espacio colaborativo para la incubación de células en el ITSSLP-C.

Finalmente, los integrantes de la primer célula de desarrollo lograron clarificar y concretar la forma en que son capaces de desarrollar software, convirtiéndolo de algo abstracto a algo práctico. Además, la experiencia adquirida dentro de la célula al realizar proyectos reales mejoró las competencias técnicas y conductuales de sus integrantes. Lo anterior permitió fortalecer habilidades relacionadas con el autoaprendizaje, adquiriendo así herramientas útiles para el ámbito laboral.

5.3 Compromisos y metas planteadas por el ITSSLP-C

La academia de Ingeniería en Sistemas Computacionales en común acuerdo, incorporará KUALI-BEH como una herramienta de apoyo en las diferentes asignaturas con la finalidad de fortalecer las buenas prácticas.

Una vez conformada esta primera célula de desarrollo, el siguiente paso será apoyar al Departamento de Tecnologías de la Información con el desarrollo de proyectos internos incorporando a las nuevas células de desarrollo y, mediante KUALI-BEH, comenzar a generar diversos métodos de desarrollo de software que permitan la construcción de una base de conocimiento para la institución.

Para fortalecer el emprendedurismo en los alumnos, el área de sistemas computacionales contribuye con la generación de células de desarrollo de software desde el ámbito académico y propicia la conformación de start-ups de corte tecnológico. Siendo esta última una alternativa para los alumnos que egresan.

En la actualidad se requiere que las empresas sean competitivas y un factor de éxito es la calidad, por lo que resulta fundamental contar con una manera de trabajo bien establecida y que haya demostrado ser productiva. Es por ello que se está trabajando para que las empresas de desarrollo de software generadas en el ITSSLP-C desde su concepción incorporen buenas prácticas y las conjuguen con la creatividad y frescura de un recién egresado.

A mediano plazo, se pretende contar con al menos una célula consolidada para apoyar a otras células en formación y llevarlas a un estado de madurez. Esta idea se basa en un principio de sustentabilidad que permita repetir el ciclo para mantener activo el proyecto dentro del ITSSLP-C.

6. Conclusiones

El grupo de desarrollo pudo experimentar la diferencia de trabajo entre un ambiente académico y uno real. La implementación de KUALI-BEH permitió definir las actividades de los practicantes tomando en cuenta sus habilidades, intereses y metas particulares.

Una vez que se definió el primer método y se aplicó a un segundo proyecto, el esfuerzo necesario para la comprensión de la forma de trabajo por parte de los integrantes se redujo, al retomar lo registrado en la base de conocimiento generada, logrando el objetivo de utilizar un método propio, lo cual resultó ser un aliciente para los integrantes del grupo.

Después de aplicar KUALI-BEH, podemos concluir que provee de flexibilidad al método creado, al permitir incorporar prácticas que no se habían tomado en cuenta, adaptarlas y mejorarlas. La definición de un método propio del grupo fue una labor que requirió del compromiso de todos los integrantes de Tic-Tac-S, profesores y alumnos.

Como trabajo futuro, en el lado académico, profesores de la carrera de ISC tomaron la decisión de incorporar a KUALI-BEH en la asignatura de Ingeniería de Software como un nuevo tópico, dados los resultados hasta ahora obtenidos.

Para la conformación de las siguientes células de desarrollo, el ITSSLP-C estableció KUALI-BEH como herramienta base para la conformación de la manera de trabajo.

En cuanto a Tic-Tac-S, la célula continuará con las mejoras al método definido, ejecutándolo en nuevos proyectos y capacitando a los nuevos integrantes de la célula. De manera global, el ITSSLP-C ha confiado en el proyecto de incubación en vista de los buenos resultados y decidió invertir en

infraestructura para que este proyecto crezca y pueda formar más células de desarrollo en beneficio de la comunidad estudiantil.

Agradecimientos

Este trabajo fue posible gracias al apoyo del ITSSLP-C y a los departamentos involucrados en la gestación del proyecto del Centro de Desarrollo de Software del Instituto Tecnológico Superior (CEDESITS), así como al Posgrado en Ciencia e Ingeniería de la Computación (PCIC-UNAM) y al Programa de Apoyo a los Estudios de Posgrado (PAEP-UNAM).

Referencias

Humphrey, W., 2010. Why teams need Operational Processes. Software Engineering Institute, Carnegie Mellon University.

Instituto Tecnológico Superior de San Luis Potosí, 2014. Informe de la 3a Reunión Ordinaria de la H. Junta Directiva.

International Organization for Standardization, 2007. ISO/IEC 24744:2007 Software engineering – Metamodel for development methodologies.

International Organization for Standardization, 2008. ISO/IEC 12207:2008 Systems and software engineering – Software life cycle processes.

International Organization for Standardization, 2011. ISO/IEC 29110:2011 Software engineering – Lifecycle profiles for very small entities (VSEs) – Management and engineering guide: Generic profile group: Basic profile.

Object Management Group, 2008. Software and systems process engineering metamodel (SPEM). Especificación formal.

Object Management Group, 2011. Unified modeling language (UML) infrastructure. Especificación formal.

Object Management Group, 2014a. KUALI-BEH: Software Project Common Concepts. Normative Annex en ESSENCE – Kernel and Language for Software Engineering Methods. Especificación formal.

Object Management Group, 2014b. ESSENCE – Kernel and Language for Software Engineering Methods. Especificación formal.

Software Engineering Institute, 2010. CMMI for development. Version 1.3. Carnegie Mellon University.

Notas biográficas:



Emmanuel Arroyo-López Ingeniero en Sistemas Computacionales por el Instituto Tecnológico Superior de San Luis Potosí, Capital desde 2014. Actualmente se desempeña como Jefe del área de desarrollo de software de la empresa Masfletes. Sus áreas de interés son el desarrollo de software y la administración de bases de datos.



Teresa Ríos-Silva Ingeniera en Sistemas Computacionales por el Instituto Tecnológico Superior de San Luis Potosí, Capital desde 2014. Actualmente se desempeña como Administradora Estatal del Banco Nacional de Datos e Información sobre casos de Violencia contra las Mujeres en los Servicios de Salud del Estado de San Luis Potosí. Sus áreas de interés son el análisis y especificación de requerimientos y el diseño de interfaces de usuario.



Alejandro Rico-Martínez Ingeniero en Sistemas Computacionales por el Instituto Tecnológico de San Luis Potosí desde 1994, realizó estudios de posgrado en la Maestría en Inteligencia Artificial de la Universidad Veracruzana-Laboratorio Nacional de Informática Avanzada. Actualmente se desempeña como Jefe de la División de Ingeniería en Sistemas Computacionales en el ITSSLP-C.



Miguel Ehécatl Morales-Trujillo Doctor en Ciencias (Computación) por la Universidad Nacional Autónoma de México desde 2015. Actualmente se desempeña como Profesor de Asignatura en la Facultad de Ciencias de la UNAM. Sus áreas de interés son la Ingeniería de Software y Bases de Datos.



Hanna Oktaba Doctora por la Universidad de Varsovia, Polonia. Desde 1983 es profesora de la UNAM, tanto en la Facultad de Ciencias como en el Posgrado en Ciencia e Ingeniería de la Computación. Entre 2002 y 2004 estuvo a cargo de los proyectos del modelo de procesos para las pequeñas organizaciones de software MoProSoft, modelo de evaluación EvalProSoft y Pruebas controladas con empresas, apoyados por el programa PROSOFT de la Secretaría de Economía, cuyo resultado fue la creación de la norma mexicana MNX-I-059-NYCE en 2005. Desde 2006 es representante de México ante el Work Group 24 de ISO JTC1/SC7 Software and System Engineering.



Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 2.5 México.

Un modelo para la solución de requerimientos no alineados: El caso del Software lúdico para la divulgación

Héctor G. Pérez-González
Universidad Autónoma de San Luís Potosí
hectorgerardo@acm.org

Rosa M. Martínez-García
Universidad Autónoma de San Luís Potosí
rosma.fi.uaslp@gmail.com

Francisco E. Martínez-Pérez
Universidad Autónoma de San Luís Potosí
fcoemtz@gmail.com

Sandra Nava-Muñoz
Universidad Autónoma de San Luís Potosí
senavam@uaslp.mx

Alberto S. Nuñez-Varela
Universidad Autónoma de San Luís Potosí
alberto_snv@hotmail.com

Resumen: Este artículo propone lineamientos metodológicos para ser aplicados en el proceso de requerimientos cuando estos se presentan como “No alineados”. Se considera que esta situación se da cuando los objetivos del propietario del software son distintos a los del usuario del mismo. Como caso de estudio se utiliza el desarrollo de software lúdico (también conocido como juegos digitales, videojuegos, o juegos serios) con fines de divulgación. Esto involucra generalmente un conjunto de stakeholders más heterogéneo que el del software convencional. Adicionalmente, si el software se utiliza con objetivos de divulgación de ciencia, tecnología o innovación se agrega la complicación de la presencia de objetivos no alineados ya que el usuario final puede ignorar el objetivo real del software que utilizará. Consecuentemente el análisis de requerimientos cobra una importancia crucial. La literatura presenta propuestas para adaptar el proceso completo de software con objetivos educativos, sin poner atención al proceso de requerimientos ni a los objetivos de divulgación. Esta propuesta puede ser extrapolada a cualquier proceso de Ingeniería de requerimientos de Software donde se involucren requerimientos no alineados.

Palabras Clave: Ingeniería de Requerimientos, Divulgación de Ciencia, Tecnología e Innovación.

A model for the solution of non-aligned requirements: The case of ludic Software for science communication

Abstract: This paper proposes methodological guidelines to be applied in the requirements engineering process when these requirements are presented as "not aligned". It is considered that this situation occurs when the software owner's objectives are different from those of its user. The software whose goal is the dissemination of science, technology and innovation, can have not aligned goals. In these cases, the end user can ignore the real purpose of the software to be used. Consequently the requirements analysis becomes crucial. The literature presents proposals to adapt the entire process of software for

educational purposes, without paying attention to the process of requirements or science communication. This proposal can be extrapolated to any Software requirements engineering process where not aligned requirements are involved.

Keywords: Collaborative Virtual Environments, Fictitious scenarios, Teamwork.

1. Introducción

La ingeniería de Requerimientos es una rama de la Ingeniería de Software que ha cobrado gran importancia debido a que representa la etapa del proceso de software en la que se minimizan los costos debido a la realización de correcciones a los productos generados por etapa. Es decir las modificaciones a los requerimientos son mucho menos costosas que las modificaciones a los programas. Un buen proceso de Ingeniería de requerimientos reduce los riesgos de inconsistencias entre las necesidades declaradas por los usuarios y las entendidas por los desarrolladores.

Bajo esta premisa podemos establecer un esquema de tres roles generales: El propietario del software, el desarrollador y el usuario. La probabilidad de éxito del software es directamente proporcional al grado en que estos tres actores coincidan en el entendimiento de los objetivos. El objetivo general de un producto de software es la premisa de la cual se derivan los objetivos particulares, es por ello que estos deben ser congruentes y consistentes.

El objetivo general de un producto de software puede ser de productividad, de entretenimiento, de aprendizaje, etc. Los usuarios generalmente están conscientes de estos objetivos, al igual que los diseñadores, programadores, propietarios, etc. A esto lo denominamos Alineación de Requerimientos.

La alineación de requerimientos debería en principio ser un prerequisite para iniciar el proceso de Ingeniería de requerimientos, sin embargo hay caso en

donde esta situación no se da por la misma naturaleza de la aplicación y su dominio del problema.

Este artículo reporta la investigación sobre diversos tipos de software susceptibles de presentar objetivos generales no alineados que deriven en requerimientos no alineados, y se analizan diversos tipos de software en función del dominio del problema.

Derivado de la investigación, se propone el análisis del software lúdico para la divulgación como caso de estudio, en el que se observa que de manera sistemática los objetivos del propietario del software y los del usuario son claramente diferentes.

La investigación sobre el estado del arte muestra que existen requerimientos muchas veces no explícitos pero que pueden ser cruciales para la consecución exitosa del proceso. Estos requerimientos pueden ser por ejemplo de naturaleza legal o ser tan subjetivos como los generados por las motivaciones de los propios desarrolladores. En secciones posteriores se explora la posibilidad de que estos puedan ser considerados como no alineados. Para su estudio, se analiza el caso del software para divulgación de ciencia, tecnología e innovación en sus diferentes facetas. Para ello se analiza el concepto de divulgación en la siguiente sección.

1.1 Divulgación

La divulgación de Ciencia, Tecnología e Innovación es un concepto poco entendido por lo que debe ser definido con precisión. En un sentido amplio, la divulgación es un sistema de conocimiento, cuyo principio rector es la reformulación clara, amena y delimitada del conocimiento científico y tecnológico (Albourkrek, 2005), y una forma especial de transmitir dicho conocimiento (López Beltrán, 1983). De acuerdo con (Albourkrek, 2005): “La divulgación de la ciencia, la tecnología y la innovación juegan un papel crucial

en el desarrollo de la sociedad dado que erradica mitos, aumenta la calidad de vida de las personas, coadyuva a generar mejores decisiones vocacionales y propicia una mas informada participación de las personas en las decisiones colectivas”.

Los conceptos, metodologías, teorías, leyes y prácticas científicas no pueden ser comunicados con medios convencionales de manera atractiva al público en general, esto es debido a la complejidad de los mismos (Aitkin, 2005).

Esta cultura científica puede ser transmitida de manera exitosa con el uso de sistemas de cómputo con características muy particulares. Existen dos tipos de software que pueden ser útiles en la transmisión del conocimiento: De simulación y Lúdicos (Susi et al., 2007). Los sistemas de cómputo de simulación se han venido utilizando con éxito por los científicos para facilitar su propia construcción del conocimiento. Más recientemente el software lúdico con un componente de enseñanza ha sido denominado software de juegos serios. Este ha jugado un papel importante como herramienta para facilitar el aprendizaje (Gunter et al., 2006). Lo anterior es corroborado por el trabajo en (Dempsey & Johnson, 1998) quienes establecen que un juego logrará objetivos de aprendizaje si cuenta con las siguientes características: Atención, relevancia, confianza, reto, satisfacción y éxito.

Ambos tipos de software (simulación y juegos serios) han tenido éxito como coadyuvantes del aprendizaje y del autoaprendizaje.

Un sistema de cómputo que contribuya con efectividad a los objetivos de la divulgación consta de características muy particulares: Metas, propietarios, tipos de personas involucradas (stakeholders), tipos de usuarios, tipos de interacciones, proceso de desarrollo, evaluación de objetivos, etc.

Realizar acotaciones en relación al tipo de software que mejor cumpla con las expectativas de divulgación puede derivar en la maximización de sus posibilidades de éxito.

Este artículo propone una implementación sistemática de principios de Ingeniería software, (en particular de Ingeniería de requerimientos) para minimizar las dificultades que puedan presentarse por la presencia de requerimientos no alineados. Lo anterior deberá derivar en la construcción efectiva de aplicaciones de cómputo que presenten este problema.

La sección 2 muestra principios básicos de Ingeniería de requerimientos, describe la literatura relacionada con la alineación de los mismos y con la adaptación de lineamientos de dicha Ingeniería en función del tipo de aplicación a desarrollar.

La sección 3 explica el significado de la divulgación de Ciencia, Tecnología e Innovación, sus objetivos y sus motivaciones.

La sección 4 describe los lineamientos propuestos para solucionar el problema de requerimientos no alineados tomando como caso e estudio el proceso de desarrollo de software lúdico con fines de divulgación.

Finalmente las secciones 5 y 6 muestran la experiencia en el uso de esta metodología, los resultados observados, las conclusiones y el trabajo futuro propuesto.

2. Trabajo relacionado

Aunque la importancia de la Ingeniería de Requerimientos es bien conocida, la literatura no muestra evidencia sobre estudios de casos en los que los requerimientos no presentan consistencia entre ellos. Esta situación se presenta cuando los stakeholders declaran objetivos con alto grado de disparidad.

Esta sección muestra los resultados encontrados en la literatura en relación al estado del arte en Ingeniería de requerimientos, las dificultades que pueden presentarse cuando los requerimientos son inconsistentes entre ellos y las

diversas adaptaciones a metodologías de requerimientos en función del dominio del problema.

2.1 Ingeniería de Requerimientos

De acuerdo con la IEEE (IEEE Standard 610.12-1990) un requerimiento o requisito se define como una condición o capacidad que debe cumplir un sistema para satisfacer un contrato, un estándar, una especificación o cualquier documento formalmente prescrito. Los requerimientos deben ser: Correctos, consistentes, verificables y rastreables. La Ingeniería de Requerimientos es el proceso de licitar, entender, especificar y validar los requerimientos de los diferentes actores involucrados (stakeholders). En (Wieggers, 2000) se identifican tres niveles de requerimientos de software: Requerimientos del negocio que se reflejan en el documento de visión, Requerimientos de usuario, que se reflejan mediante un modelado (casos de uso) y Requerimientos funcionales y no funcionales documentados en las Especificaciones de Requerimientos de Software (SRS por sus siglas en inglés). Lo anterior se ilustra en la Figura 1.

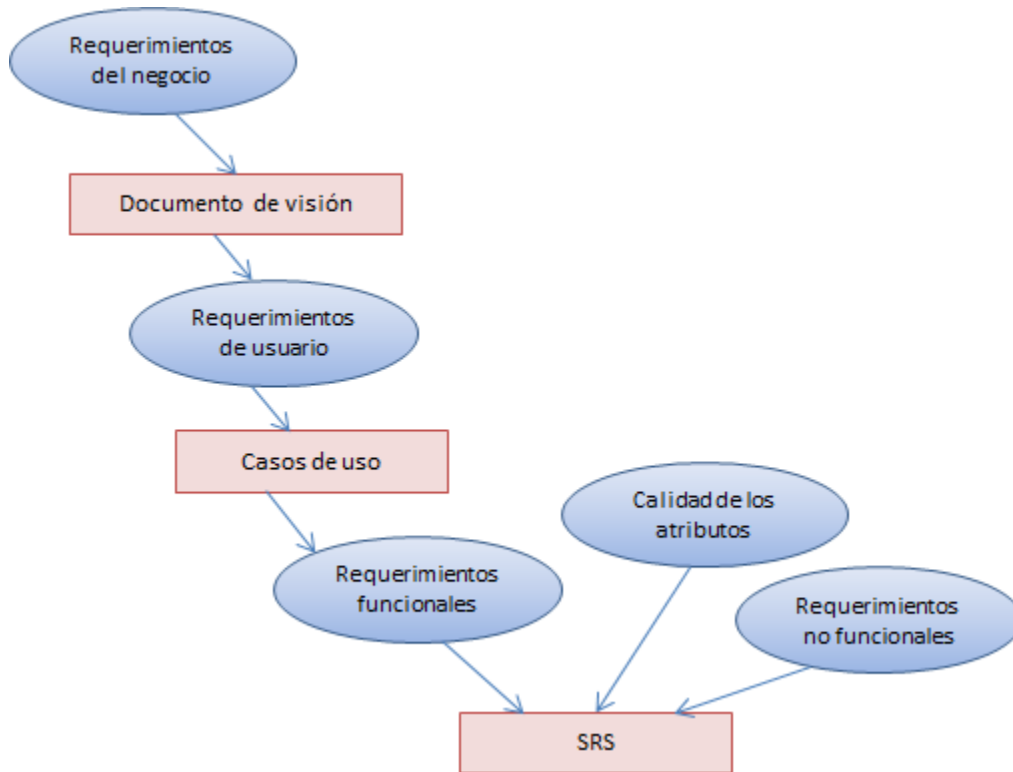


Figura 1.

Proceso de Ingeniería de Requerimientos (Wiegiers, 2000)

En (Sawyer & Kotonya, 2001) se describe el proceso de Ingeniería de Requerimientos con un modelo en espiral para cualquier tipo de software lo cual se ilustra en la figura 2. El proceso incluye las actividades de planificación/extracción, análisis, especificación y validación de requerimientos.

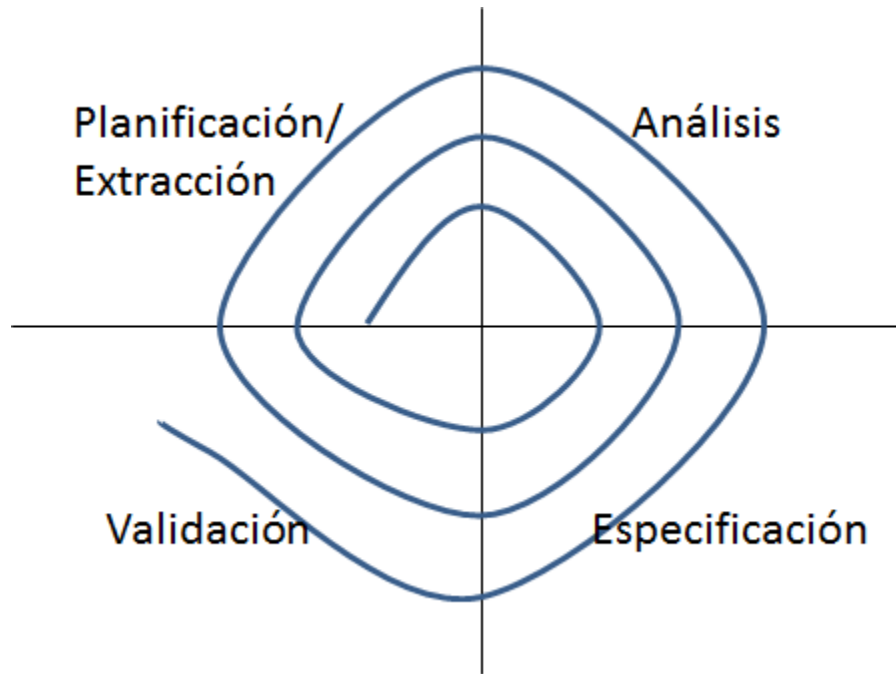


Figura 2.

Proceso de Ingeniería de Requerimientos (Sawyer & Kotonya, 2001)

De acuerdo con (Pressman, 2005) el proceso de requerimientos se compone de los siguientes pasos no necesariamente secuenciales: Concepción, obtención, elaboración, negociación, especificación, validación y gestión de requerimientos. En (Sommerville, 2010) se propone un modelo de proceso en espiral que incluye las actividades de obtención, especificación y validación de requerimientos (Figura 3).

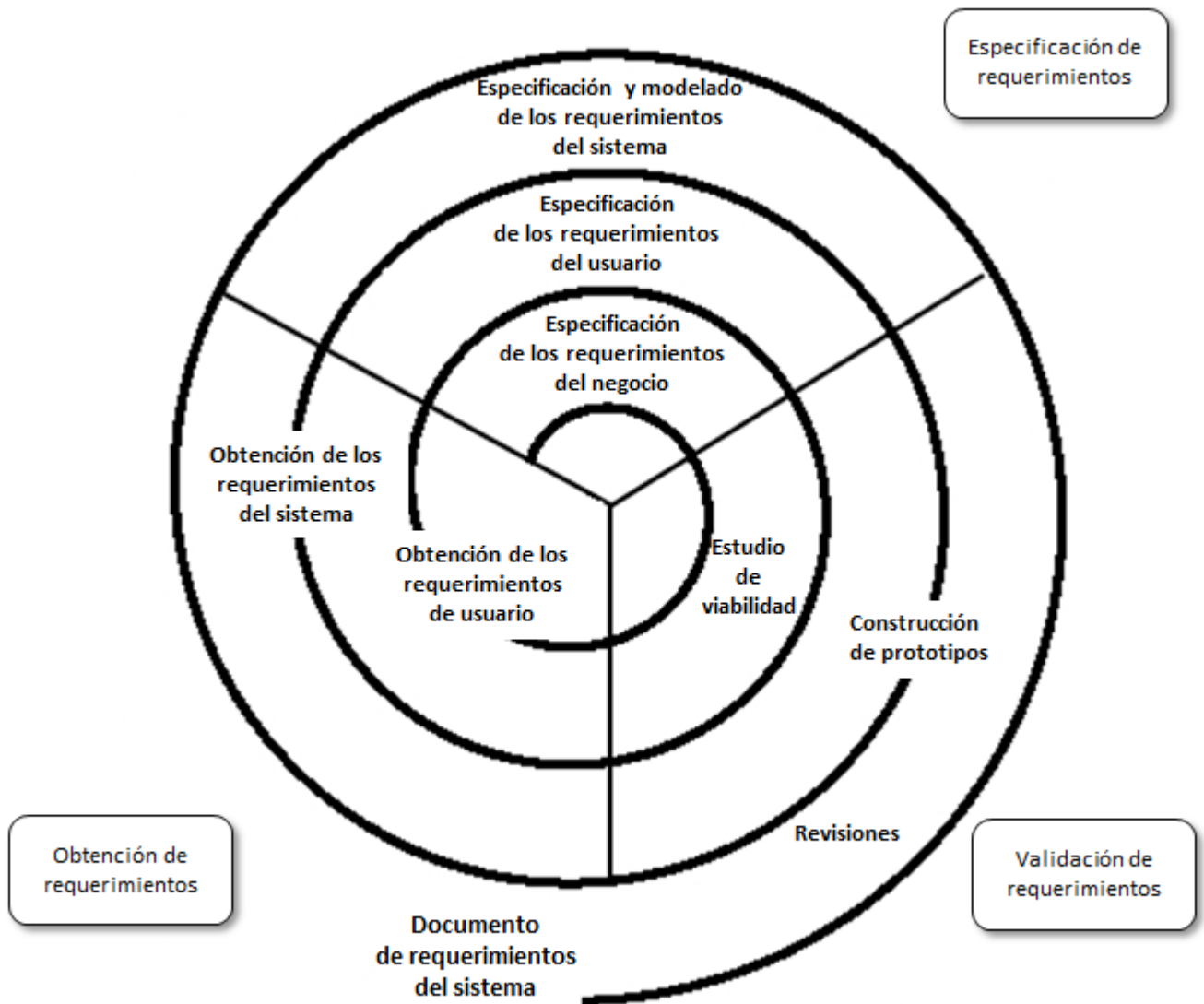


Figura 3.

Proceso de Ingeniería de Requerimientos (Sommerville, 2010)

Como se observa, los autores coinciden en las siguientes propuestas:

1. El proceso de Ingeniería de Requerimientos debe incluir al menos las etapas de Obtención, Especificación y Validación.
2. El conjunto propuesto de participantes del proceso no presenta alta especificidad ya que del lado del cliente se propone solo los stakeholders:

Propietario e Usuario y del lado del desarrollador solo se incluye al Ingeniero de requerimientos.

3. Los entregables propuestos son al menos los documentos de Visión, Casos de Uso y Especificación de Requerimientos

Para saber si es factible adaptar el proceso de requerimientos para facilitar la construcción de software con fines específicos se presenta un resumen del estado del arte relativo al tema.

2.2 Alineación de Requerimientos

Se propone la noción de Alineación de Requerimientos cuando estos son claros y consistentes en el mayor nivel de abstracción posible. Este nivel es el asociado al objetivo general del producto de software y no a los objetivos específicos de los diferentes stakeholders.

Los stakeholders de software convencional tienen objetivos pragmáticos de productividad. Por ejemplo, un software para uso de un banco (ventanillas) tiene varios objetivos: Para el que lo opera (empleado del banco), el objetivo es realizar operaciones financieras (depósitos, pagos, traspasos, etc.), esto lo hace en representación del cliente del banco (usuario indirecto). El objetivo del propietario del software (El banco) es obtener ganancias económicas sin embargo para lograrlo se sabe que será utilizado para hacer operaciones financieras. Para este tipo de aplicaciones, por ejemplo, el diseñador de las interfaces de usuario podría tener el objetivo de mostrar información de manera, precisa y estética. Como se observa, aunque las metas individuales de los diferentes roles son claramente distintas y en ocasiones hasta contradictorias, la alineación de alto nivel se presenta ya que se coincide en el objetivo general de cubrir los requerimientos de los clientes.

En el caso de un software educativo, el propietario busca que los usuarios obtengan aprendizaje a partir del uso de sus productos. El usuario, por su parte esta consiente de esa meta y busca con su uso adquirir dicho aprendizaje.

Toda la gama de stakeholders desde el usuario final hasta el productor del software, (pasando por los analistas, diseñadores, programadores, pedagogos, ilustradores, etc.) tienen una idea clara y consistente sobre el objetivo general a cumplir.

Los anteriores ejemplos muestran lo que denominamos: alineación de requerimientos. Esta se da entonces cuando los objetivos generales de los grupos involucrados muestran coincidencia.

La literatura muestra limitada información acerca de problemas de requerimientos no alineados.

En (Breux et al., 2006) se propone el concepto de Ingeniería de conformidad (Compliance Engineering). Bajo este esquema, el autor presenta una metodología que provee un enfoque sistemático para la eliminación de ambigüedades en documentos sobre leyes escritos en lenguaje natural. El artículo describe un marco de trabajo para extraer derechos y obligaciones a partir de políticas y regulaciones y un modelo para alinear estos artefactos con requerimientos de software. Este es el primer trabajo que presenta la alineación de requerimientos de software como un problema por resolver. En este caso la no alineación se da entre el conjunto de los requerimientos de usuarios y desarrolladores y entre el conjunto de los requerimientos por cumplir de leyes y regulaciones.

En (Hans, 2013) se presenta un modelo de alineación entre los requerimientos del proyecto y los requerimientos de los miembros del equipo de desarrollo del mismo. Este modelo considera las necesidades de los empleados que desarrollarán el software y las clasifica de acuerdo con el modelo jerárquico de Maslow (Taormina & Gao, 2013). Este modelo toma en cuenta necesidades de Auto-actualización, de autoestima, de seguridad y fisiológicas.

El trabajo de Hans busca coadyuvar en la búsqueda de compatibilidad en las motivaciones de las personas, sin embargo no aborda el problema que se da

cuando las motivaciones de los usuarios del software son diferentes de las que los desarrolladores tienen en relación al objetivo perseguido al usar dicho software.

No se encuentra evidencia de trabajo relacionado con el problema explícito de requerimientos no alineados de software.

De acuerdo con la investigación realizada se propone el estudio de los productos de software lúdico (software de juegos) con objetivos de divulgación o comunicación pública de la ciencia como aquel que sistemáticamente puede presentar Requerimientos No Alineados.

Las siguientes secciones exploran el estado del arte en relación a las adaptaciones propuestas al proceso de Ingeniería de requerimientos en función del dominio de los problemas que abordan.

2.3 Adaptaciones al Proceso de requerimientos

La literatura muestra propuestas para adaptar el proceso de Ingeniería de Requerimientos de acuerdo al tipo de aplicaciones a desarrollar. De acuerdo con (Eljabiri & Deek) los requerimientos de proyecto pueden influenciar la decisión en relación al modelo de proceso de software a utilizar. En (Escalona & Koch, 2004) se propone la adaptación de dicho proceso al desarrollo de aplicaciones web, ellos proponen la inclusión de validación en la actividad de especificación. Su argumento en favor de proponer un enfoque diferente por el hecho de considerar aplicaciones web radica en la existencia de diferencias cruciales entre las que destacan: El gran número de stakeholders, los aspectos de navegación, la calidad del esquema (layout) de presentación y la sincronización multimedia. En (Farooq & Arshad, 2010) se propone un modelo de proceso de software específico para desarrollo de aplicaciones de web semántica, mientras que en (Shah, 2003) se centra en la etapa de diseño de este mismo tipo de aplicaciones. Estas propuestas coinciden en que la

existencia de diferencias claras en las características que rodean el desarrollo inicial del software es razón suficiente para adaptar los procesos de requerimientos para un tipo de aplicación en particular, en favor de obtener mejores resultados.

Se concluye que en caso de encontrar tipos de aplicaciones en donde sistemáticamente se presenten requerimientos contradictorios entre ellos, es válida la propuesta de un conjunto de lineamientos metodológicos que ayuden a mejorar este tipo de proyectos.

2.3.1 Adaptaciones al Proceso de Requerimientos para Aplicaciones de Educación

Existen propuestas para adaptar los procesos al desarrollo de software educativo. Ejemplos de estas, incluyen la metodología de software educativo bajo un enfoque de calidad sistémica propuesta en (Díaz-Antón et al., 2008) y MeISE: una metodología de Ingeniería de Software Educativo (Figuerola, 2009). Ambas se enfocan en la producción de software para aprendizaje formal e incluyen fundamentación en teorías de educación establecidas. En relación al desarrollo de software educativo en el ámbito de la tecnología más reciente, destaca la propuesta presentada en (Cook et al., 2008), quienes proponen adaptaciones en el proceso de construcción de software móvil. En (Pérez-González et al., 2013) se propone una metodología de desarrollo de secuencias de video para la divulgación de ciencia, tecnología e innovación que incluye la programación de software para la utilización de un robot humanoide como personaje principal (COPOCYT. Robot Ruidozo), sin embargo éstas aunque son de entretenimiento, no pueden catalogarse como juegos.

2.3.2 Adaptaciones al Proceso de Requerimientos para Aplicaciones Lúdicas

En relación al uso de software lúdico con objetivos específicos, en (Murphy-Hill et al., 2014) se describe en qué sentidos el desarrollo de videojuegos es diferente del desarrollo de software convencional. Ellos concluyen que:

Los desarrolladores de juegos con respecto a los desarrolladores de otro tipo de software:

1. Reciben requerimientos con menor precisión.
2. Tienden a usar lo que ellos perciben como proceso ágil.

Los equipos de personas para el desarrollo de juegos le confieren mucho más valor:

1. A la creatividad
2. A las habilidades de comunicación con los no ingenieros.
3. A la confluencia de integrantes que formen un grupo más diverso.

Finalmente identifican que las personas se impresionan más con el trabajo de los desarrolladores de juegos que con el software convencional. En (Cruz-Cunha, 2012) se hace una compilación acerca de los trabajos de investigación sobre juegos serios utilizados como herramientas educativas, de negocios y de investigación. De esta compilación destaca el trabajo de (Cowley, 2014), donde se propone un modelo de proceso de software (QUARTIC) para el desarrollo de este tipo de juegos. En (Gunter et al., 2006) van más allá y proponen un paradigma formal para el diseño de este tipo de software. Finalmente en (Wilson, 2005) se destaca la existencia de un “Vibrante y variado sector del diseño, producción y distribución independiente de juegos”.

Expertos de la Industria sugieren que este tipo de desarrollo conocido como Indie gaming será cada vez más importante y creciente en el futuro inmediato. La literatura no presenta propuestas específicas de procesos para su adaptación con objetivos de divulgación. Dentro de los esfuerzos similares

destaca la propuesta en (Flanagan & Nissenbaum, 2007). Ellos proponen una metodología de diseño que considera a nivel fundamental la incorporación de valores éticos y de activismo social en los juegos de software. En (Huang & Yang, 2012) se estudian los resultados del uso de un software en línea y multi-jugador de tipo juego de rol cuyo principal objetivo es el aprendizaje incidental de vocabulario.

Finalmente en (Aitkin, 2005) se concluye que “La literatura relativa a simulaciones y juegos demuestra colectivamente que su habilidad para recrear la realidad, modelar sistemas complejos y su capacidad de visualización e interacción fomenta la práctica de la ciencia”.

La tabla 1 muestra una caracterización de las propuestas encontradas en la literatura. Como se ilustra, la mayoría de ellas propone lineamientos generales en los procesos de software, cuatro de ellas se enfocan en la etapa de diseño y solo dos se concentran en la etapa de Ingeniería de Requerimientos. Por otro lado, de las adaptaciones propuestas aunque la mitad de ellas se refieren al ámbito de aplicaciones lúdicas, solo dos consideran objetivos de divulgación. No se encontró evidencia de la existencia de propuestas centradas en la etapa de requerimientos para el desarrollo de software lúdico con fines de divulgación.

Dado que los objetivos generales de divulgación denotan la existencia de características especiales (a detallarse en la sección 3). se concluye que una propuesta como la presentada puede ser de valor.

Publicación	Proceso I-SW	Ing de. Req.	Diseño	Adaptación
Escalona y Koch [13]		✓		Web
Farooq y Arshad [14]	✓	✓		Web semántica
Shah [15]			✓	Web semántica
Díaz-Antón et al [16]	✓			Aprend. Formal
Figuerola [17]	✓			Aprend. Formal
Cook et al. [18]			✓	SW Movil
Perez-Gonzalez et al. [19]	✓			Divulgación
Cruz-Cunha [22]	✓			Juegos serios
Cowley et al. [23]	✓			Juegos serios
Gunter et al. [5]	✓			Juegos serios
Wilson [24]	✓			Video Juegos
Flanagan y Nissenbaum [25]			✓	Juegos/Activ.Social
Huang y Yang [26]	✓			Juegos/Vocabulario
Aitkin [3]			✓	Divulgación

Tabla 1.

Propuestas de Procesos de software con adaptaciones

De lo anterior se concluye lo siguiente:

1. Existen trabajos que buscan adaptar los procesos de Ingeniería de software y en particular de Ingeniería de Requerimientos a los diferentes dominios de aplicación, en particular al dominio de la educación (Díaz-Antón et al., 2008) (Figuerola, 2009) (Cook et al., 2008).
2. Existen trabajos que analizan los procesos de producción de software de juegos con fines educativos (Cruz-Cunha, 2012) (Cowley, 2014) (Wilson, 2005) (Flanagan & Nissenbaum, 2007) (Huang & Yang, 2012).
3. La literatura no presenta propuestas específicas de procesos de desarrollo de software lúdico para su adaptación con objetivos de divulgación enfocándose en la etapa de requerimientos.

3. Divulgación de ciencia, tecnología e innovación

Para proponer adaptaciones al proceso de Ingeniería de Requerimientos es necesario conocer con más precisión los fundamentos y objetivos de la divulgación. Como se establece en (Sánchez Ramos & Barradas Bribiesca, 2014): “En la divulgación confluyen tres partes fundamentales: la ciencia, el divulgador y el público;” Es importante delimitar su función ya que si bien es una alternativa en la transmisión de conocimiento científico “traducido”, no pretende sustituir a la educación formal ya que no se rige bajo las mismas directrices educativas que se implementan dentro del aula, aunque esto no quiere decir que sea un proceso sin planeación, objetivos y evaluación. Ramos y Bribiesca establecen que “La divulgación no está limitada por el tiempo ni por el espacio, en ella el conocimiento fluye y es de libre elección para los usuarios”.

Para clarificar los conceptos en (Sánchez Ramos & Barradas Bribiesca, 2014) debemos también caracterizar el aprendizaje. El aprendizaje, desde un punto de vista general puede clasificarse como formal o informal. Formal es aquel que se da de manera sistemática dentro de un contexto institucional, escolarizado y bajo un esquema de evaluación. El aprendizaje informal es aquel que se da fuera del contexto referido y puede ser experimentado en cualquier otro escenario de tiempo y espacio (Cook et al., 2008). La divulgación de la ciencia, la tecnología y la innovación juega un papel crucial en el ámbito de la educación informal (Calvo Hernando, 2012).

Aunque como veremos más adelante, la divulgación no solo se circunscribe al ámbito de la enseñanza, es útil conocer los posibles diferentes objetivos que cualquier software puede tener en relación a la educación.

Se propone un esquema para caracterizar un producto de software en un espacio de dos dimensiones (figura 4). El eje horizontal representa el grado en

que los objetivos prediseñados y deliberados de un software lo colocan en una clasificación que va de totalmente educativo a totalmente de esparcimiento. El eje vertical representa el grado en que un software puede ser utilizado en procesos de aprendizaje de formal a informal.

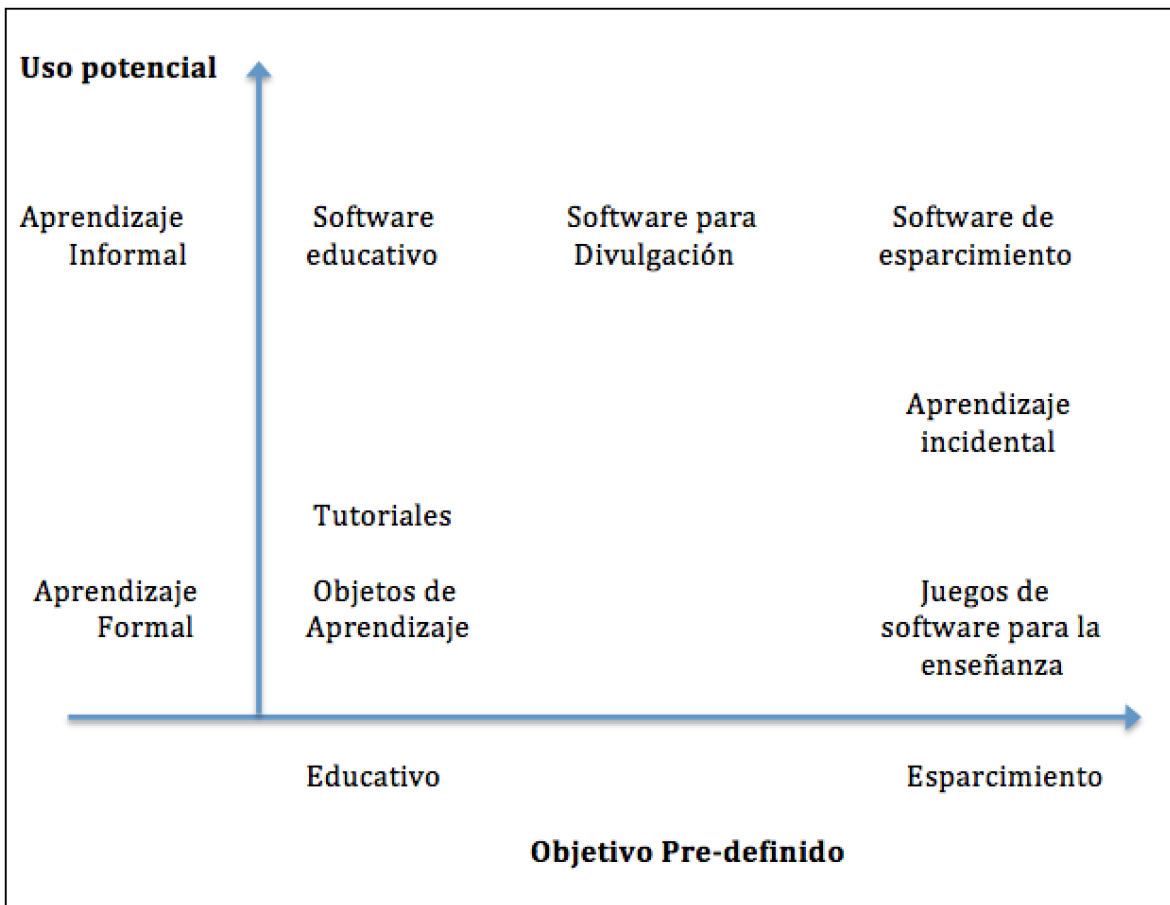


Figura 4.

Marco de referencia: Productos de software, Objetivos y usos potenciales.

El software con el objetivo pre-definido de esparcimiento se presenta en productos que pueden contribuir a los dos tipos de aprendizaje (formal e informal); no obstante, al aprendizaje incidental se presenta en todos ellos. Los objetivos del software de divulgación son educativos y de esparcimiento pero el aprendizaje al que contribuyen es de tipo informal.

3.1 Objetivos de la Divulgación

A pesar del poco conocimiento que incluso la comunidad científica tiene acerca de los objetivos de la divulgación, la mayoría de los expertos y practicantes de esta disciplina coinciden en un conjunto de ellos los cuales son enumerados a continuación. La primera propuesta formal documentada acerca de la divulgación es la presentada por (Albourkrek, 2005). En ésta se establece que el fomento a la curiosidad, la imaginación y el espíritu de investigación son los objetivos a ser fomentados en el público objeto de esta actividad. Además se destaca que esta actividad puede contribuir a la orientación vocacional, a la erradicación de mitos y en general al mejoramiento de la sociedad. La tabla 2 muestra este conjunto de objetivos.

1. Es capaz de crear una atmósfera de estímulo a la curiosidad por la ciencia y su método.
2. Ayuda a despertar la imaginación.
3. Cultiva el espíritu de investigación.
4. Desarrolla la capacidad de observación, la claridad de pensamiento y la creatividad.
5. Contribuye a descubrir vocaciones científicas.
6. Propicia una relación más humana con el científico.
7. Erradica mitos.
8. Abre caminos hacia la participación del desarrollo cultural universal.
9. Enriquece la condición humana, en un sentido más filosófico.

Tabla 2.

Objetivos de divulgación (Albourkrek, 2005)

Con lo anterior se deduce que la esencia de la divulgación no está encaminada solo hacia el aprendizaje sino que tiene objetivos mucho más complejos. Estudios posteriores proponen otros grupos de objetivos. En (Shults, 2008), más recientemente, se definen los objetivos de la tabla 3.

1. Obtener recursos (fondos) para investigación
2. Entender las formas y herramientas para comunicarse con la sociedad
3. Facilitar el entendimiento público de la ciencia
4. Fomentar vocaciones científicas.

Tabla 3.

Objetivos de la divulgación (Shultz, 2008)

La tabla 3 muestra que la divulgación también contribuye a la obtención de fondos para la investigación; esto se logra debido a la premisa de que una sociedad más informada y consiente del valor de la ciencia puede influir en las decisiones que conlleven a dirigir fondos para financiarla. En este contexto podemos preguntarnos: ¿Se pueden presentar requerimientos no alineados en procesos de construcción de software lúdico para la divulgación?

Se identifica que una característica no común pero si posible es la existencia de requerimientos con inconsistencias entre ellos.

4. Solución de requerimientos no alineados. Caso de estudio: Software lúdico para divulgación

Con base en lo establecido en las secciones anteriores, podemos deducir que existen características especiales que hacen al desarrollo de software lúdico para divulgación (al cual denominaremos SWLD) diferente del desarrollo de software convencional.

Se puede afirmar que los objetivos de los stakeholders de SWLD no están alineados. El objetivo del usuario final (jugador) es simplemente la diversión, mientras que el que pagó por el desarrollo del mismo (el propietario, que es típicamente una institución gubernamental, educativa o filantrópica) tiene alguno(s) de los objetivos de las tablas 2 y 3. Por tanto y como se muestra en (Aitkin, 2005): “Los Juegos digitales son capaces de comunicar ciencia de manera encubierta, debido a que el jugador no está tratando de aprender” sino solamente de ganar el juego.

Este proceso es ilustrado en el diagrama de actividad de la figura 5.

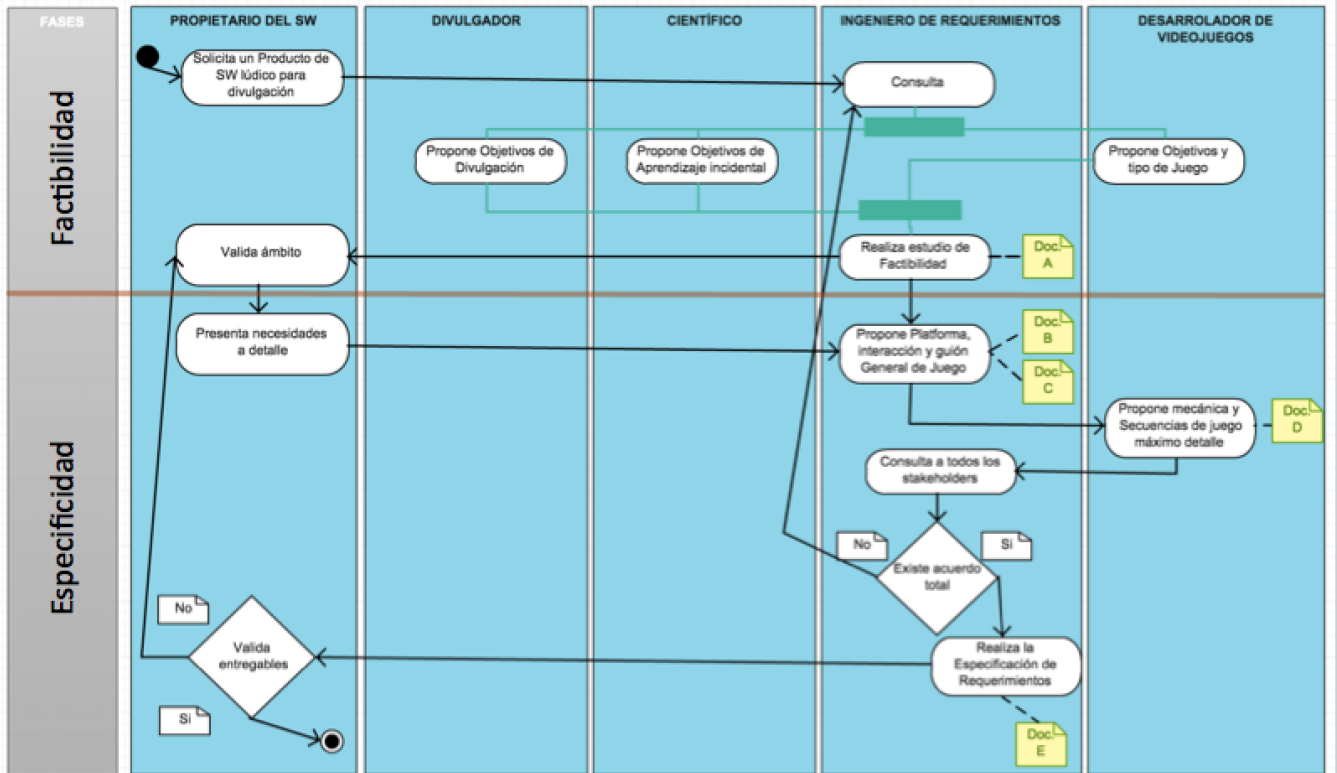


Figura 5.

Ingeniería de Requerimientos para el desarrollo de software Lúdico para la divulgación.

En este proceso participan cinco tipos de stakeholders: propietario, divulgador, científico, Ingeniero de requerimientos y desarrollador de video juegos y se divide en dos fases: Factibilidad y Especificidad. El proceso es iniciado por el propietario del sistema, quien plantea al ingeniero de requerimientos sus necesidades iniciando así la etapa de factibilidad. Este tipo de propietarios, establecen generalmente solo una muy vaga definición del o los objetivos de divulgación perseguidos. A continuación se realiza una consulta con el divulgador, el científico y el desarrollador de video juegos (de ser posible de manera síncrona en tiempo y espacio), como resultado se genera un estudio de factibilidad (Doc. A). Este entregable debe incluir las siguientes secciones: Un documento de ámbito, uno de objetivos y uno de concepto. El documento

de ámbito debe responder las preguntas: ¿Quién está solicitando el software?, ¿quién lo usará?, ¿Qué beneficios traerá? y ¿En qué entorno se utilizará?.

El documento de objetivos es de crucial importancia para minimizar los riesgos antes referidos, éste debe incluir objetivos de divulgación y objetivos de aprendizaje incidental (ambos se constituyen como los objetivos de proyecto). Finalmente, el documento de Concepto establece el tipo de juego (Primera o tercera persona, estrategia, aventura, rompecabezas, etc.), los objetivos generales del juego y su tema general a divulgar así como su público objetivo. Estos documentos deben ser validados por el propietario del software para poder pasar a la segunda etapa. La etapa final se denomina especificidad. Esta se inicia cuando el propietario tras validar el estudio de factibilidad (Doc. A) y a la luz de éste, comunica sus necesidades a mayor detalle. Con esto el proceso garantiza la alineación de objetivos generales. El ingeniero de requerimientos prepara entonces el documento de características del sistema (Doc. B) y el Guión general del Juego (Doc. C). El Doc. B. debe precisar las plataformas en que se ejecutará el sistema (PCs, Mac, Tablet, Web, Xbox, etc.), los sistemas operativos (Linux, Windows, MacOS, IOS, Android, etc.) y muy especialmente las formas de interacción (Consola, ratón, joystick, lentes de realidad virtual, guantes, sensores, etc.). Lo anterior deberá justificarse plenamente asociándolo a los objetivos de divulgación. El Guión general del Juego (Doc. C.) debe incluir la forma de ganar, cantidad máxima y mínima de jugadores, personajes, forma de progresar y de retroceder, mecánica general de juego y duración mínima y máxima de juego. El doc. C es revisado por el desarrollador de videojuegos quien produce el Documento de Juego (Doc. D) que incluye la información de su predecesor y una muy precisa y detallada descripción del juego, con todos los posibles cursos de acción. Especialmente importante es la referencia que se debe hacer en cada especificación del juego que corresponda al o a los objetivos de divulgación que serán logrados con dicha especificación. Una vez que los cuatro documentos son producidos, estos son validados por todos los stakeholders. Con ello, el Ingeniero de requerimientos puede generar el Documento final de Especificación de Requerimientos de

Software (SRS por sus siglas en inglés) (Doc. E). El SRS es un documento común en la literatura (Sommerville, 2010). Éste incluye la funcionalidad (lo que el software debe hacer), las interfaces externas (personas, hardware y software), el desempeño (Velocidad, tiempo de respuesta, etc.), atributos (portabilidad, seguridad, mantenibilidad, etc.) y restricciones de desarrollo. Cada uno de los requerimientos tiene un indicador único y su especificidad debe ser la máxima posible, Adicionalmente se propone el uso de una matriz de rastreabilidad en donde cada requerimiento sea asociado al o a los objetivos de divulgación pre establecidos.

5. Experiencia de uso de metodología propuesta y resultados

Los autores de este artículo han participado en el desarrollo de software en diversos dominios del problema, con ellos se ha identificado el tipo de aplicación que presenta invariablemente Lineamientos No Alineados, este tipo de producto es el software lúdico para divulgación.

En la primera mitad de los proyectos de divulgación desarrollados por los autores, no se siguió ninguna metodología, y por la experiencia obtenida, se llegó a la necesidad de generar una metodología basada en procesos de requerimientos, la cual ha sido aplicada a la otra mitad de los proyectos realizados por los autores.

Debido a que los propietarios de los productos generados utilizando y sin utilizar esta metodología eran los mismos, fue posible consultarlos para conocer sus opiniones sobre la utilidad de la misma. Una de los autores participó como experta en divulgación en la totalidad de los proyectos.

A cada producto de software le correspondió un tema general a divulgar (mineralogía, agua, metalurgia, tecnologías de la información, etc.).

Todos los proyectos contaron con el mismo ingeniero de requerimientos y el mismo líder de desarrollo.

Como resultado, los stakeholders manifestaron las siguientes ventajas por el hecho de usar los lineamientos propuestos:

- Mayor visibilidad del proceso de requerimientos
- Ahorro de recursos de tiempo de entre 5 a 21 %
- Reducción de la necesidad de reuniones síncronas
- Menor cantidad de iteraciones.
- Mayor rastreabilidad de los objetivos de divulgación en relación a los requerimientos específicos.

Así mismo, se enumeran las desventajas que se manifestaron:

- Inversión de demasiado tiempo en documentación
- Cuellos de botella cuando un stakeholder se demora en su trabajo.

6. Conclusiones y Trabajo futuro

En este trabajo hemos presentado el estado del arte en Ingeniería de requerimientos adaptados a propósitos específicos. Así mismo se ha caracterizado la divulgación de ciencia, tecnología e innovación. Se ha observado que las características especiales del software lúdico para la divulgación, en particular la no alineación de objetivos, sugiere la necesidad de establecer un proceso particular que garantice la minimización de los riesgos de obtener malos entendidos, inconsistencias y requerimientos contradictorios. A partir de los resultados expuestos se concluye que los lineamientos aquí propuestos se constituyen como una técnica útil para la consecución de mejores resultados en la construcción de software específico para la divulgación mediante juegos.

Con esta metodología se asegura que la presencia de requerimientos no alineados no impida la consecución de un esquema para el logro de objetivos de divulgación que esté incluido a nivel fundamental. Este paradigma deberá facilitar el diagnóstico y verificación de dichos objetivos y proveer un espacio conceptual común para desarrolladores, usuarios y propietarios de productos de software.

Se propone como trabajo futuro la utilización de estos lineamientos en desarrollo de software a mayor escala (más complejo y mayor cantidad y distribución de stakeholders) y la creación de herramientas automatizadas que permitan el uso del proceso propuesto de manera más ágil y generalizada.

Referencias

Aitkin, A. L., 2005. Playing at Reality: Exploring the potential of the digital game as a medium for science communication. Faculty of Science, The Australian National University.

Albourkrek, A., 1991. La divulgación de la ciencia. Citado en Calvo Hernando, M., 1997, Objetivos de la divulgación de la ciencia, Chasqui, 60.

Breaux, T. D., Vail, M. W., & Antón, A., 2006. Towards regulatory compliance: Extracting rights and obligations to align requirements with regulations. In Requirements Engineering, 14th IEEE International Conference, pp. 49-58. IEEE.

Calvo Hernando, M., 2012. Objetivos y funciones de la divulgación científica. ACTA, pp. 99-106.

Cook, J., Pachler, N. & Bradley, C., 2008. Bridging the gap? Mobile phones at the interface between informal and formal learning. Journal of the Research Center for Educational Technology, Vol. 4, pp. 3-18.

COPOCYT. Robot Ruidozo. Available:
https://www.youtube.com/results?search_query=%22robot+ruidoso%22

Cowley, B., 2014. The QUARTIC Process Model for Developing Serious Games: 'Green My Place' Case Study. Digital Da Vinci, N. Lee, Ed., ed: Springer New York, pp. 143-172.

Cruz-Cunha, M. M., 2012. Handbook of Research on Serious Games as Educational, Business and Research Tools (2 Volumes). Hershey, PA, USA: IGI Global.

Dempsey, J. V. & Johnson, R. B., 1998. The development of an ARCS Gaming Scale. *Journal of Instructional Psychology*, Vol. 24.

Díaz-Antón, M. G., Pérez M. A., Grimmán, A. C. & Mendoza, L. E., 2008. Propuesta de una Metodología de Desarrollo de Software Educativo bajo un Enfoque de Calidad Sistémica.

Eljabiri, O. & Deek, F. P. Tailoring the Software Process Model to Project requirements. *Citeseer Scientific Literature Digital Library and Search Engine*.

Escalona, M. J. & Koch, N., 2004. Requirements engineering for web applications-a comparative study. *J. Web Eng.*, Vol. 2, pp. 193-212.

Farooq, A. & Arshad, M. J., 2010. A Process model for Developing Semantic Web Systems. *New York Science Journal*, Vol. 3, pp. 43-39.

Figueroa, M. A. A., 2009. MeISE: Metodología de Ingeniería de Software Educativo. *Revista Internacional de Educación en Ingeniería*, Vol. 2.

Flanagan, M. & Nissenbaum, H, 2007. A game design methodology to incorporate social activist themes. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 181-190. ACM.

Gunter, G. A., Kenny, R. F. & Vick, E. H., 2006. A Case for a Formal Design Paradigm for Serious Games. In *Human Systems, Digital Bodies*.

Hans, R., 2013. A Model for Aligning Software Projects Requirements with Project Team Members Requirements.

Huang, B. G. & Yang, J. C., 2012. A Multiplayer Online Role-Playing Game for Incidental Vocabulary Learning. In Proceedings of the 20th International Conference on Computers in Education (ICCE 2012), Singapore.

IEEE Standard Glossary of Software Engineering Terminology, 1990. IEEE Std 610.12-1990, pp. 1-84.

López Beltrán, C., 1983. La creatividad en la divulgación de la ciencia. *Naturaleza*, Vol. 14.

Murphy-Hill, E., Zimmermann, T. & Nagappan, N., 2014. Cowboys, ankle sprains, and keepers of quality: how is video game development different from software development?. In Proceedings of the 36th International Conference on Software Engineering, pp. 1-11. ACM.

Pérez-González, H. G., García, R. M. M. & Cuervo, F. D. C., 2013. Metodología para el desarrollo de proyectos de divulgación utilizando un robot humanoide. XIX Congreso Nacional de Divulgación de la Ciencia y la Técnica, Zacatecas, México, pp. 649-658.

Pressman, R. S., 2005. *Software Engineering: a practitioner's approach*. McGraw-Hill International Edition.

Sánchez Ramos, M. E. & Barradas Bribiesca I., 2014. Divulgación de la Ciencia a través de los dispositivos móviles. *Alternativa Educativa en México*. Congreso Virtual sobre Tecnología, Educación y Sociedad.

Sawyer, P. & Kotonya, G., 2001. Software Requirements. In IEEE SWEBok Project Report. 34, ed: Schwabe, D., Rossi, G.

Shah, A., 2003. OODM: an object-oriented design methodology for development of web applications. Information modeling for internet applications, ed: Patrick van Bommel, pp. 189-229.

Shults, A., 2008. Objectives and tools of science communication in the context of globalization. Doktors der Philosophie, Universität des Saarlandes, Saarbrücken.

Sommerville, I., 2010. Software Engineering. Addison-Wesley, 9th edition.

Susi, T., Johannesson, M. & Backlund, P., 2007. Serious Games – An Overview. University of Skövde HS-IKI-TR-07-001

Taormina, R. J. & Gao, J. H., 2013. Maslow and the motivation hierarchy: Measuring satisfaction of the needs. The American journal of psychology, 126(2), pp. 155-177.

Wieggers, K. E., 2000. Karl Wieggers Describes 10 Requirements Traps to Avoid. Software Testing and Quality Engineering, Vol. 2

Wilson, J., 2005. Indie Rocks! Mapping Independent Video Game Design. Media International Australia, Incorporating Culture & Policy, Vol. 5, pp. 109-122.

Notas biográficas:



Héctor G. Pérez González Obtuvo el grado de Maestro en Ciencias Computacionales en la Universidad Nacional Autónoma de México in 1993 y el de Doctor en Ciencias Computacionales en la Universidad de Colorado en 2003. Sus áreas de interés son la Ingeniería de Software y la Interacción Humano Computadora.



Rosa M. Martínez-García Licenciada en Ciencias de la Comunicación por la Universidad del Centro de México (UCEM). Se desempeñó como Coordinadora de las Licenciaturas en Comunicación Gráfica y Ciencias de la Comunicación de la UCEM de 1996 a 1999 y como Directora de Divulgación del Consejo Potosino de Ciencia y Tecnología (COPOCYT) de 2010 a 2013. Es Miembro Fundador de la Red de Divulgación de Ciencia Tecnología e Innovación (REDICITI) del estado de San Luis Potosí. Actualmente, es Jefa del Departamento de Difusión y Divulgación “D3”, de la Facultad de Ingeniería de la UASLP. Responsable del Programa de Divulgación Ingenialidades y coordinadora del grupo “ingeniosos divulgando”.



Francisco E. Martínez-Pérez Obtuvo el Título de Ingeniero en Computación en la Facultad de Ingeniería de la Universidad Autónoma de San Luis Potosí (UASLP) en 2001, el grado de Maestro en Ciencias Computacionales por la UASLP in 2005 y el grado de Doctor en Ciencias en la Universidad Autónoma de Baja California en 2012. Actualmente es administrador del laboratorio UDICEI. Sus áreas de

interés son la Interacción Humano Computadora, Ingeniería de Software, Cómputo ubicuo y procesamiento de imágenes.



Sandra E. Nava-Muñoz Obtuvo el Título de Ingeniero en Computación en la Facultad de Ingeniería de la Universidad Autónoma de San Luis Potosí (UASLP) en 2001, el grado de Maestro en Ciencias Computacionales por la UASLP in 2005 y el grado de Doctor en Ciencias en la Universidad Autónoma de Baja California en 2013. Sus áreas de interés son la Interacción Humano Computadora, Ingeniería de Software y Cómputo ubicuo.



Alberto Nuñez-Varela Obtuvo el Título de Ingeniero en Computación en la Universidad Autónoma de San Luis Potosí (UASLP) en 2005, el grado de Maestro en Ingeniería en Computación en el Centro de Investigación y Estudios de Posgrado en la UASLP en 2011. Actualmente es estudiante del programa de Doctorado en Ciencias Computacionales y profesor en la Universidad Autónoma de San Luis Potosí. Su área de interés es la Ingeniería de Software, en particular, Calidad de Software y Métricas de software.



Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 2.5 México.

Reducciones temporales para convertir la sintaxis abstracta del diagrama de flujo de tareas no estructurado al álgebra de tareas

Carlos Alberto Fernández-y-Fernández
Universidad Tecnológica de la Mixteca
caff@mixteco.utm.mx

José Angel Quintanar Morales
Universidad Tecnológica de la Mixteca
joseangel@mixteco.utm.mx

Resumen: Este artículo describe nuestro trabajo en el modelado de software usando reducciones temporales para representar diagramas de flujo no estructurado, como una representación intermedia para construir una expresión textual en una álgebra de procesos particular. Este trabajo fue realizado para poder construir una herramienta CASE de apoyo para la fase del modelado de tareas en el Método Discovery para el desarrollo de software. Inicialmente explicaremos las similitudes entre dos tipos de diagramas, el diagrama de actividades de UML y el diagrama de flujo de tareas con su representación formal (el álgebra de tareas). Posteriormente, ofreceremos una explicación explicando la generación automática, usando las reducciones temporales, de expresiones en el álgebra de tareas usando información abstracta que es obtenida de los diagramas de flujo de tareas.

Palabras Clave: Reducciones temporales, modelado visual, diagramas de actividad.

Temporary reductions for converting the abstract syntax from an unstructured task flow diagram to the task algebra

Abstract: The present paper describe our work modeling software using temporary reductions to represent unstructured flow diagrams as an intermediate representation to build textual expression in a particular process algebra. This work was realized in order to build a CASE tool supporting the task modeling phase from the Discovery Method for software development. We begin explaining the similarities between two types of flow diagrams, the UML activity diagram and the task flow diagram with its formal representation (task algebra). Next, we offer an explanation of the work to automatically generate, using the temporary reductions, expressions in the task algebra using abstract information from the task flow diagrams.

Keywords: Temporary reductions, visual modelling, activity diagrams.

1. Introducción

La programación para el modelado de sistemas se refiere al desarrollo de software donde las notaciones gráficas y los componentes visuales manipulables interactivamente son usados principalmente para definir el comportamiento de un sistema (Budinsky et al., 2003).

Para que el modelo de un sistema logre un grado de compresión mayor, se busca semejar uno de los objetivos de la programación visual, el cual es facilitar la comprensión de los programas y simplificar la programación en sí. Más allá, la programación visual debe permitir a los usuarios finales el construir sus propios programas de una manera más sencilla y comprensible (Hernández Valdemar, & León, 2012.). Al cumplir con esta expectativa, las

personas sin conocimientos técnicos que estén involucradas en el proyecto, podrán comprender el comportamiento del sistema modelado (Chonoles, 2003).

Si bien, los lenguajes visuales en su mayoría son creados para que los usuarios poco experimentados en algún tipo de lenguaje de programación puedan llevar a cabo la construcción de sus propios programas, los desarrolladores profesionales también utilizan este tipo de lenguajes (Fernández-y-Fernández et al., 2011). Este enfoque se aplica a la verificación del software. Para iniciar este proceso, primero es indispensable contar con un lenguaje de modelado. Como principal ejemplo tenemos a UML.

UML es utilizado como una notación estandar de modelado visual, ya que posee 14 diferentes diagramas que pueden ser utilizados para representar un sistema de software detallando diversos aspectos, así como diversas perspectivas.

Desde el año de 1997 y bajo la supervisión del OMG, ha evolucionado, sin embargo su notación se ha vuelto más compleja, con la finalidad de abarcar todo el proceso completo para el modelado de sistemas, así como de procesos (Graham, & Simons,1999). Esto ha originado algunas críticas respecto a su composición semántica y las diferentes formas de interpretar los modelos.

El problema principal, es la ambigüedad con la que se pueden crear los diversos diagramas existentes en el lenguaje, esto provoca que se carezca de un sentido específico y la validez del mismo no puede establecerse con la claridad necesaria (Graham, & Simons,1999), (Schmuller, 2009).

Existen otras alternativas a UML, con la cual se pueden modelar sistemas de una forma similar a la que presenta el diagrama de actividades de UML, por ejemplo la notación del diagrama de flujo de tareas del Método Discovery. Este diagrama de flujo de tareas y su respectiva notación, puede tomarse como un subconjunto del diagrama de actividades definido en UML (Simons, 1998). Fue

probado en diversos proyectos por estudiantes de posgrado de la Universidad de Sheffield. La notación simple y sin ambigüedades lo hace una opción viable y consistente en comparación con UML.

1.1 El diagrama de actividades

El diagrama de actividades muestra un flujo de acciones, generalmente secuenciales, el propósito de un diagrama de actividades es: Modelar el flujo de las tareas. Modelar las operaciones. Todos los diagramas de actividades tienen una serie de elementos básicos, su comportamiento se describe mediante la integración de los elementos siguientes:

- Particiones.
- Nodos de acción.
- Nodos de control.
- Nodos de Objeto.
- Control de flujo.

Es a partir de estos objetos que se pueden modelar sistemas completos, sin embargo, como se mencionó en la sección anterior, presenta ciertas ambigüedades. Una posible mejora estaría dada por una notación orientada a objetos más sencilla y pequeña, fácil de aprender, que sea más exacta y replicable, dicha notación debe estar soportada por un lenguaje formal para representar la semántica de una manera precisa, de manera que los modelos se puedan verificar unos con otros (Simons, 2006).

Una posible solución para tal notación es restringir un perfil UML, denotado por el Método Discovery y definido en el diagrama de flujo de tareas, el cual pone especial atención al minimalismo y la coherencia.

1.2 El diagrama de flujo de tareas

Durante el proceso de modelación del sistema, se necesitará realizar la identificación de tareas y la estructuración de relaciones para las diferentes perspectivas de los actores en el modelo, se deberá representar el flujo de las relaciones de trabajo. El Método Discovery representa este flujo de trabajo por medio del Diagrama de flujo de tareas.

Básicamente un diagrama de flujo de tareas representa el orden en que las tareas se llevan a cabo en la organización, también denota la relación lógica entre todas las tareas que componen el diagrama.

La notación utilizada en el Método Discovery se basa en el diagrama de actividades de UML, sin embargo, se mantiene constante la figura de una elipse para la representación de una tarea. Con la inclusión del componente final de flujo en UML 2.0 (círculo cruzado por dos líneas inclinadas), se cubre la funcionalidad de Failed (círculo cruzado por una sola línea inclinada) logrando una compatibilidad mayor, con un ligero cambio en la representación visual. En la tabla 1 se muestra la notación para un diagrama de flujo de tareas, así como el nombre y un identificador numérico asociado al tipo de componente (Fernández-y-Fernández, 2010).

Componente visual	Nombre	Identificador numérico
●	<i>Start</i>	0
○	<i>Task</i>	1
⌵	<i>Fork</i>	2
⌶	<i>Join</i>	3
▷	<i>Exception</i>	4
⊘	<i>Failure</i>	5
◇	<i>Choice</i>	6
⦿	<i>End</i>	7

Tabla 1.

Componentes visuales para el diagrama de flujo de tareas.

1.3 Del diagrama de flujo de tareas al álgebra de tareas

Una vez que los elementos gráficos han sido presentados, se detallará como es que estos componentes gráficos poseen una representación semántica la cual es representada por medio del álgebra de tareas, con ello se tiene la representación semántica de la que carece UML.

La representación formal del diagrama de flujo de tareas está dada por el álgebra de tareas, ésta álgebra es una traducción o representación abstracta de dicho diagrama, la sintaxis abstracta del álgebra de tareas se muestra en la Figura 1.

1.4 Naturaleza no estructurada del diagrama de flujo de tareas

La meta del diseño de los diagramas de flujos de tareas, es aprovechar el comportamiento de los diagramas de actividad de UML, con la intención de brindar una facilidad a los usuarios y tener una mejor aceptación y difusión.

Esta decisión tiene puntos a favor y en contra, se puede tomar como un ejemplo práctico el lenguaje HTML. HTML es muy flexible, y los exploradores HTML aceptan cualquier cosa que se parezca a HTML, esta característica ayudó a la temprana adopción de HTML, pero ahora es un problema la solución para esta problemática en particular fue la aparición de lenguajes más estrictos (Benot, 2001).

En resumen, implementar la automatización de la traducción de un diagrama no estructurado, significa no poder utilizar métodos o técnicas que si podrían utilizarse a diagramas estructurados, por tal motivo se debe emplear otro tipo de técnicas.

Activity ::= ε	-- empty activity
σ	-- <i>succeed</i>
ϕ	-- <i>fail</i>
Task	-- a single task
Activity ; Activity	-- a sequence of activity
Activity + Activity	-- a selection of activity
Activity Activity	-- parallel activity
$\mu x.(Activity ; \varepsilon + x)$	-- until-loop activity
$\mu x.(\varepsilon + Activity ; x)$	-- while-loop activity
Task ::= Simple	-- a simple task
{ Activity }	-- encapsulated activity

Figura 1.

Sintaxis abstracta del álgebra de tareas [10].

2. Definición del problema

Como se mencionó en la sección 1.4, se debe mantener la simplicidad en cuanto al diseño de los diagramas de flujos de tareas, sin embargo, se presenta una problemática dada esta misma simplicidad, la causa es que a partir del elemento Choice, mostrado en la tabla 1, se pueden construir tres estructuras tal como lo demuestra la sintaxis abstracta en la Figura 2, dichas estructuras son:

- Selección binaria figura 2.a.
- Ciclo Until figura 2.b.
- Ciclo While figura 2.c.

De tal manera que la generación de la respectiva álgebra no pudo realizarse en una pasada, ya que para llevar a cabo el proceso completo destinado a obtener el álgebra de tareas es necesario realizar varias pasadas. Esta forma de generar el álgebra de tareas se debe a que se necesita analizar una ramificación completa de un diagrama de flujos de tareas y así detectar que estructura de la sintaxis del álgebra de tareas representa. Cabe mencionar que los diagramas son diseñados con la herramienta presentada en (Fernández-Santos, H. et al., 2011).

3. Reducciones temporales

Por los motivos anteriores, se optó por la vía de reducción a expresiones temporales, una expresión temporal la podemos definir como, una estructura compuesta por componentes básicos de la sintaxis del álgebra de tareas, que a su vez tiene un representación dentro de la misma sintaxis abstracta.

La reducción a expresiones temporales retoma el concepto de la optimización local de la fase de generación y optimización de código del desarrollo de compiladores (Aho, A. V. et al., 1990), dicho proceso consiste en examinar

grupos de instrucciones para sustituirlas por instrucciones más baratas. Las optimizaciones pueden dar lugar a otras optimizaciones de manera que es necesario repetir varias veces la optimización local para obtener la mejora del código (Teufel, B. et al., 1995). Por su parte la reducción de expresiones temporales trata de obtener un diagrama elemental formado por un inicio y un componente terminal.

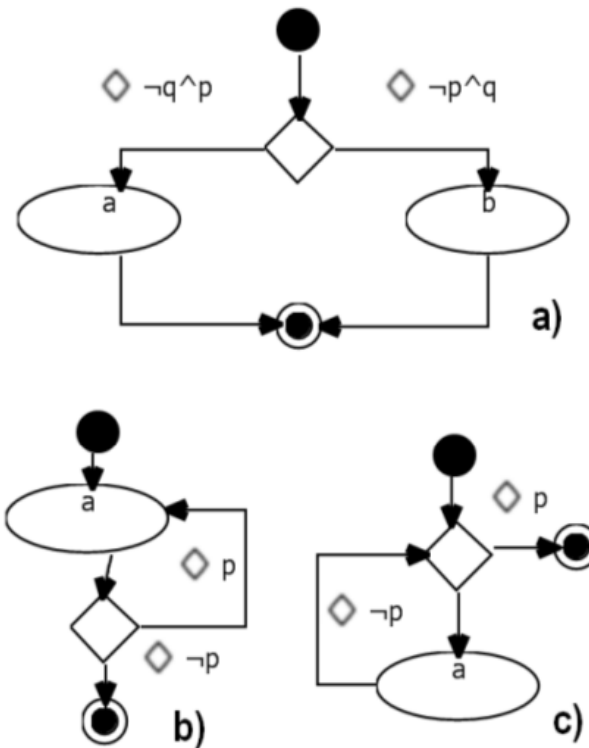


Figura 2.

Estructuras construidas con el componente Choice.

3.1 Reducción de expresiones temporales

Para llevar a cabo la reducción del diagrama, se tiene la necesidad de abstraer el modelo visual y almacenarlo en las estructuras de la figura 3 y figura 4, y realizar una cantidad indeterminada de pasadas para formar estructuras completas con los componentes hasta tener un diagrama elemental.

A continuación se describen los datos que almacenan cada uno de los campos de la estructura 3.

- NOMBRE: el nombre del componente gráfico con un valor de texto.
- TIPO: el tipo de identificador conforme la tabla 1.
- ID: identificador único asignado automáticamente.
- OP1: campo destinado a información introducida por el usuario, en este caso para los componentes Task, Choice y Exception.
- OP2: campo destinado a información introducida por el usuario, en este caso para los componentes Choice, Exception.
- ENTRADA T: número total de entradas del componente.
- SALIDA T: número total de salidas del componente.
- MARCA: campo auxiliar libre y sin representación de información.

Ahora se detallará la información que contiene la estructura dinámica 4 que representa los flujos del sistema.

- TIPO: el tipo de identificador del componente origen conforme la tabla 1.
- ORIGEN: id único del componente origen.
- TIPO: el tipo de identificador del componente destino conforme la tabla 1.
- DESTINO: id único del componente destino.
- RAMA: campo para validación de ramas.
- MARCA: campo auxiliar libre y sin representación de información.
- ENT_T: número total de entradas del componente.
- SALT_: campo auxiliar libre y sin representación de información.

3.1.1 Reducción de tareas lineales

Esta reducción se enfoca a la detección y posterior unión de tareas lineales. Una tarea lineal se define como la sucesión de componentes Task, la fusión de un par de tareas lineales da como origen a una tarea compuesta. Para llevar a cabo este proceso, primero se deben determinar el número de entradas y

salidas de los componentes Task, si para ambos es uno, entonces se procede a la fusión.

El procedimiento para fusionar las tareas se puede explicar suponiendo la existencia de dos tareas, A con destino B, y consiste en:

1. Obtener el id destino de la tarea B.
2. Obtener la información de la localidad marca de B.
3. Sustituir el destino de la tarea A por el de B.
4. Fusionar el valor de la localidad marca de A con el de B.
5. Eliminar a B de la estructura flujos.

NOMBRE	TIPO	ID	OP1	OP2	ENTRA T	SAL T	MARCA
0	1	2	3	4	5	6	7

Figura 3.

Estructuras para abstraer información de los componentes visuales.

TIPO	ORIGEN	TIPO	DESTINO	RAMA	MARCA	ENT_T	SAL_T
0	1	2	3	4	5	6	7

Figura 4.

Sintaxis abstracta del álgebra de tareas [10].

3.1.2 Reducción de tareas compuestas

El proceso para reducir tareas compuestas lleva a cabo un proceso similar al método para tareas lineales, excepto que éste opera sobre tareas compuestas. Una tarea compuesta se define como la sucesión de componentes reducidos, la fusión de un par de tareas compuestas da como origen a otra tarea compuesta. Para llevar a cabo este proceso, primero se deben determinar el número de entradas y salidas de las tareas compuestas, si para ambos es uno, entonces se procede a la fusión.

El procedimiento para fusionar las tareas compuestas se puede explicar suponiendo la existencia de dos elementos, A con destino B, y consiste en:

1. Obtener el id destino del elemento B. O
2. Obtener la información de la localidad marca de B.
3. Modificar el tipo de componente en las estructuras dinámicas flujos y componentes.
4. Sustituir el destino de la tarea A por el de B.
5. Fusionar el valor de la localidad marca de A con el de B.
6. Eliminar a B de la estructura flujos.

La eliminación del componente B es relativo, ya que ahora quedará contenido en A.

3.1.3 Reducción de tarea lineal con terminal

La reducción consiste en generar un elemento terminal etiquetado como final. Definimos al elemento terminal como la estructura compuesta por una secuencia de tareas lineales o tareas compuestas seguido de un componente End, o un componente Failure o un elemento terminal. Para obtenerlo, es necesario identificar una secuencia de tarea lineal o compuesta que apunta a un elemento terminal, una vez hecho esto, se lleva a cabo el siguiente procedimiento.

1. Identificar el tipo de componente C.
2. Si no se tratará de un componente End o Failure, se obtiene el valor de la localidad marca de la estructura flujos.
3. Se obtiene el valor de la localidad marca de B.
4. Se sustituye destino de A por el valor de C.
5. Se fusiona la localidad marca de A con la B y con C, sólo si éste no fuera un componente End o Failure.
6. Se marca a B como elemento terminal.
7. Se elimina a C.

3.1.4 Reducción del ciclo For

Basa su búsqueda únicamente en los componentes Choice, en cada recorrido donde se encuentre este elemento, será analizada la posible formación de una estructura For. Cuando se detecta una estructura de este tipo, y suponiendo que se tiene la estructura de la figura 5 donde el elemento Choice apunta hacia A para formar la estructura For y hacia B cuando la condición no se cumple, tenemos que:

1. Se identifica el flujo que da salida al For, es decir el flujo que seguiría cuando la condición de iteración ya no se cumpla, en este caso B.
2. Se obtiene el valor de la localidad marca del componente al que apunta el Choice cuando la condición se cumple, componente A.
3. A la localidad marca del Choice se le asigna la información obtenida en el paso 2 bajo el formato $\text{Mu}(\text{Epsilon}+\text{datos};x)$.
4. Se elimina el componente A.

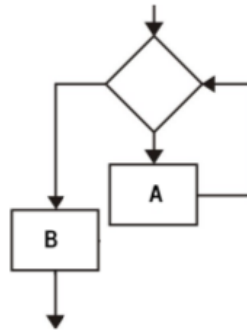


Figura 5.

Estructura For identificada para ser reducida.

3.1.5 Reducción del ciclo Until

De la misma forma en que el proceso para reducir una estructura For, se analizan los componentes Choice. se busca la posible formación de estructuras Until, con la diferencia de que el componente interno debe ser a su vez origen y destino, además el componente interno debe poseer dos flujos de entrada y uno de salida.

El procedimiento para reducir una estructura Until es el siguiente, suponiendo se tiene la imagen de la Figura 6, donde el elemento A apunta un Choice, éste último apunta al elemento A y finalmente cuando la condición no se cumple, el elemento Choice apunta al elemento B, tenemos que:

1. Se identifica el destino del componente Choice cuando la condición no se cumple, es decir cuando el flujo apunta a B.
2. Se sustituye el destino de A, por el destino obtenido anteriormente, es decir, el flujo de A tendrá como destino a B.
3. El campo marca del componente A se modifica para tener el formato $\text{Mu}(\text{datos}; \text{Epsilon}+x)$.
4. El componente Choice es eliminado.

3.1.6 Reducción del paralelismo

La identificación de la estructura Fork-Join resulta hasta cierto punto sencilla. Para su identificación se necesita que por cada componente Fork, exista una tarea lineal o compuesta seguido de un componente Join, si hubiese una tarea terminal, no es necesario que esta apunte a un componente Join. Para su reducción se lleva a cabo el siguiente procedimiento.

Suponiendo que se tiene la estructura de la Figura 7, donde el componente Fork apunta a las tareas compuestas A1, A2 ,..., An donde todas las tareas desde A1 hasta An apuntan a un único componente Join y éste último apunta a un componente B, tenemos qué:

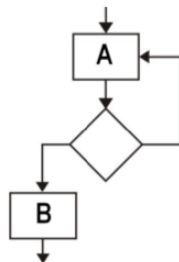


Figura 6.

Estructura Until identificada para ser reducida.

1. El contenido de la localidad marca del componente Fork se fusiona con el elemento A_1 mediante el formato $A_1 || A_2 || \dots A_n$.
2. Se elimina el componente A_1 .
3. Esto se repite hasta A_n .
4. Se sustituye el destino del componente Fork por el destino del componente Join, en este caso B.
5. Se elimina el componente Join.

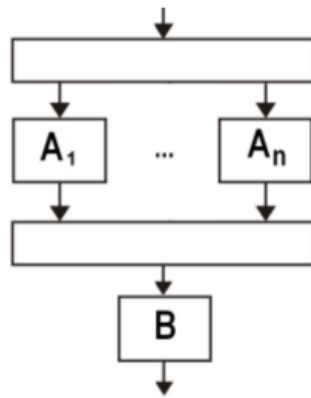


Figura 7.

Estructura Fork/Join identificada para ser reducida.

3.2 Reducción de la estructura Or

La reducción a la estructura Or es la de mayor complejidad, ya que existen 6 diferentes formas de reducir una estructura Or.

1. Or simple.
2. Or terminal doble.
3. Or terminal simple.
4. Or terminal simple vacío.
5. Or vacío simple.
6. Or vacío doble

A continuación se detalla el proceso de reducción para cada una de ellas.

3.2.1 Reducción de un Or simple

Esta reducción es considerada como el mejor de los casos, ya que este tipo de estructura posee contenido en ambos destinos del componente Choice, por lo tanto no es necesario llevar a cabo ninguna evaluación. Para llevar a cabo este proceso suponemos que se tiene la estructura de la Figura 8:

Los destinos del componente Choice se dirigen a un elemento A y B respectivamente, y cada uno de estos componentes tiene como destino común al elemento C, luego.

1. Se identifica el destino de A ó B, en este caso el elemento C.
2. Se obtiene el valor de la localidad marca de A y de B.
3. Se crea el contenido de la localidad marca del componente Choice bajo el formato (A+B).
4. Se sustituye en el componente Choice el destino B por C.
5. Se elimina el destino A y B.
6. Se modifica el tipo de componente de Choice a Or.

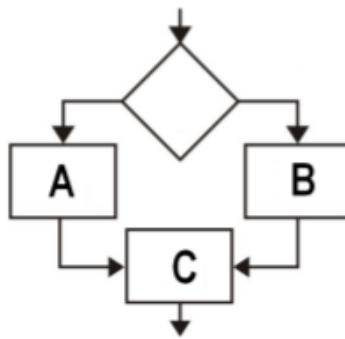


Figura 8.

Estructura Or simple identificada para ser reducida.

3.2.2 Reducción de un Or terminal doble

Este tipo de reducción es un caso especial de Or, ya que se convertirá en un elemento terminal, esto, debido a que ambos destinos del componente Choice tienen como destino alguna combinación de los elementos A y B del tipo End, Failure o un componente terminal. El proceso de reducción de la estructura de la Figura 9 se detalla a continuación.

1. Se identifica el tipo de componente destino del elemento Choice, si es un componente terminal se almacena el valor de la localidad marca, en caso contrario se almacena el tipo.
2. Se identifica el tipo de componente del siguiente destino del elemento Choice, si es un componente terminal se almacena el valor de la localidad marca, en caso contrario se almacena el tipo.
3. Si uno o ambos tipos de destino corresponden a un componente terminal esta información se almacena en la localidad marca del elemento Choice, bajo el formato(A+B).
4. Si ambos componentes son elementos del tipo End o Failure o alguna combinación de ellos, se almacena el tipo del componente en la localidad marca del elemento Choice bajo el formato (A+B).
5. Se modifica el tipo de componente de Choice a Terminal.
6. Se eliminan los componentes A y B.

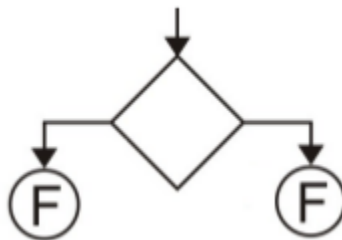


Figura 9.

Sintaxis abstracta del álgebra de tareas (Fernández-y-Fernández, 2010).

3.2.3 Reducción de un Or terminal simple

Este tipo de reducción es un caso especial del Or, ya que como se observa en la Figura 10 uno de los destinos del componente Choice es un componente terminal y el otro destino, por el contrario es un elemento no terminal el cual pertenece a la estructura Or, es decir el único flujo que apunta a dicho componente no terminal, es el flujo destino del elemento Choice. Para el proceso de reducción, definimos un componente Choice que tiene como uno de sus destinos a un elemento terminal F, el otro destino se dirige a un elemento no terminal A, el cual a su vez tiene como destino un componente B. Entonces:

1. Se identifica el componente terminal F, si es un elemento End o Failure, se guarda su tipo de componente, en caso contrario se obtiene el valor de la localidad marca de la estructura dinámica Flujos.
2. Se obtiene el valor de la localidad marca del componente no terminal A.
3. Se obtiene el destino del componente A, en este caso el elemento B.
4. Se fusiona el valor obtenido de los componentes F y A bajo el formato (F+A).
5. Se sustituye el destino A del componente Choice por B.
6. Se elimina el componente F.
7. Se elimina el componente A.
8. Se modifica el tipo de componente Choice por Or.

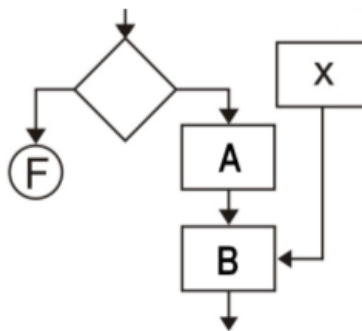


Figura 10.

Estructura Or terminal simple identificada para ser reducida.

3.2.4 Reducción de un Or terminal simple vacío

Este tipo de reducción es similar a la llevada a cabo en Or terminal simple, la variación que presenta, es que ningún componente no terminal pertenece a la estructura Or, tal como se muestra en la Figura 11. Para el proceso de reducción, definimos un componente Choice que tiene como uno de sus destinos a un elemento terminal F, el otro destino se dirige a un elemento no terminal A el cual no pertenece a la estructura Or, es decir el elemento A también es destino de otro componente. Por lo tanto tenemos que:

1. Se identifica el componente terminal F, si es un elemento End o Failure, se guarda su tipo de componente, en caso contrario se obtiene el valor de la localidad marca de la estructura dinámica Flujos.
2. Se fusiona el valor obtenido del componentes F y el simbolo Epsilon bajo el formato (F+Epsilon).
3. Se elimina el componente F.
4. Se modifica el tipo de componente Choice por Or.

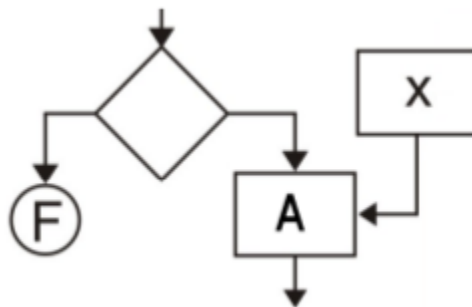


Figura 11.

Estructura Or terminal simple vacía identificada para ser reducida.

3.2.5 Reducción de un Or terminal vacío simple

La reducción de este componente es un tanto simple, similar al proceso llevado a cabo en `orTerminalSimple`, con la excepción de que el componente terminal no existe, ese destino se dirige a un componente que está fuera de la estructura Or, tal como se muestra en la Figura 12. Este proceso de reducción se inicia suponiendo que el componente Choice tiene como uno de sus destinos al elemento A, y como segundo destino al elemento B, el cual también es destino del elemento A. Dadas estas suposiciones tenemos que:

1. Se obtiene el valor de la localidad marca del componente A en la estructura dinámica flujos.
2. Se sustituye el valor de la localidad marca del componente Choice por el contenido de A bajo el formato (A+Epsilon).
3. Se elimina el componente A.
4. El tipo de componente del Choice es modificado a Or.

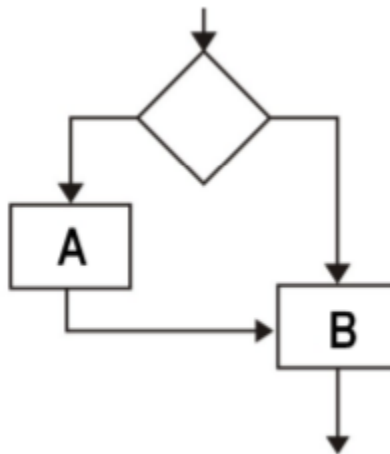


Figura 12.

Estructura Or vacía simple identificada para ser reducida.

3.2.6 Reducción de un Or vacío doble

Esta reducción es la más simple para la estructura Or y corresponde a la mostrada en la Figura 13. Para llevarla a cabo se identifica la estructura de la siguiente forma. El componente Choice tiene ambos destinos dirigidos a un único componente A, por lo tanto se realiza el siguiente proceso.

1. Se sustituye el valor de la localidad marca del componente Choice por la cadena (Epsilon+Epsilon).
2. Se elimina un flujo que apunta hacia A. El tipo de componente Choice se modifica por un Or.

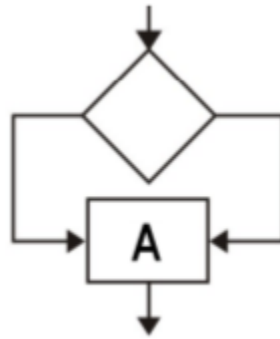


Figura 13.

Estructura Or vacía doble identificada para ser reducida.

4. Resultados

Como resultado final se obtiene el álgebra de tareas asociado a un diagrama de flujo de tareas, sin embargo, para comprobar la salida, es necesario utilizar el compilador del álgebra de tareas, desarrollado en (Fernández-y-Fernández, 2010). Dicho compilador transforma la expresión del álgebra de tareas y, si la expresión es correcta, genera un conjunto de trazas, una sola traza es una cadena simbólica que denota una ruta de posible ejecución a través del sistema y el conjunto de trazas denota todas las rutas de ejecución posibles (Fernández-y-Fernández, 2010).

4.1 Estructuras de reducción

Como se ha explicado, el plugin genera el álgebra de tareas al llevar a cabo una reducción de componentes a estructuras temporales; cada una de estas estructuras se pusieron a prueba; para ello se creó el diagrama correspondiente y se obtuvo el álgebra asociada al diagrama. El plug-in genera el álgebra de tareas asociado a un diagrama de flujo de tareas, sin embargo, para comprobar la salida, es necesario utilizar el compilador del álgebra de tareas. Dicho compilador transforma la expresión del álgebra de tareas y, si la expresión es correcta, genera las trazas correspondientes para la expresión. El resultado es un sistema de tareas dado como un conjunto de trazas, en el que una sola traza es una cadena simbólica que denota una ruta de posible de ejecución a través del sistema y el conjunto de trazas denota todas las rutas de ejecución posibles (Fernández-y-Fernández, 2010). A continuación se presentan las estructuras temporales implementadas en el plug-in.

4.1.1 Estructura lineal

La estructura lineal de tareas (mostrada en la Figura 14) es la composición básica para reducir diagramas, su correspondiente álgebra de tareas se muestra en la Tabla 2.

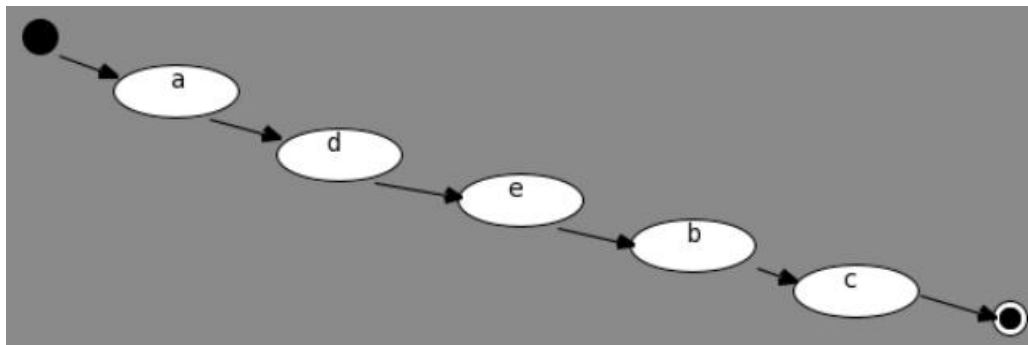


Figura 14.

Diagrama de flujos de tareas mostrando una estructura lineal.

```
{a;d;e;b;c;Sigma}  
fromList [[a,d,e,b,c]]
```

Tabla 2.

Álgebra de tareas para una estructura lineal y las trazas derivadas del álgebra.

4.1.2 Estructura Fork

Una forma compleja de la estructura Fork se muestra en la Figura 15, ya que contiene un elemento terminal el cual no posee un flujo que vaya hacia el componente Join. El álgebra de tareas de este diagrama y las trazas originadas derivadas del álgebra de tareas se presentan en Tabla 3.

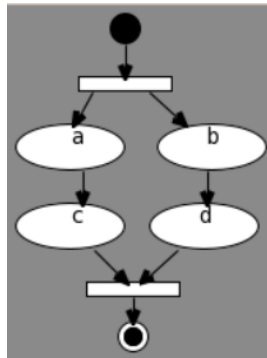


Figura 15.

Diagrama de flujos de tareas mostrando una estructura Fork/Join.

```
{((a;c)||b;d);Sigma}  
fromList [[a,b,c,d],[a,b,d,c],[a,c,b,d],[b,a,c,d],[b,a,d,c],[b,d,a,c]]
```

Tabla 3.

Álgebra de tareas para una estructura Fork/Join y las trazas derivadas del álgebra.

4.1.3 Estructura Or

La estructura or presenta una serie de variantes, debido a que puede contener componentes terminales o vacíos, la Tabla 4 muestra las posibles formas de construir la estructura or, así como su correspondiente álgebra de tareas.

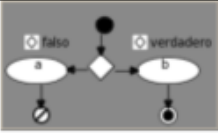

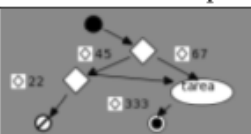
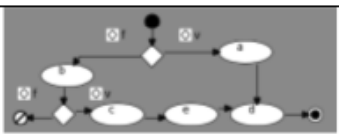

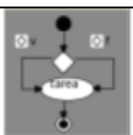
Diagrama de flujo de tareas	Álgebra de tareas	Trazas
 <p>Estructura Or terminal doble.</p>	$\{(a;\Phi)+(b;\Sigma)\}$	<code>fromList [[!,a,Phi],[!,b]]</code>
 <p>Estructura Or simple.</p>	$\{((b)+(a);\Sigma)\}$	<code>fromList [[!,a],[!,b]]</code>
 <p>Estructura Or terminal simple vacío.</p>	$\{(((\Phi+\text{Epsilon}))+\text{Epsilon});\text{tarea};\Sigma)\}$	<code>fromList [[!,tarea],[!,Phi]]</code>
 <p>Estructura Or terminal simple.</p>	$\{((a)+(b;(\Phi+(c;e))));\text{d};\Sigma)\}$	<code>fromList [[!,a,d],[!,b,!c,e,d],[!,b,!Phi]]</code>
 <p>Estructura Or vacío simple.</p>	$\{((a)+\text{Epsilon});b;\Sigma)\}$	<code>[[!,a,b],[!,b]]</code>
 <p>Estructura Or vacío doble.</p>	$\{(\text{Epsilon}+\text{Epsilon});\text{tarea};\Sigma)\}$	<code>fromList [[tarea]]</code>

Tabla 4.

Posibles formas de construir la estructura Or.

4.1.4 Estructura Until

La forma básica de la estructura Until se presenta en la Figura 16, de igual forma en Tabla 5 se muestra el álgebra de tareas asociado a dicha estructura y sus correspondientes trazas originadas por el álgebra de tareas.

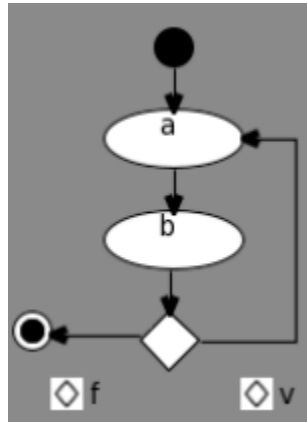


Figura 16.

Diagrama de flujos de tareas mostrando una estructura Until.

```
{Mu.x((a;b);Epsilon+x);Sigma}  
fromList [[a,b,!],[a,b,!,a,b]]
```

Tabla 5.

Álgebra de tareas para una estructura Until y las trazas derivadas del álgebra.

4.1.5 Estructura For

La Figura 17, muestra un diagrama de flujo de tareas con la estructura For. El álgebra de tareas de este diagrama y las trazas originadas derivadas del álgebra de tareas se presentan en la Tabla 6.

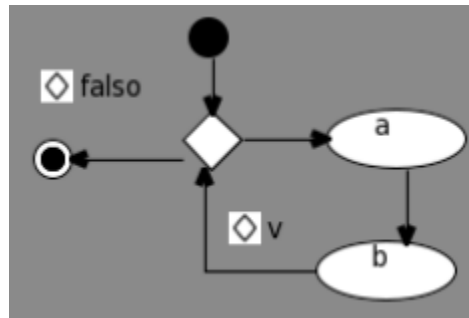


Figura 17.

Diagrama de flujos de tareas mostrando una estructura For.

```
{Mu.x(Epsilon+(a;b);x);Sigma}  
fromList [[!],[!,a,b,!],[!,a,b!,a,b]]
```

Tabla 6.

Álgebra de tareas para una estructura For y las trazas derivadas del álgebra.

4.2 Generación con un diagrama de flujos completo

Para ejemplificar la correcta generación del álgebra de tareas mediante la reducción de expresiones temporales, se utiliza parte del trabajo propuesto en (Fernández-y-Fernández, C. A. et al., 2012), se analiza un diagrama de flujo de tareas obtenido a partir del análisis de un caso de uso.

El diagrama de la Figura 14 se compone de cinco tareas, una de ellas conforma un ciclo de repetición Until, dos más se realizan de forma paralela, las tareas son las siguientes.

1. Proyectar la palma de la mano sobre el componente deseado.
2. El componente es sombreado y anclado al cursor.
3. Buscar posición del componente.
4. Emitir alerta visual.
5. Asignar el ID del componente.

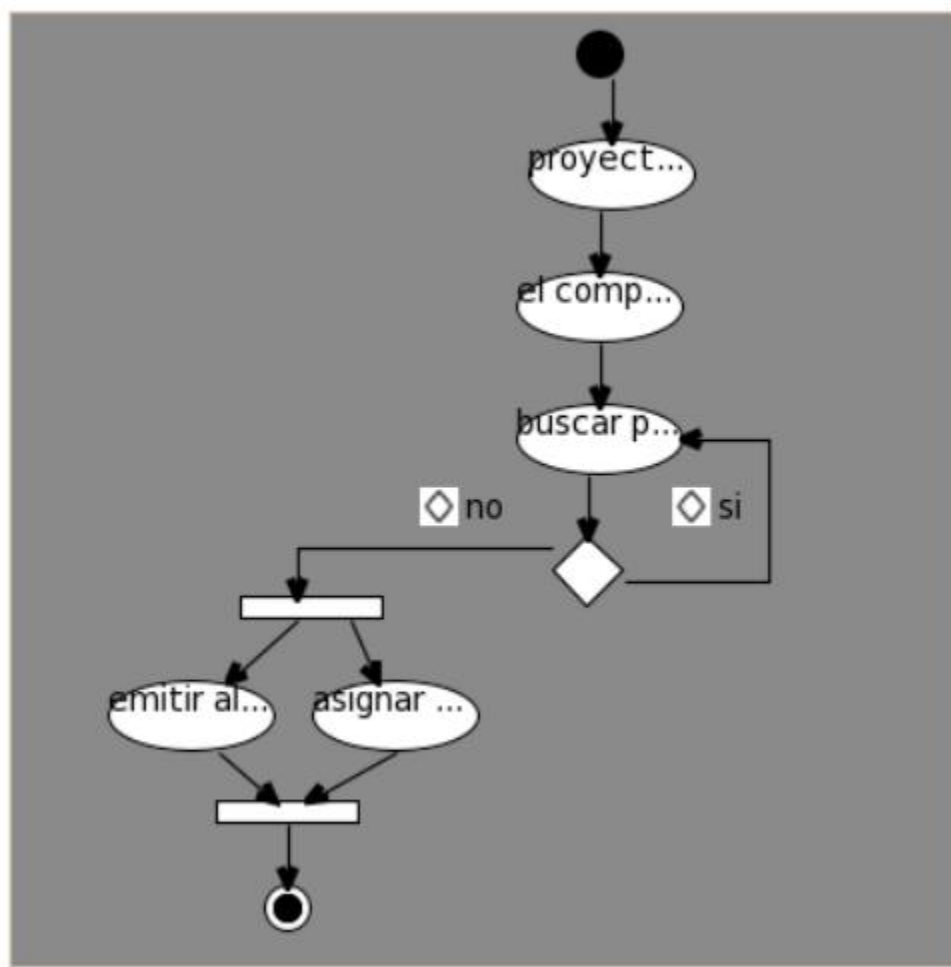


Figura 18.

Diagrama de flujo de tareas diseñado sobre el plugin.

Para una mejor comprensión del álgebra de tareas generado, se muestra la tabla 7, donde el contenido ha sido sangrado de acuerdo al estilo Allman. Cuando se obtiene el álgebra de tareas correspondiente al diagrama de flujos de tareas, esta álgebra es procesada por el compilador del álgebra de tareas. Como resultado se obtienen las trazas de la tabla 8. Al obtener las trazas correctas correspondientes al diagrama de flujo de tareas, queda de manifiesto que los resultados obtenidos son válidos.

```
{
  proyectarLaPalmaDeLaManoSobreElComponenteDeseado;
  elComponenteEsSombreadoYAncladoAlCursor;
  Mu.x
  (
    (
      buscarPosiciónDelComponente
    );
    Epsilon
    +
    x
  );
  (
    emitirAlertaVisual
    ||
    asignarElIDDelComponente
  );
  Sigma
}
```

Tabla 7.

Álgebra de tareas del diagrama de la Figura 14 sangrado con el estilo Allman.

```
fromList
[[proyectarLaPalmaDeLaManoSobreElComponenteDeseado,
elComponenteEsSombreadoYAncladoAlCursor,
buscarPosiciónDelComponente,!,asignarElIDDelComponente,
emitirAlertaVisual],
[proyectarLaPalmaDeLaManoSobreElComponenteDeseado,
```

```
elComponenteEsSombreadoYAncladoAlCursor,  
buscarPosiciónDelComponente,!,  
buscarPosiciónDelComponente,asignarElIDDelComponente,  
emitirAlertaVisual],[  
proyectarLaPalmaDeLaManoSobreElComponenteDeseado,  
elComponenteEsSombreadoYAncladoAlCursor,  
buscarPosiciónDelComponente,!,  
buscarPosiciónDelComponente,emitirAlertaVisual,  
asignarElIDDelComponente],[  
proyectarLaPalmaDeLaManoSobreElComponenteDeseado,  
elComponenteEsSombreadoYAncladoAlCursor,  
buscarPosiciónDelComponente,!emitirAlertaVisual,  
asignarElIDDelComponente]]
```

Tabla 8.

Trazas correspondientes al diagrama de flujo de tareas trazar componente.

Una imagen del pulgín ejecutándose en Eclipse puede verse en la Figura 19, donde se muestra el diagrama realizado dentro del mismo IDE.

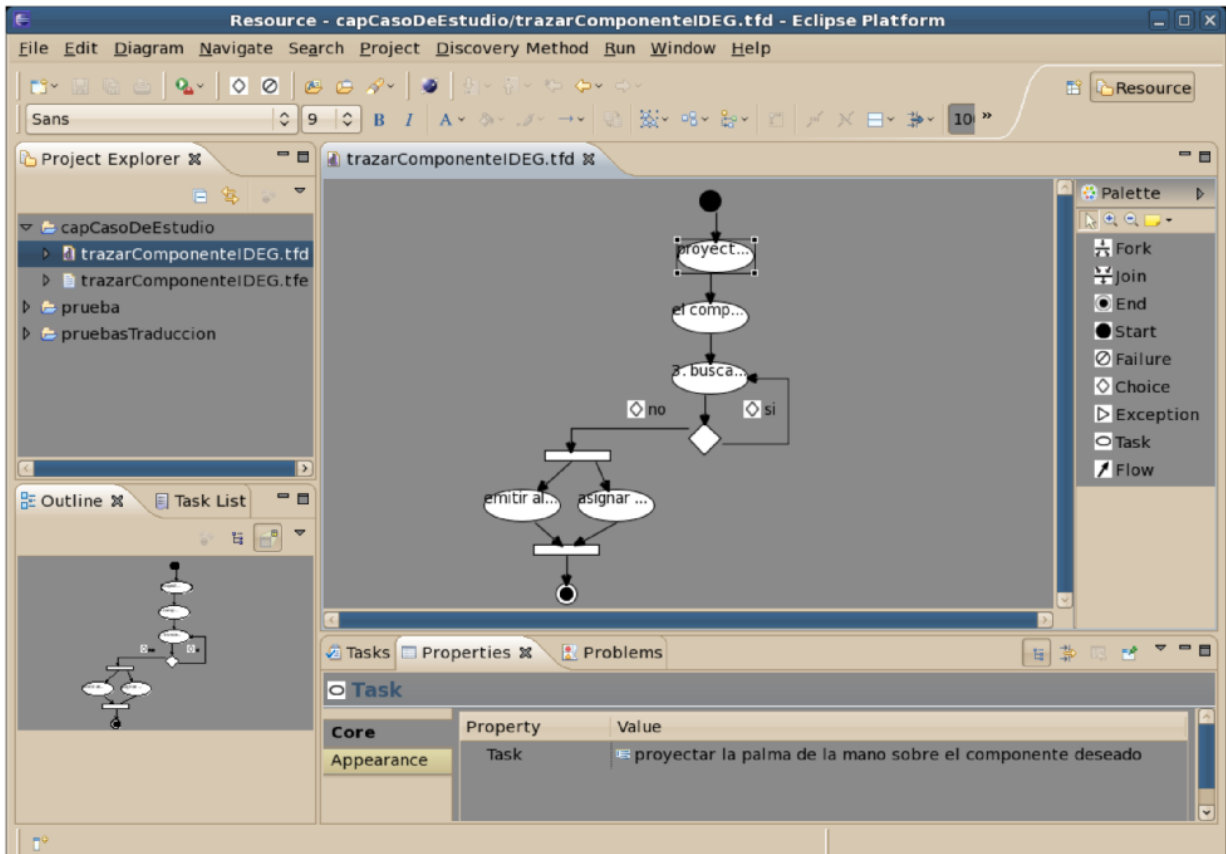


Figura 19.

Diagrama de flujos de tareas diseñado sobre el plugin.

5. Conclusiones

Generar el álgebra de tareas por medio de la utilización de expresiones temporales, es una forma adecuada, sobre todo al saber que los diagramas de flujo de tareas tienen la característica de ser no estructurados. Al usar esta técnica, se evita la necesidad de modificar los componentes visuales (Tabla 1) pudiendo mantener el objetivo planteado de hacer del diagrama de flujo de tareas algo sencillo de construir, además, se mantiene la similitud con el diagrama de actividades definido en UML.

Una de las ventajas de utilizar esta técnica de reducción de expresiones temporales, es que se podría diseñar e implementar una herramienta que

refleje de forma visual dichas reducciones, es decir, por cada reducción dada por una expresión temporal, el diagrama debe ser modificado para reflejar esa expresión temporal específica. De esta manera al realizar paso a paso la reducción completa de un diagrama de flujo de tareas, se puede llegar a una compresión más rápida del proceso para generar el álgebra de tareas. También se reconoce la necesidad de establecer mejoras en el álgebra de tareas, tal como la que se presentó en (Fernández-y-Fernández, 2012).

Agradecimientos

Esta investigación fue financiada parcialmente por la Universidad Tecnológica de la Mixteca.

Referencias

Aho, A.V., Sethi, R., Ullman, J.D. & Flores-Suarez, P. 1990. *Compiladores: principios, técnicas y herramientas*. México: Addison-Wesley Iberoamericana.

Benot, M. 2001. *XML con ejemplos*. México: Pearson education.

Budinsky, F., Steinberg, D., Merks, E., Ellersick, R. & Grose, T.J. 2003. *Eclipse Modeling Framework: A Developer's Guide*. Boston, Massachusetts: Addison Wesley, first ed.

Chonoles, M. J. 2003. *UML 2 for dummies*. 909 Third Avenue New York, NY 10022: Wiley Publishing, Inc.

Fernández-Santos, H., Fernández-y-Fernández, C.A. & Quintanar-Morales, J.A. 2011. An idea to build and check task flow models, *Advances in Computer Science and Applications, Research in Computer Science*, vol. 53, pp. 23–33.

Fernández-y-Fernández, C.A. & Quintanar-Morales, J.A. 2012. Integrated development environment gesture for modeling workflow diagrams, *Congreso Internacional de Investigación e Innovación en Ingeniería de Software (Conisoft 2012)*, vol. abs/1205.0751.

Fernández-y-Fernández, C.A. 2010. *The Abstract Semantics of Tasks and Activity in the Discovery Method*. PhD thesis, The University of Sheffield, Sheffield, UK.

Fernández-y-Fernández, C.A. 2012. Towards a new metamodel for the task flow model of the discovery method, *Congreso Internacional de Investigación e Innovación en Ingeniería de Software*.

Fernández-y-Fernández, C.A., Acosta-Mesa, H.G. & Cruz-Ramírez, N. 2011. Apuntes para un aprendiz de programador app inventor, programación de dispositivos móviles al alcance de todos, Temas de ciencia y Tecnología, vol. 15.

Graham, I. & Simons, A.J.H. 1999. 30 things that go wrong in object modelling with uml 1.3, Kluwer Academic Publishers.

Hernández Valdemar, E. J. & León, U. 2012. El paradigma de la programación visual, Fundación Arturo Rosenblueth.

Schmuller, J. 2009. UML en 24 hrs. Madrid España: Prencice Hall.

Simons, A.J.H. 1998. Object Discovery: A process for developing applications. Oxford : BCS.

Simons, A.J.H. 2006. Discovery history, url: <http://staffwww.dcs.shef.ac.uk/people/A.Simons/discovery/>, Último acceso Junio 23 2012.

Teufel, B., Schmid, S. & Teufel, T. 1995. Compiladores conceptos fundamentales. México: AddisonWesley Iberoamericana.

Notas biográficas:



Carlos Alberto Fernández y Fernández Egresado de la Facultad de Informática de la Universidad Veracruzana, con una Maestría en Ciencias de la Computación en la Fundación Arturo Rosenblueth. Recibió el grado de Doctor en Ciencias de la Computación en la Universidad de Sheffield, Inglaterra. Se encuentra adscrito al Instituto de Computación de la Universidad Tecnológica de la Mixteca. Ha sido coordinador de la Universidad Virtual y de la Maestría en Computación con especialidad en Sistemas Distribuidos. Trabaja dentro del área de Ingeniería de Software, particularmente en las líneas de modelado visual, métodos de desarrollo y especificación formal de software. Ha sido responsable del Cuerpo Académico de Ingeniería de Software en la UTM y miembro del Verification and Testing Research Group en la Universidad de Sheffield.



José Ángel Quintanar Morales Ingeniero en Computación egresado de la Universidad Tecnológica de la Mixteca. Realizó estudios de posgrado en la Maestría en Medios Interactivos en la misma universidad, de la cual esta realizando su tesis de grado. Actualmente se desempeña como líder de proyectos en KadaSoftware.



Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 2.5 México.

Impacto del aprendizaje basado en proyectos implementado en una empresa escolar de Base Tecnológica dedicada al desarrollo de Software

María Angélica Astorga Vargas
Universidad Autónoma de Baja California
angelicaastorga@uabc.edu.mx

Brenda Leticia Flores Rios
Universidad Autónoma de Baja California
brenda.flores@uabc.edu.mx

Jorge Eduardo Ibarra Esquer
Universidad Autónoma de Baja California
jorge.ibarra@uabc.edu.mx

Josefina Mariscal Camacho
Universidad Autónoma de Baja California
josefina.mariscal@uabc.edu.mx

Luis Enrique Vizcarra Corral
Universidad Autónoma de Baja California
luisvi@uabc.edu.mx

Resumen: En este trabajo se destacan los beneficios del Aprendizaje Basado en Proyectos (ABP) al aplicarlo en una empresa escolar de base tecnológica

en la que participan alumnos de un programa educativo de licenciatura. Los alumnos, antes de finalizar su formación profesional, desarrollan un conjunto de competencias asociadas a los roles más representativos en la Industria de software, que después desempeñarán en su ejercicio profesional. Se describe cada una de las fases del proceso de desarrollo mediante el cual se estableció una estrategia de ABP asociada al plan de estudios. Para determinar el impacto del ABP en el desarrollo de sus competencias, se aplicó un instrumento a alumnos y egresados que han participado como personal de la empresa escolar. Los resultados han sido positivos y reflejan la madurez de los alumnos cuando se ven expuestos a la participación en contextos de solución de problemas reales.

Palabras Clave: Aprendizaje Basado en Proyectos, Empresa escolar de base tecnológica.

Impact of project-based learning on a Software development school-based enterprise

Abstract: This paper highlights the benefits of Project-Based Learning (PBL) when applied in a school-based enterprise with participation of undergraduate students. By the time students finish their studies, they develop a set of competences associated to the most representative roles in the software industry, which are necessary for their professional practice. It describes each phase of the development process by which a PBL strategy was associated to the curriculum. To determine the impact of PBL in the development of their competences, an instrument was applied to current and former students who had participated as personnel of the school-based enterprise. The results have been positive and reflect the maturity of the students when exposed to a context of real-world problem solving.

Keywords: Project-Based Learning (PBL), School-based enterprise.

1. Introducción

El perfil de los egresados de las Instituciones de Educación Superior (IES) debe satisfacer las necesidades actuales de la industria (Gómez Álvarez et al., 2014) desde el ámbito internacional, nacional y local. Específicamente para la industria de software, es importante que un egresado posea las competencias relacionadas con conocimientos profesionales, manejo de herramientas de trabajo, implementación de métodos y técnicas y todas aquellas que se relacionan con la forma en que las personas colaboran, se comunican o manejan sus emociones (CIDAC, 2015).

Para que los egresados de los programas de licenciatura ofertados por las IES alcancen las competencias requeridas por el mercado laboral, no sólo es suficiente proponer contenidos temáticos que favorezcan su desarrollo; también es necesario que los profesores incorporen estrategias que favorezcan el proceso de enseñanza-aprendizaje, estimulando y motivando a los alumnos a participar de manera activa para que su aprendizaje sea significativo (Disla García, 2013). Diferentes experiencias y estrategias de enseñanza-aprendizaje han cambiado el papel que había desempeñado un alumno de receptor de conocimiento pasivo a ser activo, el cual tiene pensamiento crítico con los conocimientos adquiridos dentro y fuera del aula (Reitmeier, 2002). En estudios realizados se ha comprobado que la retención del conocimiento adquirido después de 24 horas en un alumno es del 5% para clases magistrales, 50% para la discusión en grupo, 75% para las experiencias prácticas y un 90% por enseñar a otros (Sousa, 1995; Rodríguez-Sandoval et al., 2010).

El aprendizaje basado en proyectos (ABP) es un método de enseñanza efectivo comparado con las estrategias de enseñanza cognitivas tradicionales, particularmente para el desarrollo de habilidades en la solución de problemas de la vida real (Willard y Duffrin, 2003). El ABP se caracteriza porque el grupo de profesores y alumnos realizan trabajo en grupo y de manera activa, planean, implementan y evalúan proyectos que tienen aplicación en el mundo

real, más allá del aula de clase (Disla García 2013; Galeana de la O, 2006; Martí et al., 2010). De igual manera, el ABP se ha convertido en una estrategia para la disciplina de Ingeniería de Software (Gómez Álvarez et al., 2014; CIDAC, 2015; Disla García, 2013), debido a que propone y desarrolla modelos innovadores de aprendizaje logrando potenciar las capacidades para el autoaprendizaje de los alumnos.

En el caso del programa educativo (PE) Licenciado en Sistemas Computacionales (LSC), que se oferta en la Universidad Autónoma de Baja California (UABC), esta correspondencia se estableció durante el proceso de modificación del mismo, el cual se inició en el año 2007. El primer paso del proceso de modificación implica la elaboración de una evaluación diagnóstica, que a su vez se compone de una evaluación interna y una externa. En la evaluación interna participan profesores, investigadores y alumnos; mientras que en la externa, se consulta la opinión de empleadores, asociaciones de profesionistas, organismos de acreditación, otras instituciones educativas y egresados (UABC, 2010). La modificación a un PE busca como objetivo fundamental modernizar el contenido curricular de tal forma que el alumno pueda adquirir conocimientos, generar habilidades y destrezas, desarrollar y afianzar actitudes y valores que le permitan practicar la profesión de forma competente (Casallas, 2015).

Para el logro de las competencias de algunas unidades de aprendizaje asociadas al PE de LSC, se creó una empresa escolar de base tecnológica a la cual se incorporó el enfoque de ABP, por ser un modelo pedagógico favorable para la enseñanza en programas de ingeniería, en el cual los alumnos trabajan en grupos para solucionar problemas o proponer proyectos de desarrollo de software. De esta forma, un alumno de LSC que participa en esta empresa escolar adquiere y desarrolla competencias relacionadas a la Ingeniería de software (IS), brindándole una experiencia práctica para incorporarse a un ambiente profesional.

Este documento se estructura de la siguiente manera: La sección 2 presenta las características principales y el proceso de desarrollo del enfoque ABPy trabajo relacionado de su aplicación en entornos universitarios en México. En la sección tres se resalta la importancia de las empresas escolares de base tecnológica. Después, la sección cuatro presenta los resultados de la aplicación de cada una de las fases del proceso de desarrollo de ABP en una empresa escolar. En la sección cinco, se analizan los resultados de la contribución práctica de utilizar el ABP para la generación de egresados con las competencias requeridas por la industria de software de la localidad. Por último, se discuten las conclusiones y trabajos futuros que se derivan de esta experiencia.

2. Aprendizaje Basado en Proyectos (ABP)

Una de las metodologías activas que más aceptación está teniendo es el Aprendizaje Basado en Proyectos (ABP), la cual presenta una adaptación estratégica de la enseñanza de las ciencias e ingeniería (Rodríguez-Sandoval et al., 2010). ABP establece estrategias en las que los estudiantes planifican, implementan y evalúan proyectos aplicados a la realidad en las respectivas áreas de conocimiento (Sandía et al., 2010). Guitart et al., (2006) distingue tres elementos en el enfoque ABP: 1) Conceptual – relacionado con el incremento del conocimiento teórico del saber de un área; 2) Procedimental – ampliando el conocimiento práctico y metodológico del saber de un área; e 3) Integrador – se propicia el incremento del conocimiento relacionado a las destrezas, aptitudes y actitudes propias del ejercicio de una profesión.

Por medio del enfoque ABP, el alumno aprende a aprender (Murciencia, 2008), se le desarrolla un aprendizaje autónomo, se le fomenta un trabajo colectivo orientado tanto al proceso como al producto (Disla García, 2013) y se forma con un carácter interdisciplinario debido a que combina distintas áreas del

conocimiento y especialidades (Tippelt y Lindemann, 2001). Es por esto, que los tres ejes principales del ABP incluyen las relaciones, la comunicación y el aprendizaje centrado en el alumno (Galeana de la O., 2006). A medida que los alumnos y profesores interactúan y socializan fomentan su confianza, esfuerzo conjunto y comunicación. El rol del profesor es coordinar y supervisar un trabajo de mediana envergadura (proyecto) realizado por los alumnos. Después de una introducción inicial de los conceptos teóricos más importantes, se promueve que el alumno realice un proyecto sobre temas reales. El proyecto consiste en una labor, fundamentalmente práctica, con la que se tratan tanto las competencias propias de la asignatura como las de carácter transversal (Murciencia, 2008).

Según la Organización para la Cooperación y Desarrollo Económico (OCDE, 2002) una competencia es la capacidad para responder a las exigencias de un contexto o trabajo concreto, desde un nivel individual hasta un nivel social. Cada competencia reposa sobre una combinación de habilidades prácticas y cognitivas interrelacionadas, conocimientos, valores y otros elementos sociales y de comportamiento que pueden ser movilizados conjuntamente para actuar de manera eficaz. De igual manera, el proyecto Tuning Educational Structures define el término competencia como una combinación dinámica de atributos, en relación a procedimientos, habilidades, actitudes y responsabilidades, que describen los encargados del aprendizaje de un programa educativo, o lo que los alumnos son capaces de demostrar al final de un proceso formativo (Beneitone, 2007).

A partir de las ventajas de trabajar con el ABP identificadas en distintos trabajos (Martí et al., 2010; Disla García, 2013), la tabla 1 presenta la asociación con las áreas de competencias genéricas (Pavié, 2011) que se ven favorecidas por este enfoque.

Ventaja	Descripción	Área de competencia
Desarrollar habilidades y competencias	Se aumenta el nivel de conocimientos y habilidades de los alumnos en una disciplina o en un área específica permitiendo la colaboración, planeación de proyectos, comunicación y toma de decisiones (Blank, 1997). Se alcanza un elevado nivel de habilidad en un área específica. Se plantea y emprende una tarea desafiante que requiera de un esfuerzo sostenido durante un periodo de tiempo específico.	Cognitiva
Desarrollar habilidades de investigación	El proyecto mejora considerablemente las aptitudes de los alumnos para la investigación de temas que complementen el conocimiento actual. Se promueve la autonomía en el proceso de aprendizaje, para que los alumnos tomen sus propias decisiones y superen dificultades en cada paso del proyecto.	Autoaprendizaje y autoconocimiento
Desarrollar habilidades de colaboración para generar conocimiento (Disla García, 2013)	El aprendizaje colaborativo permite compartir ideas entre los alumnos, expresar sus propias opiniones y negociar soluciones, las cuales son habilidades necesarias para la exteriorización de su conocimiento en sus futuros puestos de trabajo (Reyes, R., 1998).	Social

Incrementar las capacidades de análisis y de síntesis	Especialmente cuando el proyecto está orientado a que los estudiantes conozcan más profundamente una realidad, simplificar su descripción, descubrir relaciones entre los elementos de la realidad en estudio y construir nuevos conocimientos a partir de otros que ya se poseían.	Resolución de problemas
Capacidad para comunicarme con los demás de forma eficaz	Compartir ideas entre los miembros del equipo de trabajo, así como notificar información relevante para el logro de las actividades entre mi equipo y fuera de él.	Social
Aprendizaje sobre cómo evaluar y coevaluar	Los alumnos incrementan esta habilidad y se responsabilizan con su propio trabajo y desempeño a la vez que evalúan el trabajo y desempeño de sus compañeros.	Resolución de problemas y social
Establecer su compromiso en un proyecto	Los alumnos se comprometen de forma activa y adecuadamente con la realización del proyecto, por lo que se encuentran internamente motivados.	Motivación hacia el trabajo

Tabla 1.

Ventajas del uso del ABP y su asociación con áreas de competencias generales. Elaboración propia

Para que los resultados de trabajo sean exitosos, se requiere de un diseño instruccional bien definido, la especificación de roles y fundamentos de diseño de proyectos. El diseño de proyectos incluye la etapa de planeación y análisis del proyecto, formulación del objetivo definido, limitación del problema o situación a resolver y la identificación de los perfiles de los actores involucrados (Galeana de la O., 2006).

2.1 Proceso de desarrollo para el enfoque ABP

En la preparación del diseño del proyecto es necesario y conveniente ajustarse a criterios y pasos metodológicos que sean capaces de adaptarse y responder a la complejidad y transformaciones de la realidad. Para el diseño de proyectos bajo el enfoque de ABP, Galeana de la O (Galeana de la O., 2006) propone seis etapas como proceso de desarrollo y tres entidades relacionadas. La figura 1 muestra cómo el enfoque ABP se hace evidente en las etapas de: 1) Planeación, 2) Análisis, 3) Diseño, 4) Construcción, 5) Implantación y 6) Mantenimiento; mientras que las entidades se refieren a los usuarios, la información y la tecnología.

Diseño del Proyecto			Proceso de Desarrollo	Productos Generados
Usuario	Información	Tecnología	Planeación Análisis Diseño Construcción Implantación Mantenimiento	Planteamiento de las Estrategias, Plan de Trabajo

Figura 1.

Etapas y Entidades en el Diseño de Proyectos. Basado en (Disla García, 2013)

En este diseño, el profesor debe fomentar en los alumnos el desarrollo de sus actividades y el planteamiento del proyecto en relación a los usuarios y riesgos implicados. En cada una de las etapas se obtienen entregables. Por ejemplo, como resultado de la planeación y el análisis se generan el planteamiento de las estrategias y el plan de trabajo. Ambos productos son esenciales para obtener resultados favorables en las demás fases. Para medir el avance se establece el grado en que los miembros de los equipos de trabajo están interactuando, el incremento y visibilidad de la solución, y el aprendizaje generado (Gómez Álvarez et al., 2014).

2.2 Experiencias con el uso de ABP en IES Mexicanas

Existen reportes de experiencias de aplicación del enfoque ABP en IES mexicanas donde se imparten cursos de Ingeniería de Software (IS) orientados tanto a alumnos de nivel de posgrado como de licenciatura. Algunos de ellos se describen a continuación:

En el periodo comprendido de 2011 a 2012, Arellano Pimentel et al., (2013) utilizaron la metodología de software incremental con el enfoque ABP para un curso de compiladores para el desarrollo de un generador L-Systems en la carrera de Ingeniero en Computación de la Universidad del Istmo, con un grupo reducido de 4 estudiantes. Al final del curso, el 100% de los alumnos lograron terminar el generador L-Systems y expresaron estar motivados en realizar mejoras o nuevos incrementos. Por otro lado, el 25% presentó fallas al momento de ejecutar su aplicación.

Anaya (2012) propone una visión de la enseñanza en la IS como apoyo a elevar la madurez de las empresas, la cual incluye una integración de factores técnicos, gerenciales y organizacionales. Su trabajo se centra en que los profesores deben de tener una vista unificada acerca del cuerpo de conocimiento que soporte esta disciplina y las IES deben de 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 la IS.

En la Universidad de Querétaro (UAQ) se están generando prácticas de tipo ABP para aprovechar los recursos de modernización educativa donde alumnos, de licenciatura y maestría, y profesores de diversas disciplinas participan en la elaboración de proyectos que resuelven necesidades sociales. Los alumnos se enfrentan a problemas complejos que la realidad presenta y que una sola disciplina no puede resolver, por lo que propician la investigación interdisciplinaria entre las licenciaturas de Ingeniería de software, Contabilidad, Administración, Diseño Gráfico y Maestría en Diseño. Los miembros de cada

equipo se encuentran geográficamente distribuidos (Amealco de Bonfil, San Juan del Rio, Juriquilla y Centro Universitario Cerro de las Campanas) por lo que se apoyan de un software para la administración de proyectos en línea para posibilitar el desarrollo del proyecto final (Sánchez Martínez et al., 2015).

La investigación realizada por García y Pacheco (2012) muestra que la necesidad de educar a un Ingeniero de software reside en la importancia de enseñarle a diseñar un sistema, crear y probar componentes de código, supervisar a los programadores, evaluadores de calidad, probadores y especialistas de gestión de la configuración. Los autores enfatizan la idea de que los alumnos desarrollen experiencias y habilidades prácticas como las requeridas por la industria de software o un entorno simulado, por lo que en la Universidad Tecnológica de la Mixteca combinan la metodología de trabajo en equipo para gestionar proyectos de software (TSPi) y el enfoque ABP.

Algunas IES mexicanas están promoviendo la mejora de capacidades de los alumnos a través de la teoría y la práctica enfocándose en las necesidades de la industria, generando productos en un esquema de fábrica real con clientes y demandas reales (García et al., 2010). Lo relevante, es que las IES reconozcan la necesidad de introducir estrategias pedagógicas alternativas (como el método ABP) dentro de escenarios simulados (fábricas de software, parques tecnológicos, empresas escolares) con características de la industria real que le permitan a los alumnos desarrollar e incrementar conocimientos y habilidades (de cooperación, colaboración y de investigación).

3. Empresas escolares de Base Tecnológica

La empresa escolar es una empresa que es propiedad de una escuela pero que es dirigida y operada por sus alumnos. Son una plataforma para que los alumnos aprendan habilidades empresariales y adquieran experiencia práctica en la gestión de un negocio (SEC, 2015). Por otro lado, la creación de empresas de base tecnológica es uno de los pilares más sólidos sobre los que se puede fundamentar el crecimiento económico y la competitividad de cualquier economía (Díaz Sánchez et al., 2013). Los Estados miembros de la Unión Europea mantienen el firme compromiso de hacer de Europa una economía basada en el conocimiento, donde las nuevas empresas de base tecnológica sean la base de su economía (Comisión Europea, 2012).

El proceso de crear o poner en marcha una empresa de base tecnológica es complejo debido a su conexión entre la investigación básica, los sectores o nichos de mercado a los que se dirigen, y a que éstos no han alcanzado su total madurez y la aplicación concreta de la tecnología. Díaz (2010) aborda el análisis de las empresas de base tecnológica en comparación con otras empresas innovadoras para esclarecer las diferencias existentes entre ambas y los mejores resultados obtenidos por las primeras. La capacidad de explotar oportunidades tecnológicas mediante la creación de nuevas empresas es aún un campo de investigación (Busenitz et al., 2003).

El enseñar un curso de IS en un programa de licenciatura, es una tarea retadora. Algunas estrategias van desde planteamientos clásicos hasta visiones dinámicas en donde el alumno se enfrenta a desarrollar proyectos con características similares a los proyectos del mundo real (Casallas, 2015). Los alumnos se consideran representantes válidos de profesionales en la industria (Höst et al., 2000; Runeson, 2003; Carver et al., 2003), quienes en ocasiones experimentan su aprendizaje en escenarios como empresas escolares de base

tecnológica o spin-offs universitarias, las cuales son un subgrupo de las empresas de base tecnológica (Smilor et al, 1990; Steffensen et al., 2000).

El éxito en los países que han surgido como potencias en el desarrollo de proyectos de software ha sido al énfasis de la educación de calidad. Los países consideran su capital humano como alumnos con alto potencial, profesionistas relacionados a la industria de software que poseen habilidades específicas, así como la capacidad y competencias para identificar y resolver problemas complejos reales (Colomo et al., 2013).

Por lo anterior, en una empresa escolar de base tecnológica, se espera que los responsables de proyectos de software creen, usen y transmitan conocimiento, competencias y habilidades para desempeñar sus actividades. Para lograrlo, se requieren mecanismos o estrategias que no sólo les ayuden en la realización de proyectos de desarrollo de software, sino que también permitan incrementar sistemáticamente el conocimiento, así como reducir el desaprovechamiento y el riesgo de pérdida del mismo.

En la siguiente sección, se describen las experiencias en el uso del enfoque ABP en el PE de LSC de la Facultad de Ingeniería UABC campus Mexicali, bajo su modelo de empresa escolar.

4. Contexto de estudio

La experiencia que se presenta es de un carácter descriptivo analítico; su objetivo es el de contribuir con un esfuerzo para conseguir las competencias necesarias para enfrentar problemas de la industria de software (Gómez Álvarez et al., 2014) y lograr una mejor práctica de la disciplina de la IS (Juárez-Ramírez et al., 2013).

La descripción del programa y plan de estudios del PE de LSC distribuye 55 unidades de aprendizaje en tres etapas de formación: Básica, Disciplinaria y

Terminal (UABC, 2009). La etapa básica es de carácter interdisciplinario, en la que se incluyen 15 unidades de aprendizaje con una orientación eminentemente formativa, mediante la cual se adquieren conocimientos en las diferentes disciplinas elementales para las áreas informáticas, integrando así unidades de aprendizaje contextualizadoras, metodológicas y cuantitativas, esenciales para la formación del alumno. En la etapa disciplinaria el alumno tiene la oportunidad de conocer, profundizar y enriquecerse de los conocimientos teórico-metodológicos y técnicos de la profesión, orientados a un aprendizaje genérico del ejercicio profesional. Comprende la mayor parte de los contenidos del perfil del programa y el nivel de conocimiento es más complejo; se compone de 28 unidades de aprendizaje. Por último, la etapa terminal se establece al final del programa reforzando los conocimientos teóricos específicos; en esta etapa se incrementan los trabajos prácticos y se empieza a desarrollar la participación del alumno en el campo ocupacional, donde podrá explorar las distintas orientaciones a través de la integración y aplicación de los conocimientos adquiridos, para enriquecerse en áreas afines y poder distinguir los aspectos relevantes de las técnicas y procedimientos que en este perfil profesional se requieren para la solución de problemas o en la generación de alternativas. Se integra de 12 unidades de aprendizaje (Proyecto reestructuración, 2009).

La descripción del programa también incluye las competencias generales, específicas y particulares del PE, así como las formas que se han establecido para la evaluación, seguimiento y retroalimentación durante el proceso de su implementación para un óptimo resultado. Se incluyen también los programas de las unidades de aprendizaje, en donde se incorporan las competencias y evidencias de desempeño de cada una de las etapas de formación.

A continuación, se describe cómo se implementó cada una de las fases del proceso de desarrollo del ABP (Fig. 1) dentro de un entorno de empresa escolar de base tecnológica y cómo se relacionan los elementos del Diseño de proyecto: Usuario, Información y Tecnología.

4.1 Planeación

En el 2005, la industria de software local y la Secretaría de Economía (SE) en el estado de Baja California expusieron la necesidad primordial de que el perfil de egreso de los estudiantes de Tecnologías de Información cumpliera con las competencias requeridas en la adopción de modelos de mejora de procesos de software, de manera particular promoviendo el Modelo de Procesos para la Industria del Software en México (MoProSoft), establecido como la norma NMX-I-059-NYCE-2011 (SE, 2014). Como respuesta, se planteó un proyecto cuyo objetivo principal fue aproximar a los alumnos a que vivan la experiencia de trabajar en un ambiente real-laboral dentro del aula en apego a la mejores prácticas de Gestión e IS propuestas por la norma NMX-I-059 como modelo de referencia de procesos de software y con el soporte de metodologías específicas para el desarrollo y mantenimiento de software como el Proceso Unificado de Rational (RUP por las siglas en inglés).

Para generar un ambiente de trabajo real, en el año 2007 en la UABC se creó una empresa escolar de base tecnológica dedicada al desarrollo de software, denominada AvanTI (por su acrónimo de Avanzando en las Tecnologías de la Información) en la que se tienen implementados los nueve procesos requeridos por la norma (NMX-I-059/02, 2011). El propósito es brindar a los alumnos la oportunidad para que desarrollen proyectos tanto de gestión como de IS (NYCE, 2011). Para los proyectos de IS se tenía que contar con un cliente externo y/o interno y dependiendo de sus necesidades definir el tipo de proyecto de software a desarrollar. Otros aspectos que se tomaron en cuenta fueron el identificar las unidades de aprendizaje que deberían ser cursadas por parte de los alumnos antes de su ingreso a AvanTI y cuáles unidades de aprendizaje deberían asociarse a AvanTI de tal manera que se impartieran bajo el enfoque ABP. Así mismo, el involucrar al grupo de profesores que guiaran a la empresa escolar como parte del Consejo Directivo. Para la implantación de la norma NMX-I-059 (MoProSoft), se definió una estructura

organizacional dividida en tres categorías y que representan la Alta Dirección (DIR), Gestión (GES) y Operación (OPE). Cada categoría comprende uno o más departamentos en los que se han implementado los nueve procesos (Astorga Vargas et al., 2010).

Para que los alumnos desarrollaran proyectos de software que resolvieran problemas reales, se estableció como estrategia la vinculación con instituciones públicas a través del programa de Servicio Social Profesional (SSP) asociado a la currícula; es decir, los alumnos al mismo tiempo de cursar las unidades de aprendizaje realizan su SSP, que los motiven a su vez a retribuir a la sociedad con los conocimientos adquiridos previo a su egreso profesional.

4.2 Análisis

Se analizaron las tres etapas de formación en relación a los contenidos temáticos requeridos para los dos tipos de proyectos de gestión y de IS. Considerando el propósito de formación del perfil y su campo ocupacional, estos contenidos temáticos se imparten en las etapas disciplinaria y terminal, en donde se logran las competencias específicas para la Gestión con las unidades de aprendizaje: Administración, Administración de Personal y Reingeniería de Procesos; y las competencias específicas para la IS con las unidades de aprendizaje: Programación Orientada a Objetos, Análisis y Diseño de Software, Base de Datos y Desarrollo de Software y Administración de Proyectos de Software.

El escenario ideal para la aplicación del APB son asignaturas en las que los alumnos tengan tiempo de realizar las tareas requeridas con la profundidad necesaria (Alcober et al., 2003). Tomando en cuenta este criterio y que la cobertura de los contenidos temáticos para alcanzar el propósito del modelo de referencia son amplias, se definió que el ingreso a Avanti se ubicaría en las unidades de aprendizaje en las que se imparte el Desarrollo y Mantenimiento de Software (DMS) y la Administración de Proyectos de Software (APE),

mismas que están diseñadas para que se brinde continuidad a un mismo proyecto de software. Por lo tanto, las unidades de aprendizaje mencionadas anteriormente, se deben cursar previo al ingreso de AvanTI.

4.3 Diseño

En semejanza con una empresa real, en el diseño instruccional se consideró de suma importancia tomar en cuenta los conocimientos previos y las motivaciones de los alumnos (Belloch, 2012). Existen algunas consideraciones de aprendizaje donde la motivación de los estudiantes depende del grado de influencia de su participación: más motivación cuantas más decisiones puedan tomar los alumnos (Kolmos, 2004). En consecuencia, ellos mismos seleccionan el(los) rol(es) que están interesados en desempeñar de acuerdo a la estructura organizacional, lo cual, a su vez, propicia la construcción de nuevos conocimientos, experiencias y actitudes (Belloch, 2012). Para que esto fuera posible, los contenidos temáticos se dispusieron de tal manera que el profesor, al ir impartiendo dichos contenidos, estableciera los espacios para:

1. Seleccionar el personal con base en los conocimientos requeridos por cada rol a través de entrevistas de trabajo, donde su aceptación depende de sus conocimientos y el resultado de su examen de selección;
2. Asignar el personal a los proyectos de gestión y de IS. Al hacerlo se promueve el desarrollo de actividades y conocimientos coherentes y que tengan sentido para el alumno, fundamentalmente porque desarrollan competencias necesarias para su futuro personal y/o profesional (Belloch, 2012);
3. Establecer los momentos de revisión de avances con los equipos de trabajo y de reflexión a partir de los resultados presentados. Así, se fomentan aspectos como el intercambio de ideas, la generación de nuevo conocimiento a partir de la investigación, la importancia de su compromiso, responsabilidad individual, de equipo y colaborativa con los demás equipos de trabajo;
4. Establecer una forma de evaluación similar a la de una empresa real, donde se observe que el proceso de Gestión de Recursos (GR) requiere la Evaluación del Desempeño del Personal. Se evalúan indicadores de conocimiento del puesto, productividad, habilidades de comunicación, equipo de trabajo y habilidades de dirección en aquellos puestos que tienen

personal a cargo. En estas evaluaciones participan los miembros de los equipos de trabajo, haciendo una evaluación por pares a manera de coevaluación.

Para el diseño del proyecto también se requiere definir las tres entidades: usuario, información y tecnología. Los usuarios son los clientes internos y/o externos quienes solicitan el desarrollo de un producto de software; la información está asociada a la unidad de negocio del cliente y a su vez a los temas de Gestión e IS que son impartidos en las unidades de aprendizaje y sobre los cuales los alumnos deben investigar para ampliar sus conocimientos en búsqueda de la solución del problema planteado; la tecnología está asociada a las herramientas de IS asistida por computadora (CASE por las siglas en inglés Computer Aided Software Engineering) que permiten automatizar los procesos propios del ciclo de vida de desarrollo y de soporte para administrar el proyecto; así como la tecnología utilizada para desarrollar las aplicaciones como el lenguaje de programación, administradores de base de datos, servidores Web, entre otros.

4.4 Construcción

En cada periodo de cambio de personal, el cual se sucede cada dos semestres. Se lleva a cabo una dinámica denominada armando MoProSoft, en la que por cada uno de los 9 procesos se conforma un equipo que presenta de manera completa el proceso que le fue asignado; principalmente se describen las actividades de cada rol y cómo interactúan con los demás roles, según su participación como Autoridad, Responsable o Involucrado. Se realizan entrevistas de trabajo donde su aceptación depende de sus conocimientos y el resultado de su examen de selección. La asignación de roles propicia que se aumente el nivel de conocimientos y habilidades de los alumnos en el área específica que les es asignada.

Una vez que es asignado el personal según su rol, entonces se asignan los proyectos. Los proyectos de Gestión están relacionados con los procesos de

Gestión de Negocios (GN) para consecución de la planeación estratégica y en este participan el Director General y los Gerentes de los procesos de gestión; Gestión de Procesos (GPR) para la definición e implantación de los procesos de MoProSoft en el que participan el Gerente de Procesos y los Responsables de los demás procesos; GR para proveer Bienes, Servicios e infraestructura (BSI), Recursos Humanos y Ambiente de Trabajo (RHAT); así como crear y mantener la Base de Conocimiento (BC) de la organización (CO). Los roles son el Gerente de Recursos, el Responsable de BSI, el Responsable de RHAT y el Responsable de CO; Gestión de Proyectos (GPY) para mantener una cartera de proyectos de software que contribuya a los objetivos de la organización, para ello participa el Gerente de Proyectos, el Director General y el Responsable de APE.

Para los proyectos de software, se definió que los proyectos se dividieran por componentes los cuales son asignados a los equipos, en AvanTI nombrados como cuadrillas de trabajo. Cada cuadrilla está integrada por los roles: Analistas de Negocios, Analistas de Sistemas, Diseñadores de Arquitectura, Diseñadores de Interfaces de Usuario, Programadores, Probadores y Encargados de Manuales y Capacitación. Según la complejidad de los componentes se asigna un determinado número de roles, por ejemplo, puede haber más de un programador. Para la Administración del Proyecto de Software se integra un equipo de trabajo conformado por el Responsable de GPY, Responsable de APE y por el Responsable de DMS. De esta manera se incorporan diferentes perfiles que comparten sus áreas específicas de conocimiento y que a su vez también fortalecen el trabajo colaborativo intrínseco en el ABP.

Se identifica el nuevo conocimiento y habilidades que deben adquirir y cómo lo pueden obtener. Esto propicia el aprendizaje autónomo al investigar los temas que complementen su conocimiento actual. Para cada uno de los proyectos de gestión y de ingeniería el Consejo Directivo apoya a los equipos con la definición de lo que cada proyecto debe lograr. Los equipos establecen un

primer plan de trabajo para llevar a cabo las actividades y generar los productos de trabajo derivados del proyecto. Así mismo, plantean cómo solucionar el problema, tomando en consideración la metodología y las herramientas necesarias.

4.5 Implantación

Dentro de los equipos de trabajo, el jefe inmediato es quién vigila que se vaya cumpliendo con lo planificado; se presentan los avances al Consejo Directivo y/o cliente de acuerdo al tipo de proyecto y plan de trabajo establecido. El Consejo Directivo va guiando a los alumnos para que realicen sus actividades y obtengan los entregables a través de reuniones establecidas en los planes de trabajo.

A partir de las reflexiones generadas en las revisiones, se decide continuar o replanificar las actividades para llegar a la solución esperada. En estas revisiones se evalúa el conocimiento adquirido como resultado de su participación en la solución del proyecto, sobre lo que funcionó bien y lo que debe mejorar, la manera en que se responsabilizan de los resultados, establecen comunicación entre ellos, demuestran que saben trabajar en equipo, tomar decisiones y negociar las soluciones, la motivación al presentar los resultados, entre otros aspectos. Además, se identifican las necesidades derivadas del nuevo conocimiento y/o habilidades, lo que permite valorar estrategias para la actualización y capacitación; por ejemplo la realización de cursos adicionales. Este ciclo de revisiones se repite hasta concluir el proyecto de acuerdo al alcance definido.

Se realizan dos coevaluaciones durante el semestre, en la que el jefe inmediato evalúa a su personal y retroalimenta los resultados hasta llegar a una aceptación por parte del evaluado. La evaluación realizada por los equipos de trabajo es equivalente al 80% de la calificación del curso, el otro 20% es

definido por el profesor con base en las participaciones en las sesiones de clase, tareas y exámenes.

4.6 Mantenimiento

La participación en proyectos de software para un cliente real, implica que el área de negocio y la tecnología requerida puedan ser diferentes a las que se imparte en el PE de LSC. Para ello, se han incorporado cursos adicionales y/o capacitación específica que aproxime al alumno a dicho conocimiento para que a partir de éste se desarrolle un aprendizaje autónomo y se le fomente un trabajo colectivo (Disla García, 2013). La complejidad de los proyectos de software puede extenderse a más de dos semestres, por lo que se delimita el alcance de cada una de las liberaciones permitiendo que los equipos de trabajo concluyan con lo planeado y logren realizar una presentación final. De esta manera, al realizarse un cambio de personal se puede tener una continuidad de los proyectos definiendo el alcance a partir de lo que ya se desarrolló.

En un principio, el interés de la adopción de la norma NMX-I-059 (MoProSoft) se debió a sus principales características de ser fácil de entender y de aplicar. No obstante, en un periodo de diez años algunas de las empresas del estado que iniciaron un proyecto de mejora de procesos de software con dicha norma y/o sólo el modelo MoProSoft, están escalando al modelo CMMI® v1.3. Por tal motivo, se refleja como una nueva necesidad para que el PE de LSC incorpore a sus unidades de aprendizaje otros modelos internacionales como CMMI, ISO/IEC 29110 perfil básico e intermedio (ISO 29110, 2011).

La implementación del enfoque ABP, por medio de las seis fases presentadas y los elementos del diseño, ha logrado desarrollar en los alumnos un conjunto de competencias asociadas a los roles de la industria de software. Su impacto se puede medir en el grado en que las competencias fueron adquiridas y la influencia que han tenido para elegir el área específica en la práctica de su ejercicio profesional dentro del mercado laboral. En la siguiente sección, se

presentan los resultados y la aportación que ha tenido la empresa escolar AvanTlen la contribución de la formación de profesionales de la industria de software.

5. Resultados y Aportación

Existe un conjunto de destrezas que resulta fundamental en la buena práctica de la IS y que tiene que ver con las capacidades descritas por los empleadores, tales como la capacidad de la buena comunicación, negociación, hábitos efectivos de trabajo, valores, habilidades de trabajo en equipo, comportamiento interpersonal, motivación, iniciativa, alta responsabilidad y tolerancia, capacidad de aprender y aprender por cuenta propia para mantenerse vigente, entre otros (CERTVER, 2015). Durante un periodo de siete años, estas capacidades se han desarrollado y puesto en práctica a 180 alumnos en la empresa escolar AvanTI a través de la realización de proyectos orientados al área de la salud y académica con las realidades del entorno laboral de su PE. Además, los alumnos han aprendido a interpretar modelos de calidad y procesos de software apropiados para pequeñas organizaciones de desarrollo de software (Astorga et al., 2010).

Ante estos referentes, resulta conveniente identificar y conocer si el entorno de una empresa escolar implementando el enfoque ABP, ha apoyado a que los alumnos participantes incrementen su nivel o grado de áreas de competencias. A continuación, se presenta como parte de la metodología la descripción del instrumento aplicado y la caracterización de los sujetos que respondieron a este; los resultados obtenidos y un panorama general de la aportación práctica de los egresados en su ambiente profesional.

5.1 Metodología

Se diseñó un instrumento de captura de información, tipo encuesta, en dos secciones. La primera sección orientada a medir el grado en el que los alumnos poseen competencias en las áreas presentadas en la tabla 1, antes de su ingreso a AvanTI (etapa de inicio) y el grado en el que fueron desarrolladas al finalizar el proyecto (etapa final). La escala utilizada es de tipo Likert con cinco opciones de respuesta, las cuales son: 1- Muy Bajo, 2-Bajo, 3-Medio, 4-Alto y 5-Muy Alto. El número total de roles fue de trece y cada alumno respondió el grado de dominio de los siete factores (ítems) solicitados, en las etapas de inicio y final, por cada uno de los roles desempeñados por él mismo (figura 4); es decir, el alumno en algunos casos desempeñó más de un rol al mismo tiempo. Estos factores corresponden a las ventajas del ABP que se mencionan en la tabla 1.

2f. Habilidades y competencias INICIAL
 Selecciona aquellos Roles que desempeñaste y el Nivel de conocimientos y habilidades en el área específica (rol) que me fue asignado.

	Muy bajo	Bajo	Medio	Alto	Muy alto
Analista de Sistemas	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

2f. Habilidades y competencias FINAL
 Selecciona aquellos Roles que desempeñaste y el Nivel de conocimientos y habilidades en el área específica (rol) que me fue asignado.

	Muy bajo	Bajo	Medio	Alto	Muy alto
Analista de Sistemas	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Diseñador de interfaces de usuario	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Diseñador de arquitectura	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Programador	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Probador	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Encargado de manuales	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Responsable de APE, Responsable de DMS y/o Líder de proyecto	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Director general	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Gerente de proyectos	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Gerente de Procesos	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Gerente de Recursos Humanos	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Responsable de la Base de Conocimiento	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Responsable de RHAT y/o BSI	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Figura 2.

Encuesta para el factor Desarrollo de habilidades y competencias para las etapas de Inicio y Final

Para determinar el impacto del ABP sobre los participantes en la empresa escolar AvanTI, se aplicó el instrumento a una muestra representativa de alumnos del PE de LSC, tanto que se encuentran actualmente en la empresa como egresados. De la generación 2014-2, el 50% de ellos colaboran activamente como personal de AvanTI y el otro 50% corresponde a alumnos que egresaron de las generaciones 2010-2, 2011-2, 2012-2 y 2013-2.

5.2 Análisis de resultados

Los roles adquieren una responsabilidad por la realización de un conjunto de actividades específicas para un proceso y del cumplimiento de sus objetivos. Además, pueden ser asumidos por una o más personas de la organización (Flores Rios et al., 2014). En el 2014, se presentó el estudio de salarios de los 20 roles más representativos de la industria de software mexicana (Galvan, 2014). En la tabla 2 se muestran los roles con mayor participación y su porcentaje de representatividad, los cuales son: Programador (PR), Analista de requerimientos (AN), Diseñador de arquitectura (DI) y Administrador de proyectos (RAPE) y sus actividades principales (NMX-I-059/03, 2011; Serna, 2013). De acuerdo a las actividades que desempeñan los roles, el desarrollo de estos factores (competencias y habilidades) les permiten ir formándose de una manera adecuada antes de adentrarse plenamente en la práctica profesional (Lethbridge, 2000).

No.	Rol	Actividades Principales	Porcentaje
1	Programador (PR)	Interpretar una especificación de diseño para implementar una solución, haciendo uso de una arquitectura de referencia establecida y utilizando adecuadamente las librerías de un lenguaje o las funcionalidades provistas en un marco de trabajo.	29.8%
2	Analista de requerimientos (AN)	Descubrir, analizar y documentar el propósito del producto, mediante la identificación de las necesidades de las partes interesadas.	10.2%
3	Diseñador de Arquitectura (DI)	Realizar el proceso de definición de la arquitectura, los componentes, las interfaces y otras características de los productos. Describe cómo se descompone el software y cómo se organizan en componentes. Se refiere a la identificación de los principales componentes hardware y software de un producto, que proporcionan las características y atributos de calidad del mismo.	9.5%
4	Administrador de proyectos (RAPE)	Planificar, supervisar y, evaluar y controlar los proyectos de software. Define el costo, calendario, proceso de DMS, equipo de trabajo, adquisiciones y capacitaciones, riesgos y protocolos de entrega.	9.1%

Tabla 2.

Roles relacionados a la industria de software (SG, 2014) y sus actividades principales (NMX-I-059/03, 2011; Serna, 2013)

A continuación, se describen los resultados obtenidos en los roles más representativos de la industria del software (tabla 2) en relación a los factores de mayor impacto a las actividades que debe desempeñar. En la figura 5 se presenta el factor 1) Desarrollo de sus habilidades y competencias, con el rol de AN desempeñado por 21 alumnos, quienes en la etapa de inicio el 38% respondieron entre Muy Bajo y Bajo, 38% Medio y 24% se sitúan entre Alto y Muy Alto. Mientras que en su etapa final, no hay quien considere el grado Muy Bajo o Bajo, siendo el 19% el que respondió como Medio, 57% Alto y 24% Muy Alto.

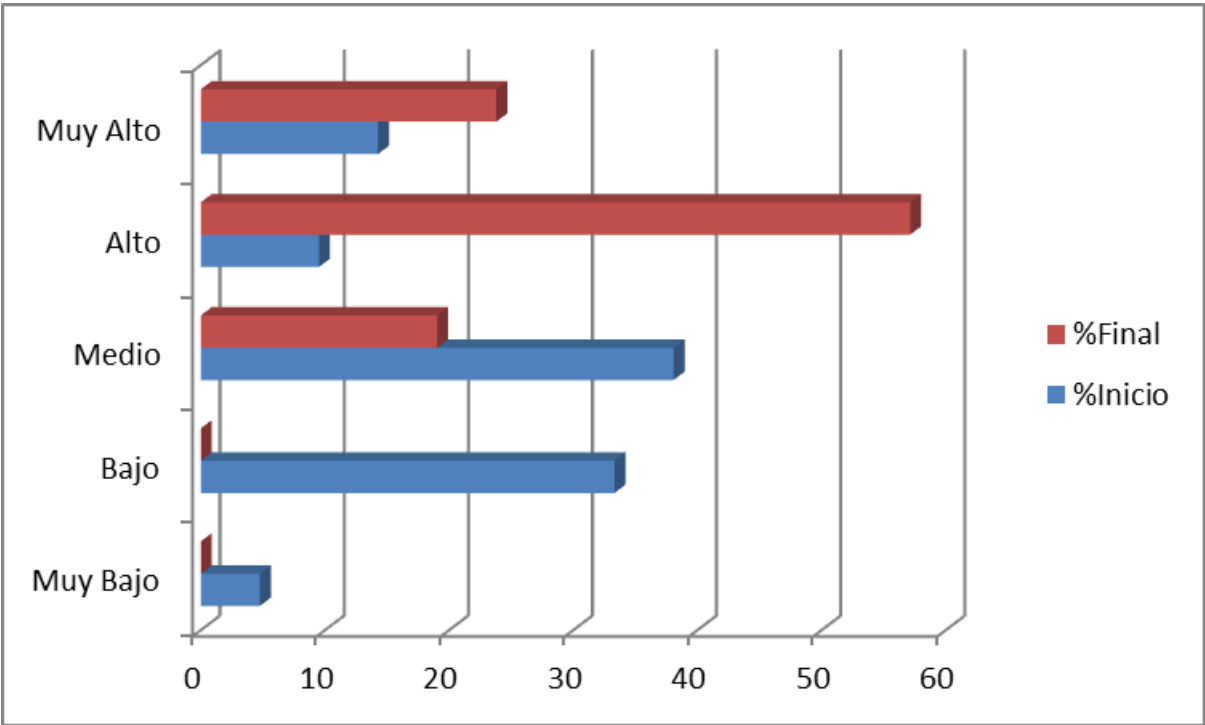


Figura 3.

Desarrollo de habilidades y competencias con el rol Analistas de Sistemas

Se observa que con el rol DI, de los 13 alumnos que lo desempeñaron respondieron que en la etapa de inicio el 38% se ubicaron en 2-Bajo y el 62% en 3-Medio. Al finalizar el proyecto ninguno se consideró como 2-Bajo, 38% se

ubicaron en 3-Medio, 46% en 4-Alto y 15% en 5-Muy Alto. En este rol se destaca que 15% de los alumnos se movieron de 2-Bajo a 4-Alto, 38% de 3-Medio a 4-Alto, 15% de 3-Medio a 5-Muy Alto. Sin embargo, el 8% consideró que no hubo cambios de la etapa de inicio a la final, respondiendo en ambos casos como 3-Medio (figura 6).

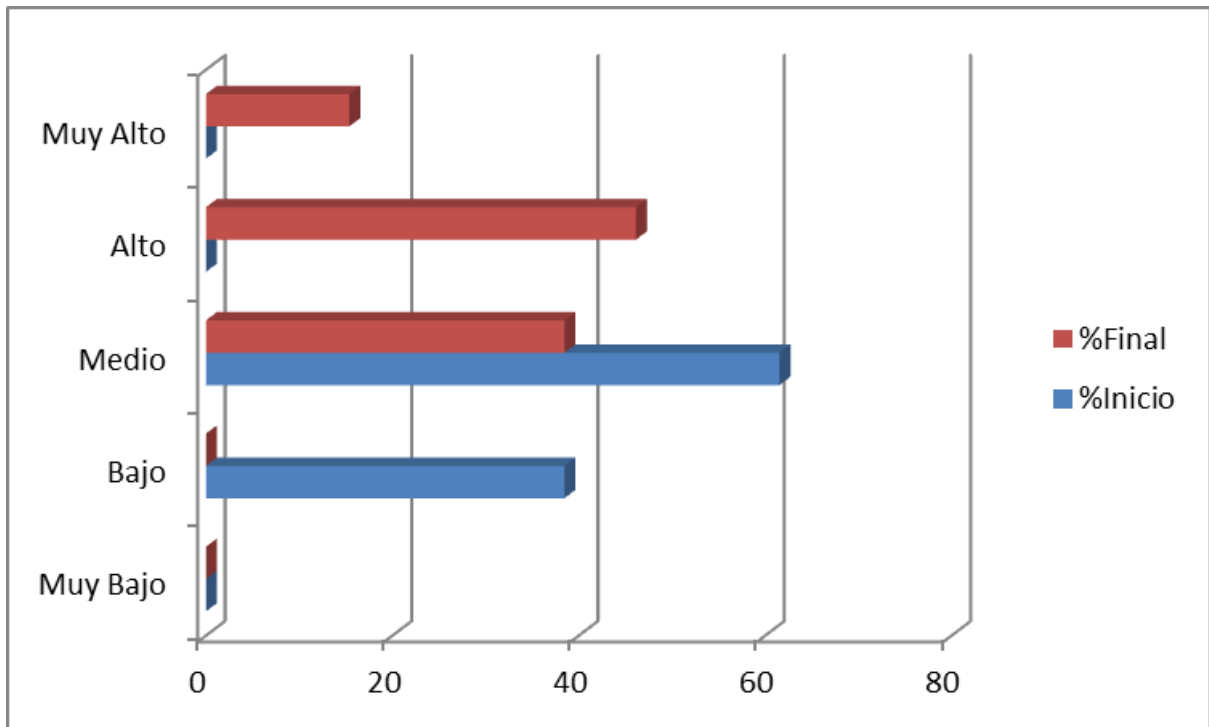


Figura 4.

Desarrollo de habilidades y competencias con el rol Diseñador de arquitectura

En el factor 2) Desarrollo de habilidades de investigación para los 29 alumnos que desempeñaron el rol PR, en la etapa de inicio el 10% respondió como Bajo, 48% Medio, 31% 4-Alto y 10% 5-Muy Alto. Al finalizar ningún alumno respondió como 2-Bajo, 7% como 3-Medio, lo que significó que el 24% del 48% que se encontraba en 3-Medio se movieron a 4-Alto y el 21% de 3-Medio a 5-Muy Alto, el 1% se mantuvo en 3-Medio, mientras que el 1% subió de 2-Bajo a 3-Medio. La respuesta de 5-Muy Alto se incrementó de 10% a 52% alumnos, al

sumarse un 21% de alumnos que se movieron de 4-Alto a 5-Muy Alto y de manera significativa 1% aumentó de 2-Bajo a 5-Muy Alto (figura 7). Al comprometer la entrega del proyecto con un cliente real, se genera en los alumnos una mayor responsabilidad para desarrollar una solución tecnológica que cumpla con las tendencias en la industria de software. La investigación de las tendencias tecnológicas es una de las buenas prácticas en la IS (NMX-I-059/03, 2011).

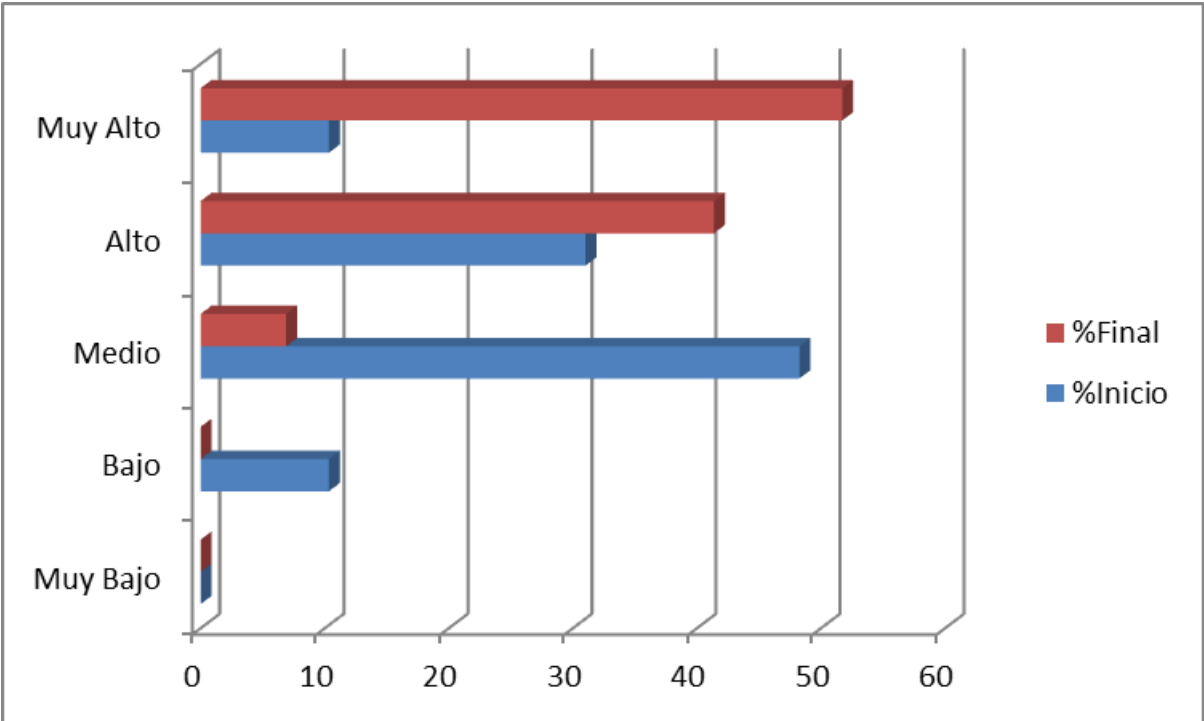


Figura 5.

Desarrollo de habilidades de investigación en Programadores

En el factor 5) Capacidad para comunicarme con los demás de forma eficaz, en el rol de RAPE, Responsable de DMS y/o Líder de proyecto de los 9 alumnos que desempeñaron este rol, el 22% se calificó al inicio como 2-Bajo, 67% 3-Medio y 11% 4-Alto. Al final el 56% respondió como 4-Alto, considerando que un 11% que se encontraban en 2-Bajo se movieron a 4-Alto y un de 33% de 3-

Medio a 4-Alto. El 44% respondió como 5-Muy Alto, sumándose un 11% de 2-Bajo a 5-Muy Alto, 22% de 3-Medio a 5-Muy Alto y un 11% de 4-Alto a 5-Muy Alto. (figura 8). La aplicación del ABP es una estrategia para mejorar la comunicación entre el grupo del proyecto y el resto del grupo de clase o los pares evaluadores (Murphy y Gazi, 2001; Rodríguez-Sandoval et al., 2010). El resultado positivo de esta área de competencia social en este rol es significativo al ser la persona líder responsable de la comunicación con todos los interesados, en particular con el patrocinador del proyecto, el equipo del proyecto y otros interesados clave (PMI, 2008).

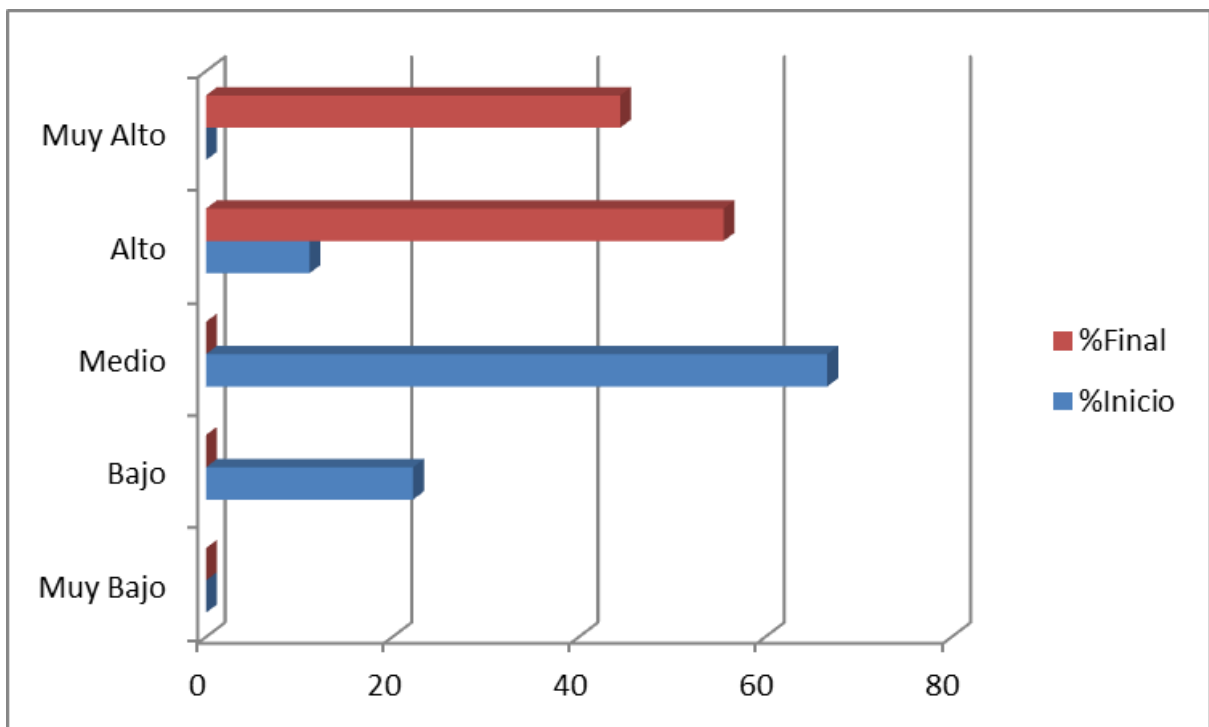


Figura 6.

Capacidad para comunicarme con los demás de forma eficaz

En la tabla 3 se pueden observar las medidas de tendencia central media, mediana y moda para cada uno de los siete factores por los trece roles. Para el caso de la media, los siete factores cambiaron un valor ascendente de 3-Medio

a 4-Alto. A diferencia de los valores de la media, la mediana se observa sin cambio a partir del factor 2 hasta el factor 7. En el cálculo de la moda, en la etapa de inicio la respuesta que representó el mayor número de veces es la opción 3 - Medio para los factores de 1, 2, 3, 4, 5 y 6; mientras que el factor 7 es la opción 4-Alto. En la etapa final, los factores 1, 2, 3, 4, 5 y 6 cambiaron de 3- Medio a 4-Alto y para el factor 7 de 4-Alto a 5-Muy Alto.

Factores	Media Inicial	Media Final	Mediana Inicial	Mediana Final	Moda Inicial	Moda Final
Desarrollar e incrementar conocimientos y habilidades	3	4	3	5	3	4
Desarrollar habilidades de investigación	3	4	3	4	3	4
Desarrollar habilidades de colaboración para generar conocimiento.	3	4	3	4	3	4
Incrementar las capacidades de análisis y de síntesis.	3	4	3	4	3	4
Capacidad para comunicarme con los demás de forma eficaz.	3	4	3	4	3	4
Aprendizaje sobre como evaluar y coevaluar.	3	4	3	4	3	4
Establecer su compromiso en un proyecto	3	4	3	4	4	5

Tabla 3.

Media, Mediana y Moda de los factores evaluados por todos los roles

De las tres medidas de tendencia central utilizadas, la mediana representa la validez de los valores reales en función de las respuestas de los alumnos (Tabla 3). Es así como de los siete factores, el factor 1) Desarrollar e incrementar conocimientos y habilidades, aumentó dos valores desde la etapa inicial de 3-Medio a 5-Muy Alto. En este sentido, se denota que si los alumnos están comprometidos y consideran que este tipo de aprendizaje es importante, por la valiosa experiencia profesional y el desarrollo de habilidades, tendrán una actitud y disposición favorables para afrontar el proyecto durante el curso (Rodríguez Sandoval et al., 2010; Liu et al., 2008).

El APB sitúa a los alumnos en un ambiente real y de colaboración, las experiencias que están viviendo al resolver un problema real son compartidas entre los miembros de un mismo equipo de trabajo y entre los miembros de los demás equipos de trabajo, lo cual genera un intercambio de ideas y reflexiones que los llevan a una indagación sistemática que genera nuevo conocimiento. A su vez, al tomar responsabilidad sobre las actividades que les son asignadas en lo individual para completar las que le son asignadas al equipo de trabajo propicia el querer mejorar su desempeño personal pues saben que serán evaluados entre ellos. Esto también obedece a la solidaridad que se presenta como equipos de trabajo. En el factor 7) Establecer su compromiso en un proyecto, la moda aumentó de 4-Alto a 5-Muy Alto, mostrando que la motivación hacía el trabajo es una de las competencias más significativas para alcanzar los objetivos planteados (Rodríguez Sandoval et al., 2010); principalmente los que han sido establecidos en los proyectos de DMS y que deben satisfacer la necesidad de un cliente real.

Por otro lado, para corroborar si estos resultados se pudieron haber obtenido en un curso (unidad de aprendizaje) sin la modalidad del ABP en el entorno de una empresa como AvanTI, se les preguntó si los resultados al finalizar su participación serían los mismos, encontrando que el 100% consideró que no lo habrían logrado. Esto permite apreciar que los alumnos tienen una actitud positiva hacia su participación en AvanTI. En este caso, se les hizo una

pregunta abierta en la que brindaran sus comentarios sobre el proyecto en el que participaron. Un ejemplo de respuesta es:

“pertenecer a la empresa escolar AvanTI me ayudó a fortalecer los conocimientos y habilidades obtenidas durante mi carrera profesional. Las actividades que desempeñé y el grado de responsabilidad que me asignaron me ayudaron a desenvolverme y a tener confianza en mi misma en el campo laboral”

5.3 Aportación en la práctica profesional

Para validar el grado de las áreas de competencias de los egresados se les preguntó sobre el grado de experiencia adquirida en AvanTI para conseguir su primer empleo y si su puesto inicial fue el mismo que desempeñó en AvanTI. Al comparar los resultados de ambas preguntas (figura 9), se observa que para el 46% de los egresados el grado de influencia fue 5-Muy alto, el 29% de ellos iniciaron su ejercicio profesional con el mismo el rol que desempeñaron en AvanTI y el 17% con un rol diferente. Para la opción 4-Alto, del 34% de los alumnos el 21% se desempeñaron en el mismo rol y 13% en un rol diferente. Al fomentar que el trabajo sea colaborativo e interdisciplinario, se logra que los alumnos aprendan de las funciones de otros roles, esto sucedió no sólo en la categoría de OPE sino también en la categoría de GES, lo que les permite desempeñarse en más de un rol en su ámbito profesional. El 17% de alumnos que contestaron 3-Medio, su primer rol no corresponde a ninguno de los roles de AvanTI como en el caso de soporte técnico. Uno de las respuestas para la opción de 1-Muy Bajo, se debe a que uno de los alumnos antes de su ingreso a AvanTI se encontraba trabajando en un puesto de apoyo administrativo, así que a diferencia de los demás alumnos fue su experiencia profesional lo que influyó en la elección de los roles que eligió en AvanTI, correspondientes a los procesos de GR y RHAT.

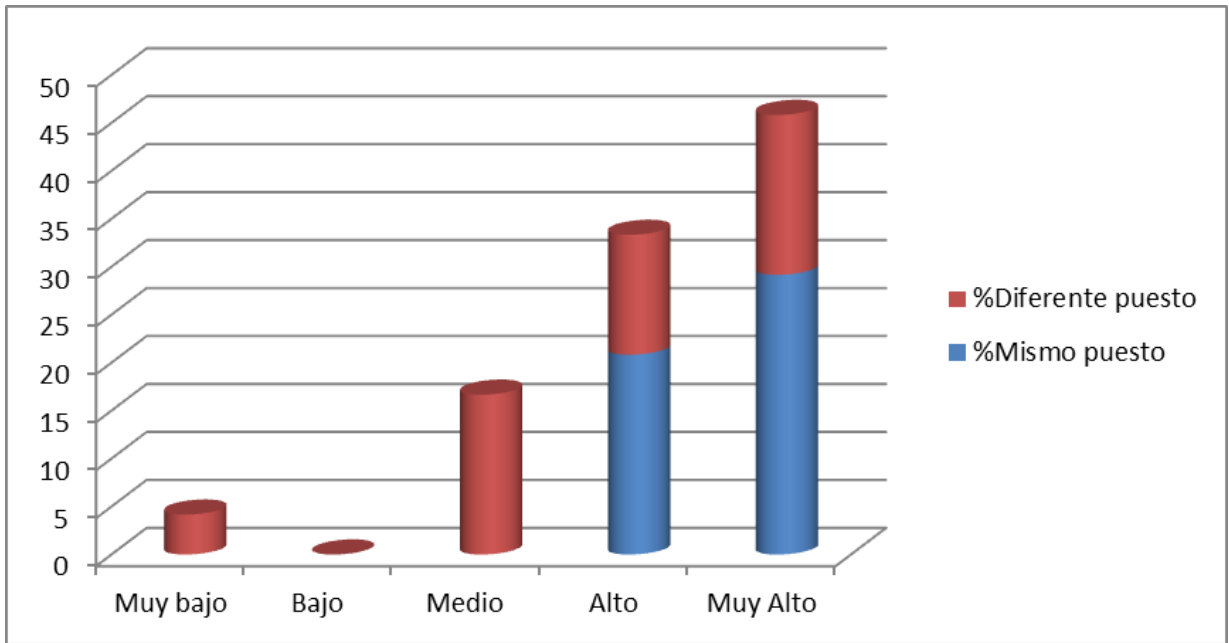


Figura 7.

Relación entre el grado de competencias obtenido en AvanTI y su relación con el rol de su primer empleo

Una situación que también es importante para la motivación es el hecho de que el alumno suele identificarse en un ambiente similar al ámbito profesional, en este caso, en una empresa de base tecnológica dedicada al desarrollo de software, desempeñando un rol que en un futuro próximo ejercerá como egresado. Para validar esta afirmación, se detectó que de los roles desempeñados en AvanTI, el 100% respondieron que también han sido desempeñados por los egresados en las empresas en donde trabajan. La figura 10 muestra que los roles en los que más se han desempeñado son AN con un 65%, en segundo lugar PR y RPU con un 56% y en tercer lugar DU con 52%. La opción de otros roles se encuentran el de responsable de soporte a usuarios y redes, supervisor del área de TI e ingeniero en manufactura. Al igual que en AvanTI, algunos desempeñan más de un rol, pues la definición de las funciones en los puestos que ofertan las empresas cubren a más de un rol. Se

observa también que hay quienes ya tienen puestos de Responsables de Proyectos con un 30%.

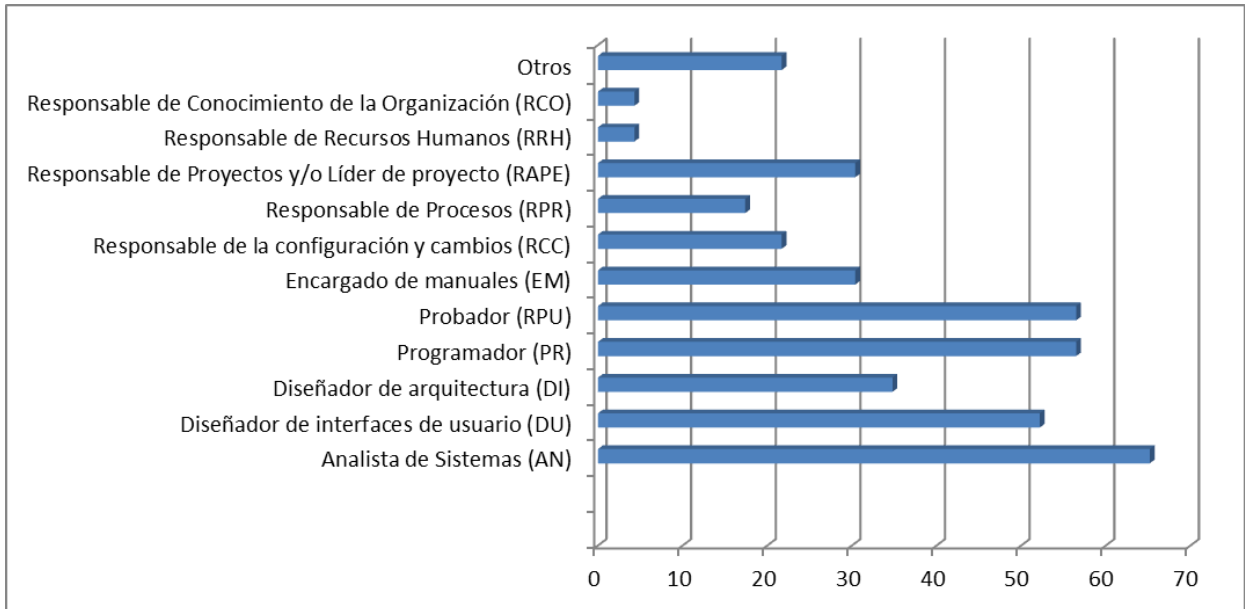


Figura 8.

Puestos y roles en los que se han desempeñado los egresados de la empresa escolar

Este trabajo expone por medio del proceso de desarrollo el enfoque de ABP, la explotación de oportunidades para la creación y desarrollo de una empresa escolar de base tecnológica. El análisis de este tipo de organización es de interés por su imprescindible rol en la competitividad y crecimiento de la industria de software mexicana, específicamente para el estado de Baja California. Actualmente, en dicho estado son siete las organizaciones dedicadas al desarrollo de software que han sido verificadas bajo la norma NMX-I-059-NYCE-2011 (SE, 2014; CERTVER, 2015), donde dos de ellas están en proceso de implementación del modelo CMMI v1.3. Se ha observado que para que las empresas logren una implementación exitosa de sus proyectos de mejora de procesos de software, no sólo adoptaron un modelo de referencia de

procesos o modelo de procesos de software (Gómez Álvarez et al., 2014) sino, como factores críticos de éxito, involucraron a personal con capacidades y competencias adecuadas y realizaron capacitaciones necesarias para elevar el conocimiento colaborativo del equipo de trabajo. De los egresados que se desempeñaron como RPR, uno de ellos actualmente está implementando el modelo de referencia de procesos CMMI Dev 1.3 Nivel 2.

6. Conclusiones

La aplicación del método de Aprendizaje Basado en Proyectos (ABP) como estrategia de aprendizaje en el entorno de la mejora de procesos de software, permitió establecer un diseño instruccional capaz de adaptar los pasos metodológicos a la realidad de una empresa escolar de base tecnológica, con el propósito de crear un ambiente de aprendizaje real-laboral.

Se formaron equipos de trabajo con roles definidos de acuerdo a la realidad de la industria mexicana del software. Se motivaron y, a su vez, crearon responsabilidades sobre la solución de problemas reales asociados a proyectos de gestión y de Ingeniería de Software. De tal forma, que la aproximación a la realidad que propicia el ABP y la participación en una empresa escolar de base tecnológica ha permitido en los alumnos desarrollar e incrementar sus competencias a partir de un aprendizaje autónomo y colaborativo; sus habilidades para comunicarse, relacionarse, negociar, tomar decisiones, investigar, generar nuevo conocimiento, tener autonomía en su aprendizaje; valorar la responsabilidad, el compromiso, el respeto, la honestidad consigo mismo y con los demás al evaluar el desempeño, entre otros aspectos.

Se aplicó un instrumento de validación social para conocer la auto-evaluación que los alumnos tienen sobre sus propias competencias en la ejecución de actividades asignadas a su rol dentro de la empresa escolar. Las instrucciones pedían al alumno que respondiera para las etapas inicial y final, cada uno de

los siete factores relacionados a las competencias que se favorecen en un método ABP. Conforme a los resultados, se observaron casos en que los alumnos respondieron en la etapa inicial la opción de 5-Muy Alto, mientras que para otros en la etapa final no hubo cambios en el desarrollo de sus competencias, tal es el caso del rol DI. Este instrumento permitió conocer la perspectiva de los estudiantes en relación al desarrollo de un proyecto real.

Por otro lado, se puso a prueba cómo el aprendizaje guiado desde el proceso de desarrollo del método de ABP se relaciona con procesos de desarrollo de software implementados en la empresa escolar. Este aspecto no fue abordado a primera instancia en este documento, pero se cuenta con el análisis de los hallazgos. Es de interés de los autores dar seguimiento a la cuantificación y evaluación del impacto de los alumnos que participaron en AvanTI sobre las quince empresas u organizaciones que utilizan procesos de calidad y/o mejora de procesos tanto nacionales como internacionales.

Finalmente, este documento trata de ser útil y práctico para aquellos que deciden incorporar estrategias de aprendizaje que motiven al alumno a la solución de problemas reales con un enfoque de proyectos y/o deseen emprender por primera vez, en empresas escolares de base tecnológica.

Agradecimientos

Este trabajo está asociado al proyecto 111/942 registrado en la Coordinación de Posgrado e Investigación de la UABC.

Referencias

Alcober, J., Ruíz, S., Valero, M., 2003. Evaluación de la implantación de. Aprendizaje basado en proyectos en la EPSC (2001-2003). XI Congreso Universitario de Innovación Educativa en las Enseñanzas Técnicas, Vilanova i la Geltrú, pp. 23-25.

Anaya R., 2012. 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.

Arellano-Pimentel, J. J., Nieva-García, O., & Algreto-Badillo, I. 2013. Aprendizaje Basado en Proyectos Utilizando L-Systems en un Curso de Compiladores. Programación Matemática y Software. Vol. 5 (1). pp. 82-96.

Astorga, M., Flores B., Chávez G., Lam M., Justo, A., 2010. Lecciones Aprendidas en la Implantación de MoProSoft en una empresa escolar: caso AvantI. Revista Ibérica de Sistemas y Tecnologías de la Información, Vol. 6, 2010, pp. 73-86.

Belloch, C., 2012. Diseño instruccional. Disponible en <http://www.uv.es/bellochc/pedagogia/EVA4.pdf>.

Beneitone, P. et al., 2007. Reflexiones y perspectivas de la educación superior en América Latina. Proyecto Tuning. Spain/impreso.

Blank, W., 1997. Authentic instruction. In W.E. Blank & S. Harwell (Eds.), Promising practices for connecting high school to the real world, pp. 15-21. University of South Florida. USA. 1997.

Busenitz, L. W.; West III, G. P.; Shepherd, D.; Nelson, T.; Chandler, G. N.; y Zacharakis, A., 2003. Entrepreneurship research in emergence: past trends and future directions. *Journal of Management*, Vol. 29 (3), 2003, pp. 285-308.

Carver, J., Jaccheri, L., Morasca, S., and Shull, F., 2003. Issues in Using Students in Empirical Studies in Software Engineering Education. In *METRICS '03: Proceedings of the 9th International Symposium on Software Metrics* (Washington, OC, USA), IEEE Computer Society. p. 239.

Casallas R., Davila J. I., Quiroga, J. P., 2015. Enseñanza de la ingeniería de software por procesos instrumentados. Disponible en <http://www.willydev.net/>

CERTVER, 2015. Lista de empresas verificadas. Disponible en: <http://www.certver.com/empresas/> Consultado el 2 de Febrero de 2015.

CIDAC, 2015. Encuesta de competencias laborales. Centro de Investigación para el Desarrollo. Disponible en <http://www.cidac.org/> Consultado el 2 de Febrero de 2015.

Colomo, R., Casado, C., Soto, P., García, F., Tovar, E., 2013. Competence gaps in software personnel. A multiorganizational study. *Computers in Human Behavior*, Vol. 29, pp. 456-461.

Comisión Europea, High-Tech SMEs in Europe, 2002. Observatory of European SMEs 2002/6 European Network for SME Research ENSR.

Díaz Sánchez, E., Souto Pérez, J. E. y Tejeiro Koller, M. R., 2013. NEBTS 3 Nuevas empresas de base tecnológica. Caracterización, necesidades y evolución en un periodo de crecimiento y en otro de alentización y recesión económica (2004-2012). España. pp. 170. 2013.

Díaz, E., 2010. Evolución de las empresas tecnológicas en sus primeros años de vida. Fundación Madri+d (Ed). Nuevas empresas de base tecnológica. Colección Madri+d num. 37, Fundación Madri+d para el Conocimiento y Comunidad de Madrid. pp. 44-57.

Disla García, Y. I., 2013. Aprendizaje por proyecto: Incidencia de la tecnología de la información para desarrollar la competitividad de trabajo colaborativo. Ciencia y Sociedad, Vol. 38 (4), pp. 691-718.

Galeana de la O, L., 2006. Aprendizaje basado en proyectos. Revista Digital Educación a Distancia. México.

Gallego, M., Gortázar, F., 2009. EclipseGavab.: Un entorno de desarrollo para la docencia online de la programación. XV Jornadas de Enseñanza Universitaria de la Informática (XV JENUI). Barcelona. pp. 501-508.

Galvan, P. 2014. Encuesta de salarios. Revista Software Guru, Vol. 46. Disponible en <http://sg.com.mx/revista/46/estudio-salarios-2014#.VfDJexGeDRY>

Garcia, I. y Pacheco, C. 2012. Using TSPi and PBL to support software engineering education in an upper-level undergraduate course. Computer Applications in Engineering Education, Published online in Wiley Online Library. DOI: 10.1002/cae.21566

Garcia, I., Pacheco, C. y Coronel, N. 2010. Learn from practice: defining an alternative model for software engineering education in Mexican universities for reducing the breach between industry and academia. Proc. of the 2010 International Conference on Applied Computer Science, WSEAS Press. pp. 120-124.

Gómez Álvarez, M. C., Gasco Hurtado, G.P., Calvo-Manzano J.A., San Feliu, T., 2014. Diseño de un instrumento pedagógico para la enseñanza de la mejora

de procesos de software: Instrumento de enseñanza para ambientes universitarios y empresariales. 9ª Conferencia Ibérica de Sistemas y Tecnologías de Información. España. pp. 295-301.

Guitart, I., Rodríguez, M. E., Cabot, J. y Serra, M., 2006. Elección del modelo de evaluación: caso práctico para asignaturas de ingeniería del software. Actas las XII Jornadas Enseñanza Univ. Informática. Jenui. pp.191–198.

Höst, M., Regnell, B., and Wohlin, C., 2000. Using Students as Subjects: A Comparative Study of Students and Professionals in Lead-time Impact Assessment. Empirical Software Engineering. Vol. 5 (3). pp. 201-214.

ISO 29110, 2011. Software engineering -- Lifecycle profiles for Very Small Entities (VSEs). Disponible en <https://www.iso.org/obp/ui/#iso:std:51154>

Kolmos, A., 2004. Estrategias para desarrollar currículos basados en la formulación de problemas y organizados en base a proyectos. Educar. No. 33, pp. 77-96.

Lethbridge, T.C. 2000. What knowledge is important to a software professional?. Computer. Vol. 33(5). pp. 44-50.

Martí, J. A., et al., 2010. Aprendizaje basado en proyectos: una experiencia innovadora docente. Revista Universidad EAFIT, Vol. 46 (158). Colombia. pp. 11-21.

Murphy, KL., Gazi, Y., 2001. Role plays, panel discussions and simulations: Project-based learning in a web-based course. Education Media International. Vol. 38 (4). pp. 261-269.

NMX-I-059/02-NYCE-2011, 2011. Tecnología de la Información - Software - Modelos de Procesos y Evaluación para el Desarrollo y Mantenimiento de Software. Parte 02, Requisitos de Procesos (MoProSoft). 2a ed. Ed. NYCE. México DF., México. pp. 62.

NMX-I-059/03-NYCE-2011, 2011. Tecnología de la Información - Software - Modelos de Procesos y Evaluación para el Desarrollo y Mantenimiento de Software. Parte 03, Guía de implantación de procesos. 2a ed. Ed. NYCE. México DF., México.

OCDE 2002. Proyecto DeSeCO. Disponible en <http://www.deseco.admin.ch/bfs/deseco/en/index/03/02.parsys.78532.downloadList.94248.DownloadFile.tmp/2005.dscexecutivesummary.sp.pdf> – Consultado agosto de 2015.

Pavié, A., 2011. Formación docente: hacia una definición del concepto de competencia profesional docente. REIFOP. Vol. 14 (1). pp. 67-80. Disponible en <http://www.aufop.com> – Consultado en agosto de 2015.

PMI, 2014. Guía de los Fundamentos para la Dirección de Proyectos (Guía del PMBOK®)—Quinta edición. Project Management Institute. ISBN 9781628250091.

R. Juárez-Ramírez,K. Cortés Verdín,B. A. Toscano de la Torre,H. Oktaba,C. A. Fernández y Fernández,B. L. Flores Rios,F. Angulo Molina, 2013. Estado Actual de la Práctica de la Ingeniería de Software en México.

Reitmeier, CA., 2002. Active learning in the experimental study of food. Journal of Food Science Education,Vol. 1. pp. 41-44.

Reyes, R., 1998. Native perspective on the school reform movement: A hot topics paper, Northwest Regional Educational Laboratory, Comprehensive Center Region X.

Rodríguez-Sandoval, E.; Vargas-Solano, É. M.; Luna-Cortés, J., 2010. Evaluación de la estrategia aprendizaje basado en proyectos. Educación y Educadores. Abril-Sin mes. pp. 13-25.

Runeson, P., 2003. Using Students as Experiment Subjects-An Analysis on Graduate and Freshmen Student Data. In Proceedings of the 7th International Conference on Empirical Assessment in Software Engineering (Keele University,UK). pp. 95-102.

Sánchez Martínez, A., Rojas Molina, A., y Sánchez Flores, L. 2015. Transformando la educación tradicional a través del aprendizaje basado en proyectos y el uso de las tecnologías de la información y comunicación. Repositorio Digital Universitario de Materiales Didácticos. Universidad Nacional Autónoma de México.

Santa, J., Zamora, M. A., Ubeda, B., 2008. El aprendizaje basado en proyectos en materias de ingeniería informática y sus implicaciones. I Jornadas sobre nuevas tendencias en la enseñanza de las ciencias y las ingenierías.

SEC. School Enterprise Challenge. 2015. Disponible en <http://www.schoolenterprisechallenge.org/es/about-2/que-es-el-concurso-escuela-emprededora/que-es-una-empresa-escolar/> Consultado agosto de 2015.

SE, 2014. Centros de desarrollo certificados/verificados vigentes en modelos de calidad. Secretaría de Economía. Disponible en <http://www.prosoft.economia.gob.mx/>

Smilor, R. W.; Gibson, D. V.; y Dietrich, G. B., 1990. University spin-out companies: technology start-ups from UT-Austin. Journal of Business Venturing, Vol. 5 (1). pp. 63-76.

Sousa, DA., 1995. How the Brain Learns. Reston, VA: The National Association of Secondary School Principals, 143 pp.

Steffensen, M.; Rogers, E. M.; y Speakman, K., 2000. Spin-off from research centers at a research university. Journal of Business Venturing. Vol. 15 (1). pp. 93-111.

Tippelt, R. y Lindemann, H., 2001. El Método de Proyectos. Berlín, 2001.

UABC, 2006. Estatuto Escolar de la Universidad Autónoma de Baja California.

UABC, 2009. Proyecto de reestructuración del programa educativo de Licenciado en Sistemas Computacionales. Facultad de Ingeniería Mexicali. Universidad Autónoma de Baja California. México.

UABC, 2010. Guía metodológica para la creación y modificación para los programas educativos de la Universidad Autónoma de Baja California. Cuadernos de Planeación y Desarrollo Institucional. Disponible en <http://www.uabc.mx/planeacion/cuadernos/c15.pdf>

Notas biográficas:



María Angélica Astorga Vargas Maestra en Ciencias en el área de Computación. Profesora titular de tiempo completo en el Programa Educativo de Licenciado en Sistemas Computacionales de UABC. Su área de investigación es Modelos de Procesos de Software. Pertenece a la Red Temática Mexicana de Ingeniería de Software. Ha participado como consultora en la implementación de la NMX-I-059-NYCE-2011 y como Miembro de Equipo Evaluador en SCAMPI A de CMMI. Cuenta con experiencia en los procesos de reestructuración y acreditación de planes de estudios de licenciatura en la Facultad de Ingeniería, UABC Mexicali.



Brenda Leticia Flores Rios Doctora en Ciencias por la UABC. Responsable del área de Ingeniería del conocimiento del Instituto de Ingeniería, desarrollando proyectos de investigación relacionados a la Gestión del Conocimiento y Mejora de procesos de software. Imparte docencia en licenciatura y posgrado. Pertenece a la Red Temática Mexicana de Ingeniería de Software. Ha participado como consultora en la implementación de la NMX-I-059-NYCE-2011 y como Miembro de Equipo Evaluador en SCAMPI A de CMMI.



Jorge Eduardo Ibarra Esquer Maestro en Ciencias por el Centro de Investigación Científica y de Educación Superior de Ensenada. Se desempeñó como coordinador del Programa Educativo de Ingeniero en Computación de UABC. Actualmente, es Profesor titular de tiempo completo y desarrolla líneas de investigación en enseñanza de la programación, sistemas electrónicos digitales e internet de las cosas. Pertenece a la Red Temática Mexicana de Ingeniería de Software. Cuenta con experiencia en los procesos de reestructuración y acreditación de planes de estudios de licenciatura ante el organismo Consejo Acreditador de la enseñanza de la Ingeniería Superior, A.C. (CACEI).



Josefina Mariscal Camacho Maestra en Ingeniería en Sistemas. Actualmente, es Coordinadora del Programa Educativo de Licenciado en Sistemas Computacionales y profesora titular de tiempo completo de UABC. Su área de investigación es Redes y Telecomunicaciones. Ha participado en los procesos de reestructuración del plan de estudios de la Licenciatura en Sistemas Computacionales de la Facultad de Ingeniería, UABC. Es responsable del proceso de reacreditación ante el organismo acreditador Consejo Nacional de Acreditación en Informática y Computación, A.C. (CONAIC).



Luis Enrique Vizcarra Corral Maestro en Ciencias en Ciencias de la Computación por el Centro de Investigación Científica y de Educación Superior de Ensenada. Profesor titular de tiempo completo en el Programa Educativo de Licenciado en Sistemas Computacionales de la Facultad de Ingeniería, UABC Mexicali. Ha participado en al menos 10 proyectos de investigación aplicada, vinculación y de desarrollo tecnológico con financiamiento externo e interno. Cuenta con experiencia en los procesos de reestructuración de planes de estudio, así como en la obtención de acreditación por parte del Consejo Nacional de Acreditación en Informática y Computación, A.C. (CONAIC).



Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 2.5 México.