

Reflexiones sobre —————○
**Ingeniería de Requisitos
y Pruebas de Software**



Reflexiones sobre —————●
**Ingeniería de Requisitos
y Pruebas de Software**





Echeverri, Jaime

Reflexiones sobre Ingeniería de Requisitos y Pruebas de Software/ Echeverri, Jaime; Aristizábal, Miguel; González, Liliana; Urrego, Germán, Polo, Ricardo [et al]. Medellín: Corporación Universitaria Remington y Organización LACREST, 2013.

126 p.

ISBN: 978-958-58070-3-7

1. Ingeniería de Software 2. Ingeniería – requisitos 3. Ingeniería – validación 4. Ingeniería - verificación

CDD – 005.1

Reflexiones sobre Ingeniería de Requisitos y Pruebas de Software

ISBN: 978-958-58070-3-7

Primera edición, diciembre de 2013.

Comité organizador LACREST:

Director general: Miguel Buitrago, CEIS

Director ejecutivo: Jorge Mauricio Sepúlveda, Corporación Universitaria Remington

Director comercial: César Sáenz, Corporación Universitaria Remington

Directora de comunicaciones: Lina María Alvarado, Corporación Universitaria Remington

Director de logística: Blanca Restrepo, Corporación Universitaria Remington

Comité científico LACREST:

Directora: Raquel Anaya, Universidad EAFIT, Colombia

Miembros:

Abrahán Eliseo Dávila, Pontificia Universidad Católica del Perú, Perú

Alfredo Matteo, Universidad Central de Venezuela, Venezuela

César Pardo Calvache, Universidad San Buenaventura, Colombia

Francisca Losavio, Universidad Central de Venezuela, Venezuela

Francisco Pino, Universidad del Cauca, Colombia

Gustavo Rossi, Universidad de La Plata, Argentina

Hernán Astudillo, Universidad Técnica Federico Santa María, Chile

Hugo Arboleda, Universidad ICESI, Colombia

Jorge Walter Orellana, Universidad Mayor de San Simón, Bolivia

Juan Pablo Carvallo, Universidad de Azuay, Ecuador

Marta Tabares Universidad de Medellín, Colombia
Mauricio Alba-Castro, Universidad Autónoma de Manizales, Colombia
Miguel Eduardo Torres, Pontificia Universidad Javeriana, Colombia
Ramón García Martínez, Universidad Nacional de Lanús, Argentina
Richard Rosales, Universidad de los Andes, Venezuela
Sergio Zapata, Universidad de San Juan, Argentina

Coordinadores de la publicación:

Raquel Anaya de Páez, directora académica de LACREST 2013, Universidad EAFIT, Colombia

Jorge Mauricio Sepúlveda, director ejecutivo de LACREST 2013, Corporación Universitaria Remington

Miembros del Comité Editorial de la Corporación Universitaria Remington:

Pedro Juan González Carvajal, rector
Andrés Mauricio Higuera Palacio, vicerrector académico
Margarita María Zapata Restrepo, directora general de investigación
Héctor Augusto Jiménez Arboleda, coordinador de publicaciones Dicur
Lina María Alvarado Pérez, directora de comunicaciones
Adriana Patricia Bustamante Fernández, jefe de biblioteca
César Augusto Muñoz Restrepo, corrector de estilo institucional

Dirección Fondo Editorial Remington:

Corporación Universitaria Remington
Editora en Jefe: Margarita María Zapata Restrepo
E-mail: mzapata@remington.edu.co
Dirección: Calle 51 No 51-27, Edificio Remington
Telefax: (57) (4) 5111000. Medellín, Colombia
<http://corporacion.remington.edu.co/fondo-editorial/publicaciones>.

Diseño y diagramación: Víctor Andrés Royo Grandeth.

Notas legales:

“Las opiniones expresadas por el autor no constituyen ni comprometen la posición oficial o institucional de la Corporación Universitaria Remington”.

“Está prohibida la reproducción total o parcial por cualquier medio sin autorización escrita de la Corporación Universitaria Remington”.



Tabla de contenido

	Pág.
Presentación	8
Parte 1, Ingeniería de Requisitos	11
Modelo para la gestión de la trazabilidad de contribuciones en ambientes co-creados	12
Propuesta integral de manejo de requerimientos en proyectos de explotación de información	26
Estudio del estado actual del proceso de ingeniería de requisitos en las empresas antioqueñas de software	45
Parte 2, Verificación y Validación	64
Integración de pruebas automáticas para la optimización de los procesos de producción de software en un estudio de caso real	65
Parte 3, Aplicaciones de la Ingeniería de Software	76
Desarrollo Ágil usando XRX: un caso práctico	77
Análisis de Exámenes en Carreras de Sistemas mediante Procesos de Explotación de Información	97
Nuevas estrategias para realizar evaluaciones en cursos de ingeniería de software: caso Universidad de Medellín	112





Presentación

Bienvenidos a la segunda versión del Congreso Latinoamericano de Ingeniería de Requisitos y Pruebas de Software LACREST 2013, que se realiza en la ciudad de Medellín, Colombia, del 4 al 6 de diciembre del 2013 y está organizado por la Corporación Universitaria Remington con el apoyo de la Universidad EAFIT.

Son diversas las actividades programadas en el marco de LACREST2013: Tendremos 2 tutoriales, 6 conferencias plenarias y la presentación de 12 trabajos que han sido sometidos, evaluados y aceptados por un comité académico.

Los tutoriales están enmarcados en los temas principales del congreso y están dirigidos a mostrar a la comunidad, prácticas recientes que están siendo aplicadas en la industria de software buscando mejorar la calidad tanto del producto como del proceso: en primer lugar Cecile Peraire, profesora de la Universidad de Carnegie Mellon en Silicon Valley, con más de 20 años de experiencia en Ingeniería de Software, estará discutiendo las tendencias actuales de los enfoques ágiles que combinan prácticas del marco de trabajo Lean. El segundo lugar, Raul De Villa, arquitecto de soluciones de Tech and Solve, estará presentado prácticas de pruebas e integración que apoyan el desarrollo ágil.

Las conferencias plenarias enmarcan temáticas de gran interés para la industria de software y servicios relacionados: En la línea de pruebas de software, Marcelo Jenkins (catedrático de la Escuela de Computación e Informática de la Universidad de Costa Rica), compartirá la experiencia de aplicación del control estadístico en los procesos de verificación y validación; Maria Clara Choucair (gerente general de Choucair Testing S.A.), discutirá el papel de los casos de prueba como elementos para evaluar la calidad de un producto software. En la línea de ingeniería de requisitos, Dante Carrizo (investigador de la Universidad de Atacama, Chile) discutirá la importancia de adaptación del proceso de ingeniería de requisitos, a las condiciones del contexto. Desde una visión más general, Gabriel Vásquez (director de soluciones estratégicas de AG LATAM), presentará la experiencia interesante de implementación de una solución integral para el manejo de las peticiones, quejas y reclamos para el Municipio de Medellín; Cecile Peraire (docente de la Universidad de Carnegie Mellon, Silicon Valley, USA), presentará una perspectiva interesante de la educación en ingeniería de software apoyada en el marco de trabajo de SEMAT; Albeiro Cuesta (representante del Ministerio de las TIC, Colombia) presentará el estado actual y las proyecciones de la industria TIC en Colombia.

Contaremos con tres espacios de participación abierta donde representantes del gobierno, la academia y la industria, discutirán experiencias y visiones acerca del estado actual de la industria de software

y servicios relacionados: En primer lugar, Angela Uribe (representante del Ministerio del Trabajo) socializará la experiencia del proyecto piloto que ha venido desarrollándose en el Cluster TIC de Medellín, para potenciar la gestión del recurso humano del sector. En segundo lugar, se analizarán las experiencias de empresas reconocidas de la ciudad (Suramericana S.A., Intergrupo S.A., Ceiba S.A.) en la adopción de los enfoques ágiles y finalizaremos el evento con un foro donde se discutirán los retos actuales de la industria TIC y cómo enfrentarlos.

Los trabajos aceptados en el congreso siguieron un proceso formal de revisión. Agradecemos a todos los grupos nacionales y latinoamericanos que enviaron sus trabajos y esperamos que esta iniciativa LACREST, pueda fortalecerse como un espacio de encuentro de academia, industria y gobierno que busca acortar la brecha entre el estado de la teoría y el estado de la práctica de la Ingeniería de Software.

*Raquel Anaya de Páez, Universidad EAFIT
Directora académica de LACREST 2013*

*Jorge Mauricio Sepúlveda, Corporación Universitaria Remington
Director ejecutivo de LACREST 2013*



Reflexiones sobre ———●
**Ingeniería de Requisitos
y Pruebas de Software**



Parte 1

Ingeniería de Requisitos





M

odelo para la gestión de la trazabilidad de contribuciones en ambientes co-creados

Echeverri Jaime¹, PhD (c); Aristizábal Miguel², PhD (c); González Liliana³, PhD (c); Urrego Germán⁴, PhD; Polo Ricardo⁵

¹ Universidad de Medellín, Programa Ingeniería de Sistemas, Colombia. jaecheverri@udem.edu.co

² Universidad de Antioquia, Departamento de Ingeniería de Sistemas, Colombia. middass@gmail.com

³ Universidad de Medellín, Programa Ingeniería de Sistemas, Colombia. lgonzalez@udem.edu.co

⁴ Universidad de Antioquia, Departamento de Ingeniería de Sistemas, Colombia. gurrego@udea.edu.co

⁵ Une EPM Telecomunicaciones, Director Desarrollo de Producto, Colombia. rikardo.polo@gmail.com

Resumen

El objetivo del presente trabajo es presentar un modelo para la gestión de la trazabilidad de contribuciones en ambientes co-creados, que se aplica dentro del ciclo de innovación. La metodología establecida parte por la introducción de un modelo de innovación definido por el grupo de trabajo, sus fases y las relaciones entre ellas, estableciendo un conjunto de actividades secuenciales y transversales. Posteriormente se define el concepto de trazabilidad aplicado a las contribuciones para la co-creación de productos innovadores y se establecen algunas definiciones importantes para comprender el modelo que

se propone. Finalmente se detalla el modelo de trazabilidad propuesto y los tipos de enlaces definidos para relacionar los artefactos obtenidos durante el ciclo.

Palabras clave: co-creación, innovación, modelo de innovación, trabajo colaborativo, trazabilidad de contribuciones.

Abstract

The objective of the present work is to present a model for the traceability of contributions management in cocreated environments, which is applied within the innovation cycle. The established methodology starts by the introduction of an innovation model defined by the work Group, its phases and the relations among them in order to establish a set of sequential and transverse activities. Afterwards, the traceability concept applied to co-creation contributions of innovative products is defined and some important definitions are presented so as to understand the proposed model. Finally, the proposed model of traceability is highlighted and the types of links are defined to relate the obtained artifacts during the cycle.

Key words: co-creation, collaboration, models of innovation, traceability of contributions.

Introducción

El diseño y construcción de productos o servicios que incorporan algún tipo de novedad, se componen de un número indeterminado de artefactos que evolucionan en el tiempo y que pasan por diferentes periodos o etapas de maduración. El modelo de innovación propuesto por el grupo de trabajo ITOS de la Universidad de Antioquia, se compone de once etapas de tipo secuencial que obedecen a un conjunto de pasos ordenados en el tiempo y cuyo inicio y fin no necesariamente coincide con el comienzo y terminación del proceso de innovación completo, y un conjunto de fases transversales que están activas durante todo el proceso de innovación, y que contienen actividades que deben ser ejecutadas en cualquier momento con el fin de apoyar cualquiera de las fases secuenciales definidas. La figura 1 muestra la arquitectura del modelo propuesto. En la tabla 1 se presentan los nombres de cada una de las fases secuenciales del modelo.

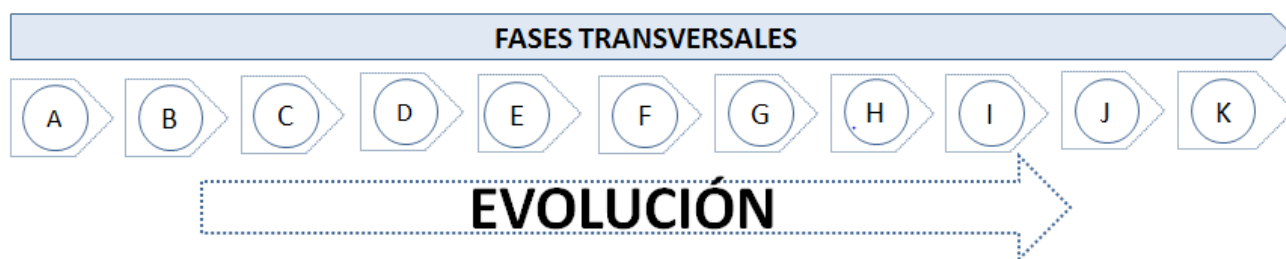


Figura 1. Fases en el ciclo de innovación propuesto.

Tabla 1. Etapas secuenciales del modelo de innovación.

Fase	Descripción
A	Identificación de oportunidades
B	Generación de ideas
C	Propuesta de producto o servicio
D	Definición de requerimientos/necesidades
E	Modelado conceptual
F	Diseño
G	Construcción
H	Pruebas
I	Operación
J	Distribución
K	Impacto

En el modelo se parte de una oportunidad/necesidad que da origen a la generación de ideas; éstas pueden transformarse en diseños, documentos, bosquejos, planos, prototipos, etc. Su evolución permite la obtención de artefactos cada vez más elaborados. Cada artefacto tiene su propio nivel de abstracción, por ejemplo la construcción de una vivienda evoluciona pasando por diferentes etapas, las cuales involucran un conjunto de artefactos, que evolucionan en el tiempo y que permiten validar el cumplimiento de las necesidades iniciales [1].

En la construcción de software se evoluciona desde las necesidades iniciales, pasando por etapas de captura de requisitos, la fase de análisis, el diseño, la implementación y las pruebas. De la misma forma la construcción de productos y servicios pasa por un número indeterminado de etapas evolutivas, con diferentes niveles de abstracción y que permiten adicionar detalles que dan forma al producto o servicio [3].

Diferentes modelos se han propuesto para hacer seguimiento y evolución de requisitos para obtener productos ajustados a los requerimientos iniciales, la mayoría de estos modelos se proponen en el contexto de la ingeniería de software, donde se dispone de un conjunto de fases bien definidas y para el cual hay herramientas que facilitan la interpretación de estas y los modelos o artefactos son característicos de la fase sobre la cual se trabaja, por ejemplo documentos de requisitos en la fase de análisis, diagramas de clase en la fase de diseño, entre otros [4].

Los avances entre fases implican la modificación de la representación del conocimiento acumulado, esto evidencia en diversos modelos una pérdida de conocimiento, pues en muchas situaciones no se hace posible representar el mismo conocimiento con modelos diferentes, y es aún más complicado garantizar la consistencia entre modelos, adicionalmente las herramientas empleadas no garantizan en todos los casos el mantenimiento del significado (semántica) del conocimiento [5].

Se presentan pues, una serie de relaciones entre los modelos y el producto final, incluso hasta etapas que determinan el impacto del producto (bien o servicio) en el medio y las expectativas del cliente. El almacenamiento y la representación de estas relaciones entre los modelos facilitan la comprensión de las partes relacionadas con otros modelos. La capacidad de rastrear hacia atrás y adelante entre los modelos es una característica fundamental para la evolución y el mantenimiento y el control en productos innovadores [6].

El presente trabajo está organizado de la siguiente manera; en el capítulo 2 se trata el concepto de trazabilidad se exponen las fortalezas de los procesos que incorporan esta práctica. En el capítulo 3 se propone un modelo para el proceso de innovación para empresas que tienen involucrada la co-creación. El modelo permite su configuración acorde con el contexto de cada empresa. El capítulo 4 articula la co-creación y sus fases, para facilitar la interacción de los agentes involucrados en la co-creación. El capítulo 5 describe el marco del trabajo de trazabilidad basando en reglas que pueden o no ser transformadas en el lenguaje de destino. El capítulo 6 plantea la representación de los vínculos de los artefactos generados diferenciando los niveles de abstracción según la trazabilidad horizontal y vertical. Por último, el capítulo 7 consigna las conclusiones.

Trazabilidad

La palabra trazabilidad es un término relativamente moderno, se define como: “La capacidad de reproducir el historial de un producto con el fin de localizar rápidamente el origen de los problemas que puedan surgir en su elaboración o distribución y evitarlos a futuro” (Gran Larousse, 2004).

Gotel y Finkelstein [5] definen la trazabilidad como “la capacidad de describir y seguir la vida de un requisito tanto hacia adelante como hacia atrás, es decir, desde sus orígenes, su desarrollo y la especificación, hasta su posterior despliegue y uso, a través de todos los períodos de refinamiento en curso y la iteración en cualquiera de las fases.

Por su parte en [8] se define la trazabilidad como: El grado en que una relación puede establecerse entre dos o más productos del proceso de desarrollo, especialmente productos que tienen un sucesor predecesor o relación maestro-subordinado

En particular, la trazabilidad es una práctica que facilita el control de las necesidades por medio de vínculos de trazado entre diferentes artefactos. La trazabilidad puede ser vista como la habilidad para determinar cómo una pieza o fragmento de conocimiento afecta a otros. Esta práctica hace posible la búsqueda de como un cambio en una fase afecta artefactos y etapas en el ciclo de innovación, los cuales se verán reflejados en los detalles del diseño y las características de los productos y servicios [5, 6, 9].

En los procesos de innovación bajo enfoques de co-creación la trazabilidad es una práctica que puede mejorar la calidad de los productos o servicios, pues facilita el establecimiento de un conjunto de características y elementos para hacer el seguimiento de la vida de los artefactos durante el proceso de desarrollo. Esta práctica implica realizar actividades de validación y verificación para garantizar características relacionadas con la confiabilidad y la exactitud de los productos.

Una herramienta ampliamente empleada para mantener conexiones entre los artefactos generados es la matriz de trazabilidad. Estos arreglos permiten relacionar las necesidades (requisitos) de los usuarios con los diferentes artefactos confeccionados durante las fases del ciclo de innovación. Aunque estos elementos han sido generosamente utilizados para establecer relaciones entre requisitos y artefactos, durante las fases del desarrollo de software, su uso para relacionar elementos en las fases del ciclo de innovación ha sido poco explorado [10, 11, 12].

Existen varios tipos de trazabilidad, generalmente usados en las organizaciones que implementan este concepto en su proceso de elaboración de productos y/o servicios, estos son:

Trazabilidad Horizontal: En este tipo de trazabilidad se cuenta con la habilidad de relacionar las secciones/ componentes de la misma fase entre sí e identificar las dependencias que haya entre ellas, dependencia/ relación de un componente/clase con otros componentes/clases. Permite fácilmente detectar si hay conflictos entre las necesidades (requisitos), diseño, lógica de codificación o casos de prueba [22].

Trazabilidad Vertical: Este tipo de trazabilidad busca garantizar que todas las necesidades (o requerimientos) sobre el producto sean abordados (diseñados), y que todos los diseños se codifiquen y se prueben. El mecanismo de trazabilidad no sólo resalta las pruebas que se deben actualizar o repetir sino que también señala los documentos (análisis de riesgos, especificaciones y manuales del usuario, por ejemplo) que se deben revisar, la figura 2 representa un ejemplo de la trazabilidad vertical [22].

Cuando ocurren cambios en el desarrollo de un producto, la trazabilidad hace que sea relativamente más fácil evaluar y controlar el impacto que los cambios podrían tener en otras partes del proceso de innovación y desarrollo. Las relaciones entre los modelos, (enlaces de trazabilidad o trazas), deben ser desarrolladas y mantenidas en la medida que el sistema evoluciona. Conjuntos iniciales de enlaces

hacia adelante pueden ser construidos de forma automática a través de herramientas de diferente índole. Cualquier enlace puede ser construido manualmente o con herramientas que ayudan en la recuperación de los enlaces. Desafortunadamente, las herramientas existentes para la recuperación de las relaciones entre artefactos (modelos, planos, documentos, bosquejos, entre otros) están lejos de ser perfectos y requieren de ajustes y evaluación constante con el fin de garantizar la completitud y la coherencia entre los artefactos obtenidos en cada fase [23].

En resumen, la práctica de la trazabilidad aplicada en ambientes de co-creación posibilita la verificación de la transformación de los aportes o contribuciones de los colaboradores en elementos de modelo sucesores, así como el análisis y gestión del cambio en ellos, cotejando en cada fase su completitud y coherencia. Por otra parte, los enfoques existentes son limitados en expresividad dado el hecho de que las relaciones son principalmente los hipervínculos sin significados semánticos [24].

Propuesta de Modelo de Proceso de Innovación con Enfoque de Co-Creación

El modelo propuesto es completamente flexible y personalizable a las necesidades de cada organización. Se constituye en una guía y referente para las organizaciones que desean hacer un proceso sistemático de innovación con enfoque de co-creación. Esta propuesta provee un grado de detalle suficiente en cuanto a las actividades recomendadas en cada fase, los productos o artefactos que se generan, los actores involucrados, y las herramientas de co-creación adecuadas. De esta manera cada organización podrá seleccionar la configuración más conveniente de acuerdo a su contexto de trabajo y otros elementos que tome en consideración.

Para la descomposición lógica del modelo de innovación se proponen dos capas: una relacionada con las fases del proceso y otra dedicada a la recomendación de herramientas y participantes, enfatizando en el hecho de que el modelo es totalmente reconfigurable de acuerdo a las necesidades de cada empresa.

Capa de proceso

El proceso de innovación inicia con la detección de un problema u oportunidad en la empresa o su entorno hasta la explotación de los resultados de innovación y medición de impacto, pasando por varias etapas de selección y la propia ejecución de los proyectos de innovación [27]. El modelo de proceso propuesto está conformado por dos tipos de fases: secuenciales y transversales, tal como se evidencia en la figura 2.

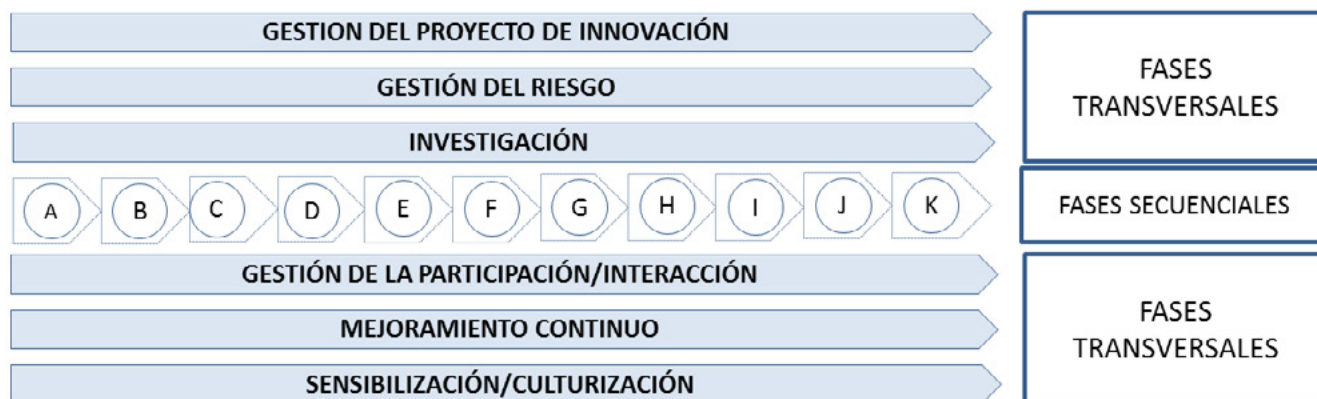


Figura 2. Descripción de las fases. Modelo de innovación propuesto

Capa de herramientas y agentes

Esta capa está compuesta por un módulo de actores (internos y externos) que pueden aportar en las diferentes fases del proceso; un segundo módulo es un catálogo de técnicas o herramientas existentes para hacer co-creación entre los participantes seleccionados.

La selección de quienes intervendrán en cada etapa de la innovación y las técnicas usadas queda en las manos del equipo encargado de coordinar el proceso, sin embargo, existen una serie de sugerencias a considerar.

Las fases secuenciales obedecen a un conjunto de pasos ordenados en el tiempo y cuyo inicio y fin no necesariamente coincide con el comienzo y terminación del proceso de innovación completo. De acuerdo a la dinámica seleccionada para construir valor, es posible que una fase secuencial se ejecute varias veces durante el proceso (pero siempre atendiendo al orden especificado).

Las fases transversales en cambio están activas durante todo el proceso de innovación, ya que contienen actividades que deben ser ejecutadas en cualquier momento y para apoyar cualquiera de las fases secuenciales definidas. Otras actividades de este tipo de procesos se repiten para refinar resultados, o alimentar bases de conocimiento de la organización. En estas fases el inicio y el fin si coincide con el principio y terminación del proceso de innovación completo.

Co-creación

Fase de revisión

La visión tradicional centrada en la empresa, en la que la empresa propone y sugiere cambios en sus productos y servicios está siendo debatida por consumidores cada vez más activos, conectados e

informados, ayudados por la combinación de tecnologías y la diversificación de sectores económicos. El valor ya no reside en el producto o en el servicio creado por la empresa y puesto a disposición de los consumidores. El valor se crea conjuntamente con la interacción entre el consumidor y la organización [28].

Aunque los límites geográficos todavía existen, se están desvaneciendo con rapidez, las reglas de la competencia están cambiando e imponiendo nuevos desafíos. El conocimiento de los consumidores es más amplio en cuanto a variedad de productos, servicios y medios de comunicación. Los consumidores ejercen influencia en todas las partes del sistema empresarial. Actualmente hay una participación más activa de los consumidores en procesos tanto de definición como de creación de valor.

Co-creación en el ciclo de innovación

El proceso de co-creación requiere de profundas interacciones entre el consumidor y la empresa para la creación conjunta de valor. Los consumidores esperan participar en cada una de las fases del ciclo, desde la concepción inicial en el diseño y elaboración, de hecho lo exigen. Los clientes requieren que los productos se ajusten perfectamente a sus necesidades, a sus deseos y esperan que la empresa se adapte, cambie, rediseñe y vuelva a imaginar el proyecto tantas veces como sea necesario, hasta conseguir la personalización. La interacción intensa con el cliente ya no es opcional dado las exigencias de los clientes. Esta manera de trabajar implica la utilización de herramientas y tecnologías que permitan una comunicación fluida entre el cliente y la organización.

El modelo de innovación adoptado posee un conjunto de etapas o fases bien diferenciadas, las cuales conducen a la obtención de un producto y/o servicio que entra en un proceso de elaboración, distribución e impacto sobre los potenciales clientes. En cada una de estas fases se posee un conocimiento o un conjunto de fragmentos de conocimiento que deben ser conectados e integrados lógicamente para obtener el producto o servicio que incluya las características innovadoras definidas por el grupo de trabajo. En cada una de estas fases los fragmentos de conocimiento deben pasar por un conjunto de filtros o fases de valoración que permitan reunir el conjunto de hechos para producir un modelo que sea soportando en la siguiente fase o etapa a partir de la fusión de ellos.

Trabajo colaborativo

El trabajo colaborativo se define como aquellos procesos intencionales de un grupo para alcanzar objetivos específicos con herramientas diseñadas para dar soporte y facilitar el trabajo. En el marco de una organización, el trabajo en grupo con soporte tecnológico se presenta como un conjunto de estrategias que propenden maximizar los resultados y minimizar la pérdida de tiempo e información a favor de los objetivos organizacionales [29].

Para realizar trabajo colaborativo se requiere de un conjunto de recursos y herramientas, de tal forma que clientes y otros agentes puedan compartir ideas, aportes y comentarios.

Marco de Trabajo

La motivación principal del presente trabajo consiste en apoyar la creación automática de enlaces entre diversos tipos de artefactos generados durante las diferentes fases del ciclo de innovación. La siguiente figura presenta el marco de trabajo definido para hacer seguimiento a las contribuciones generadas en las diferentes fases del ciclo de innovación y las relaciones entre sus diferentes módulos componentes.

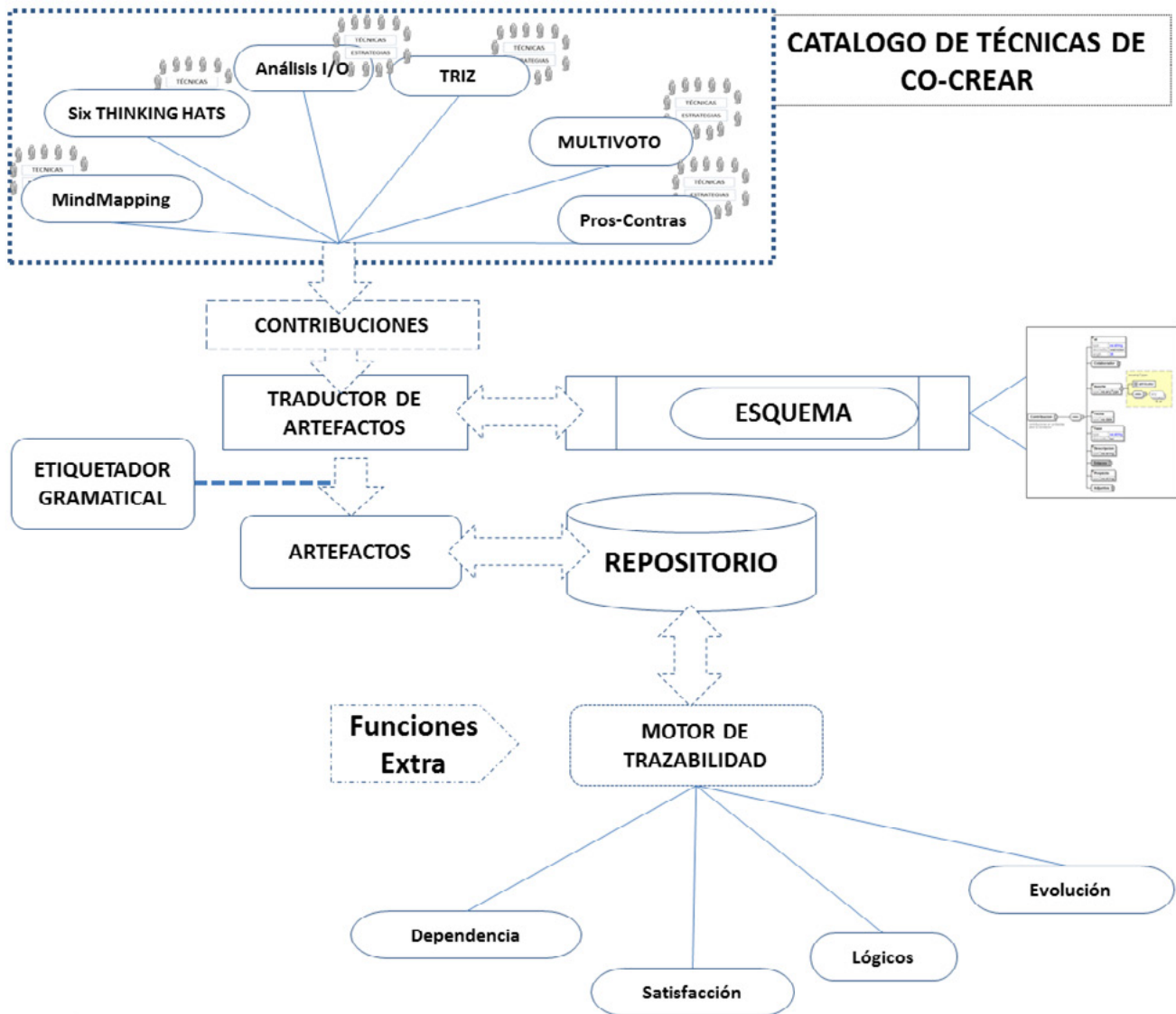


Figura 3. Marco de trabajo de trazabilidad basado en reglas.

Como se muestra en la figura 3 las contribuciones son generadas a partir de las diferentes técnicas de cocreación (normalmente definidas por el equipo de trabajo, según las etapas y tipos de participantes), las contribuciones pasan por una especie de filtro (traductor de artefactos), el cual está soportado por un esquema. El etiquetador es el encargado de colocar rótulos a cada una de las partes componentes de las contribuciones, generadas en las fases del ciclo. De esta forma las contribuciones se registran en un formato estructurado, lo que facilita el intercambio de información estructurada entre diferentes fases del ciclo y hace posible su recuperación y uso en cualquier etapa del ciclo de innovación.

El marco de trabajo se basa en un conjunto de reglas, las cuales permiten el establecimiento automático de las relaciones que facilitan el rastreo y la identificación de los artefactos creados durante el proceso de co-creación. La semántica de las relaciones de trazado puede tener diferentes estructuras semánticas, esto depende de los tipos de artefactos relacionados (origen-destino). Con el fin de facilitar la manipulación de los artefactos se asume que estos tienen una representación estándar.

Los artefactos generados en cada fase o nivel de abstracción en el ciclo de innovación sufren irremediablemente transformaciones. Esto significa que un conjunto de reglas de transformación descritas como un modelo en su lenguaje fuente podría ser transformado en otro modelo en su lenguaje destino. Así, una regla de transformación es una descripción de cómo uno o más constructores del lenguaje fuente pueden ser transformados en uno o más constructores del lenguaje destino. El objetivo de la traza es mantener un registro de las transformaciones de los artefactos a lo largo del ciclo de innovación aplicando técnicas de co-creación.

Representación

Con el fin de facilitar la comprensión del marco de trabajo, la tabla 2 presenta la manera en que se establecen los vínculos entre los artefactos generados en cada una de las fases y como estos pueden ser interpretados gráficamente. Como se observa se tienen relaciones en un mismo nivel de abstracción (Trazabilidad Horizontal) y relaciones entre diferentes niveles de abstracción (Trazabilidad Vertical). Es importante resaltar que al finalizar el grupo de trabajo debe consolidar uno o varios artefactos que reúnen las características principales del trabajo colaborativo a partir de un módulo compuesto por un sistema de recomendación basado en las valoraciones de los aportes del grupo de trabajo.

Tabla 2. Establecimiento de vínculos entre artefactos.

Dependencia	Controlan las dependencias entre objetos a nivel vertical, manejan las repercusiones de los cambios entre objetos dependientes.
Satisfacción	Relaciones entre las necesidades y cumplimiento en fases finales. Aseguran consistencia. Relación diseño Necesidades.
Lógicos	Representan las relaciones detrás de los objetos. Especifican las evoluciones en pasos evolutivos. Proveen una clara comprensión de la solución propuesta, facilitan mantenimiento y re-uso.
Evolución	Manejan relación entre elementos de entrada y salida. Permiten identificar el origen de los objetos. Ayudan a perfeccionar el seguimiento de la modificación e historia de los artefactos.

Conclusiones

La trazabilidad de contribuciones se presenta como una importante práctica en el proceso de innovación bajo enfoques de co-creación. Ayuda a establecer un marco conceptual de apoyo a los agentes que intervienen en el proceso, permite verificar aspectos relacionados con la consistencia y completitud a través de secuencias de traza. Al mismo tiempo, el marco de trabajo definido brinda la posibilidad de definir y seguir la vida de diversos artefactos en diferentes etapas con diferentes niveles de abstracción, permitiendo realizar análisis de la evolución de información en el ciclo de la innovación.

El establecimiento de enlaces entre artefactos en diferentes niveles de abstracción permite comprender la evolución del conocimiento en el ciclo de innovación y el impacto de los aportes de los agentes que participan en el proceso y como un cambio en una fase impacta en etapas posteriores. El modelo propuesto permite hacer recorridos hacia delante y hacia atrás para reconocer que artefactos dan cumplimiento a que necesidades planteadas durante las fases iniciales del ciclo.

Reconocimiento

Los autores, agradecen el apoyo de ARTICA (Alianza Regional en TIC (Aplicadas) y del proyecto Co-Creación que es soportado por la Universidad de Antioquia, Universidad Nacional, Sede Medellín, Universidad Pontificia Bolivariana, Universidad EAFIT, Universidad de Medellín y UNE EPM Telecomunicaciones.

Referencias

1. Cleland-Huang J, Berenbach B., Clark S., Settimi R., Romanova E. “Best Practices for Automated Traceability,” *Computer*, 2007, vol. 40, no. 6, pp. 9.
2. R. Oliveto, A. Marcus, J. Huffman Hayes, “Software Artefact Traceability: the Never-Ending Challenge.” *ICSM 2007*: 485-488
3. Object Management Group, “OMG Unified Modeling Language Specification”, Version 1.4, September 07, 2001, OMG 2001. Web Page:
4. Jacobson I., *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley, 1992.
5. Gotel O.C.Z., Finkelstein A.C.W. An analysis of the requirements traceability problem. In *First International Conference on Requirements Engineering (ICRE'94)*, pages 94–101. IEEE CS Press, 1994.
6. Hayes J.H., Dekhtyar A., Osborne J. Improving requirements tracing via information retrieval. In *RE*, page 138. IEEE Computer Society, 2003.
7. Cleland-Huang J., Chang C.K, Christensen M.J. Event-based traceability for managing evolutionary change. *IEEE TSE*, 29(9):796–810, 2003.
8. [*] Institute of Electrical and Electronics Engineers Inc., *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. New York, NY, USA: IEEE, 1991.
9. Palmer D. “Traceability”, in *Software Requirements Engineering*, R. Thayer and M. Dorfman, Eds., 2nd ed., Los Alamitos, California: IEEE Computer Society Press, 2000, pp. 412-422.
10. UML (2004). *Unified Modeling Language 2.0 Superstructure Specification*. Retrieved October, 2004 from <http://www.omg.org/cgi-bin/doc?ptc/2004-10-02>
11. Dömges R.m, Pohl K. Adapting traceability environments to project specific needs. *CACM*, 41(12):54–62, 1998.
12. Cleland-Huang J., Berenbach B., Clark S., Settimi R., Romanova E. “Best Practices for Automated Traceability,” *Computer*, 2007, vol. 40, no. 6, pp. 9.
13. Antoniol G., Canfora G., Casazza G., De Lucia, A. Identifying the Starting Impact Set of a Maintenance Request: a Case Study in *Proceedings of European Conference on Software Maintenance and Reengineering (CSMR'00)* (Zurich, Switzerland, February 29 - March 3, 2000), 227-230.

14. Marcus A, Maletic J.I. Recovering Documentation-to-Source-Code Traceability Links using Latent Semantic Indexing in Proceedings of 25th IEEE/ACM International Conference on Software Engineering (ICSE'03) (Portland, OR, May 3-10, 2003), 125-137.
15. Antoniol G., Canfora G., Casazza G., De Lucia A., “Information Retrieval Models for Recovering Traceability Links between Code and Documentation”, in Proceedings IEEE International Conference on Software Maintenance (ICSM'00), San Jose, CA, October 11-14 2000, pp. 40-51.
16. Antoniol G., Canfora G., Casazza G., De Lucia, A., Merlo E. “Recovering Traceability Links between Code and Documentation”, IEEE Transactions on Software Engineering, 28, 10, October 2002, pp. 970 - 983.
17. Antoniol G., Canfora G., De Lucia A., Merlo E. “Recovering Code to Documentation Links in OO Systems”, in Proceedings 6th IEEE Working Conference on Reverse Engineering (WCRE'99), Atlanta, GA, October 6-8 1999, pp. 136-144.
18. Pinheiro F, Goguen J. “An Object-Oriented Tool for Tracing Requirements”, IEEE Software, 13, 2, 1996, pp. 52-64.
19. Pohl K. “PRO-ART: Enabling requirements pre-traceability”, in Proceedings International Conference on Requirements Engineering, Colorado Springs, Colorado, 1996, pp. 76-85.
20. Mäder Patrick, Gotel Orlena. Ready-to-Use Traceability on Evolving Projects, Software and Systems Traceability, 2012, pp 173-194.
21. Niosi J. (1999). Fourth-generation R&D: From linear models to flexible innovation. Journal of business research 45, pp.111-117.
22. Hidaka S., Hu Z., Inaba K., Kato H., Nakano K. “GROundTram: An integrated framework for developing wellbehaved bidirectional model transformations” in ASE'11, 2011, pp. 480-483.
23. Arlow J, Neustadt I. (Junio 2006). UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design (2nd Edition). ANAYA - Addison-Wesley Object Technology Series.
24. Chesbrough H. Open Business Models: How to Thrive in the New Innovation Landscape. Harvard Business School Press, Boston, MA, 2006.
25. Hobday M. (2005). Firm-Level Innovation Models: Perspectives on Research in Developed and Developing Countries. University of Illinois at Urbana-Champaign's Academy for Entrepreneurial Leadership Historical Research Reference in Entrepreneurship.

26. Fields E.R., Amaldi P, Tassi A. (2005). Representing collaborative work: The airport as common information space. *Cognition, Technology and Work*, 7, 119-133.
27. Clag- Clúster Audiovisual Galego. (2012). Gestión de la innovación en el sector audiovisual Retrieved Marzo, 2012, from:
<http://www.clag.es/innovacion/espanol.html>
28. Prahalad C, Ramaswamy V. 2004a. co-creation experiences: the next practice in value. *journal of interactive marketing*.
29. Tapiador A., Fumero A., Salvachu'a J., Aguirre S.A. Web Collaboration Architecture, in the Proceedings of the 2nd IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2006), Atlanta, Georgia, USA.



Propuesta integral de manejo de requerimientos en proyectos de explotación de información

Pollo-Cattaneo, M.F.^{1,2}; Mansilla, D.²; Vegega, C.²; Pytel, P.^{2,4}; Pesado, P.³; García-Martínez, R.⁴; P. Britos, P.⁵

¹ Programa de Doctorado en Ciencias Informáticas. Facultad de Informática. Universidad Nacional de La Plata. Argentina. Correo electrónico: fpollo@posgrado.frba.utn.edu.ar

² Grupo de Estudio en Metodologías de Ingeniería de Software. Facultad Regional Buenos Aires. Universidad Tecnológica Nacional. Argentina.

³ Instituto de Investigaciones en Informática LIDI. Facultad de Informática. Universidad Nacional de La Plata. Argentina.

⁴ Grupo Investigación en Sistemas de Información. Departamento Desarrollo Productivo y Tecnológico. Universidad Nacional de Lanús. Argentina. Correo electrónico: rgarcia@unla.edu.ar

⁵ Grupo de Investigación en Explotación de Información. Laboratorio de Informática Aplicada. Universidad Nacional de Río Negro. Argentina. Correo electrónico: paobritos@gmail.com

Resumen

Los proyectos de Explotación de Información tienen como objetivo la aplicación de un proceso que convierta los datos disponibles en conocimiento útil para la toma de decisiones. Los requerimientos en este tipo de proyectos son diferentes a los que se presentan en proyectos tradicionales de construcción de software, dado que estos proyectos se enfocan en construir un producto software y por lo tanto, los procesos asociados a la obtención de estos requerimientos no pueden utilizarse en proyectos de Explotación de Información. En este contexto, se presenta un modelo integral que permita la gestión de requisitos en este tipo de proyectos, de forma de sentar las bases para el desarrollo de una ingeniería de requisitos propia de estos proyectos.

Palabras clave: explotación de información, gestión de requerimientos, Ingeniería de Requerimientos, metodología, proceso.

Introducción

La Explotación de Información (EdI) es la sub-disciplina de la Informática que consiste en la extracción de conocimiento no trivial que reside de forma implícita en los datos disponibles que se encuentran en diferentes fuentes de información [1]. Éste conocimiento es previamente desconocido y puede resultar útil para la toma de decisiones de gestión y generación de planes estratégicos en las organizaciones [2].

Existen metodologías que se aplican al desarrollo de proyectos de EdI, entre las que se destacan CRISP-DM [3], P3TQ [4] y SEMMA [5]. Estas metodologías poseen un buen grado de madurez respecto al desarrollo de un proyecto, pero descuidan aquellos aspectos relacionados a la gestión del proyecto y al contexto organizacional en el cual se desarrolla dicho proyecto, fallando al momento de educir todos los conceptos necesarios durante el conocimiento del negocio [6].

Un ejemplo de estas deficiencias se puede encontrar en la metodología CRISP-DM, donde la primera fase busca identificar y comprender los aspectos del negocio relacionados al proyecto que se llevará acabo pero no define técnicas, métodos o herramientas para obtener esta información ni realizar su documentación.

En este contexto, el presente trabajo tiene como objetivo proponer un modelo integral que permita la gestión de requisitos en proyectos de EdI, de forma de sentar las bases para el desarrollo de una ingeniería de requisitos para este tipo de proyectos. Para ello, en primera instancia, se define el problema detectado en el ámbito de los proyectos de EdI (sección 2) y se presenta una propuesta de solución para este problema (sección 3). Se detalla una prueba de concepto (sección 4), y posteriormente se presentan las conclusiones obtenidas y futuras líneas de trabajo (sección 5).

Planteamiento del problema

Los proyectos de EdI necesitan contar con requerimientos claros, completos y estables para completarse en forma exitosa [7], pero sus requerimientos son diferentes a los requerimientos de los proyectos software tradicionales, dado que un proyecto de EdI no busca la construcción de un producto software sino la aplicación de un proceso que convierta los datos disponibles en conocimiento. Al no tener este objetivo, los requerimientos no están enfocados en definir las funcionalidades y restricciones que deberá cumplir el producto software, como es el caso de la Ingeniería en Software [8] y la Ingeniería del Conocimiento [9].

Al comienzo de un proyecto de EdI se deben identificar los objetivos del proyecto que describen la necesidad del cliente y se encuentran relacionados con las metas estratégicas y tácticas del negocio [10]. Para poder comprender estos aspectos del proyecto, en este tipo de proyectos se tiene una dificultad adicional que consiste en no manejar en forma correcta el vocabulario del negocio utilizado por los miembros de la organización. Por otro lado, una vez identificados los objetivos del proyecto es necesario realizar un reconocimiento de las fuentes de información disponibles en la organización. A partir del análisis de los objetivos y las fuentes de información, es posible delimitar el alcance del proyecto y así obtener un conjunto de objetivos particulares que podrán ser resueltos a través de la aplicación de procesos de EdI que utilizan a los algoritmos de minería de datos [11]. De esta manera se podrá solucionar el problema de negocio que dio origen al proyecto.

En este contexto, las metodologías completas provenientes de la Ingeniería en Software e Ingeniería del Conocimiento no son útiles en los proyectos de EdI ya que no se ocupan de los aspectos prácticos de la especificación de requisitos propia de este tipo de proyectos [12].

A partir de esta problemática, se ha detectado la necesidad de ofrecer un modelo de proceso que permita: (a) realizar la elicitación de requisitos para identificar las principales necesidades del cliente, sus expectativas, restricciones y los principales repositorios de datos que son necesarios para realizar el proyecto y (b) formalizar los requisitos elicitados indicando la forma en que deben ser documentados.

Solución propuesta

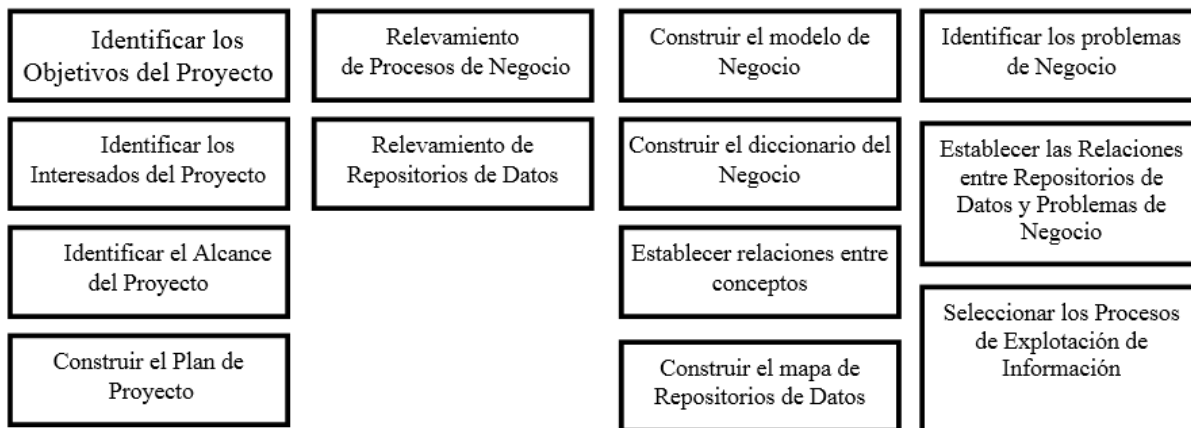
La solución propuesta a la problemática planteada en la sección anterior consiste en un modelo de procesos que se divide en cuatro fases principales: Definición del Proyecto, Educación del Negocio, Conceptualización del Negocio e Identificación de Procesos de Explotación de Información. Cada fase tiene definido un conjunto de actividades y un conjunto de procesos de formalización asociados a dichas actividades.

La figura 1 muestra el mapa conceptual completo del modelo de procesos propuesto, dividido en los diferentes niveles (Fase, Actividad y Proceso de Formalización).

Fases



Actividades de cada Fase



Procesos de Formalización asociados a las Fases

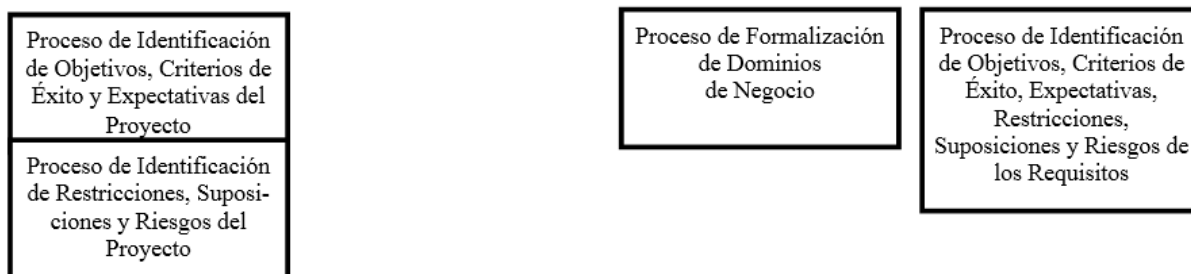


Figura 1. Mapa Conceptual del Modelo de Procesos Propuesto.

La fase de Definición del Proyecto tiene por objetivo definir el alcance del proyecto, los interesados y los objetivos que se deben alcanzar. En esta fase se realiza la planificación de las actividades de educación de requisitos, que servirá como plan para la ejecución de las diferentes actividades del proceso.

La fase de Educción del Negocio tiene por objetivo comprender el idioma utilizado en el negocio, descubrir las palabras específicas del mismo y cuál es el significado que el negocio le da a esas palabras específicas.

La fase de Conceptualización del Negocio tiene por objetivo definir el negocio en términos de conceptos utilizados en el mismo, vocabulario y repositorios donde se almacena la información de los diferentes procesos del negocio.

La fase de Identificación de Procesos de Explotación de Información define los procesos de minería de datos que se pueden utilizar para resolver los problemas identificados en el proceso de negocio.

A su vez, en este modelo de procesos participan diferentes personas que cumplen roles específicos. Dichos roles, junto con sus responsabilidades son indicados en la tabla 1.

Tabla 1. Roles del Proceso de Gestión de Requerimientos.

Rol	Responsabilidades
Líder de Proyecto	Gestionar las acciones para que se lleven a cabo las actividades del proyecto y se cumplan los compromisos del proyecto.
Analista Funcional	Relevar y analizar los diferentes procesos del negocio.
Especialista de Datos	Relevar y analizar las diferentes fuentes de información. Debe tener las capacidades técnicas necesarias para recuperar datos de dichas fuentes.
Analista en Explotación de Información	Establecer relaciones entre datos y relacionar procesos de explotación de información con los problemas del negocio detectados. Aplicar algoritmos de minería de datos sobre los datos relacionados al proceso de negocio para obtener el conocimiento que permita resolver los problemas del negocio detectados.

Fase de Definición del Proyecto

Durante esta fase se realizan las tareas asociadas a la planificación del proyecto y a establecer el alcance y las personas interesadas en el mismo. Estas actividades son la base de todo proyecto [12]. La figura 2 muestra las actividades y los procesos de formalización asociados a esta fase.

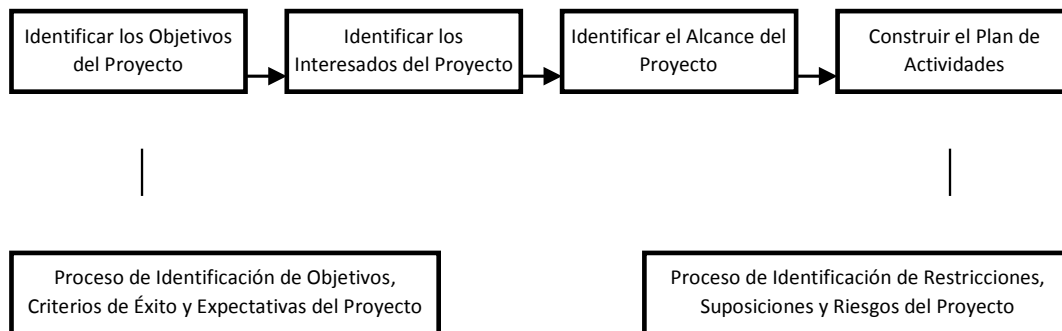


Figura 2. Actividades y procesos de formalización de la Fase de Definición del Proyecto.

El Líder de Proyecto será el responsable de las diferentes actividades de esta fase que comienza con la identificación de los objetivos del proyecto, utilizando el Proceso de “Identificación de Objetivos, Criterios de Éxito y Expectativas del Proyecto”. Con los objetivos definidos, se identifica la lista de interesados del proyecto (actividad “Identificar los Interesados del Proyecto”) en la que figuran quienes permiten definir el alcance del proyecto de EdI, actividad en la que también participa el Analista Funcional. Por último, con el objetivo, alcance e interesados del proyecto definidos, el Líder de Proyecto utiliza el Proceso de “Identificación de Restricciones, Suposiciones y Riesgos del Proyecto” para construir el Plan de Actividades, que es el documento que contiene las tareas que se deben llevar a cabo durante el proceso, define el alcance del proyecto, el equipo del proyecto y los mecanismos de seguimiento del mismo (actividad de “Construir Plan de Actividades”). Con el plan construido comienzan las actividades asociadas a las siguientes fases.

Fase de Educación del Negocio

Durante esta fase se realizan las tareas que permiten relevar (elicitar) los diferentes procesos de negocio. El Analista Funcional será el responsable de llevar a cabo las actividades definidas para esta Fase. La figura 3 muestra las actividades que la componen.

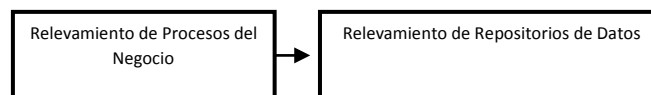


Figura 3. Actividades de la Fase de Educación del Negocio.

El Analista Funcional, en la actividad de “Relevamiento de Procesos del Negocio”, recopilará la información asociada a las diferentes actividades del negocio para identificar qué datos utilizan y cómo funcionan estos procesos, mediante la utilización de técnicas tradicionales de educación, como las que se presentan en [8] y [14]. Esta información documentada será utilizada como referencia en la siguiente fase. Con la ayuda de un Especialista de Datos, se deberán identificar los diferentes repositorios de datos utilizados y las fuentes que generan esos datos (actividad de “Relevamiento de Repositorios de Datos”) que pueden, o no, estar informatizados.

Fase de Conceptualización del Negocio

En la fase de conceptualización se construye el modelo de negocio que se utiliza como base del Proyecto de EdI. El Analista Funcional será el responsable de las actividades de esta fase que incluyen las actividades definidas en la figura 4.

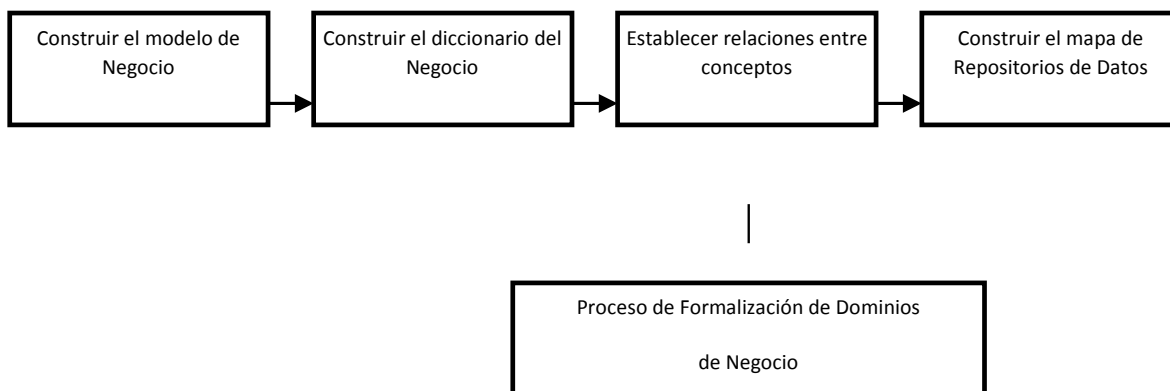


Figura 4. Actividades y Procesos de Formalización de la Fase de Conceptualización del Negocio.

Con la información obtenida en la “Educción del Negocio”, durante la actividad de “Construir el modelo de Negocio”, el Analista Funcional modela los casos de uso de negocio asociados al alcance del proyecto. A partir de este modelo, como parte de la actividad “Construir el Diccionario del Negocio”, se analiza el vocabulario utilizado en el negocio y se construye el diccionario de términos asociados a los diferentes procesos de negocio. El Analista de Explotación de Información, en la actividad de “Establecer las Relaciones entre Conceptos”, sigue el Proceso de Formalización de Dominios de Negocio, establecido en [15] para generar el Diagrama de Entidad-Relación (DER) asociado a los conceptos. Por último, el rol de Especialista de Datos será el de ser el responsable de relevar las diferentes fuentes de información asociadas a los procesos modelados, las que permitirán establecer las relaciones entre los Casos de Uso identificados y los repositorios de datos existentes. Esta relación se establece en la actividad de “Construir el Mapa de Repositorio de Datos”.

Fase de Identificación de Procesos de Explotación de Información

Esta fase es la conclusión del trabajo realizado por el equipo de proyecto y define como resultado final los Procesos de Explotación de Información a utilizar en el proyecto. La figura 5 representa las actividades y los procesos de formalización asociados a esta fase.

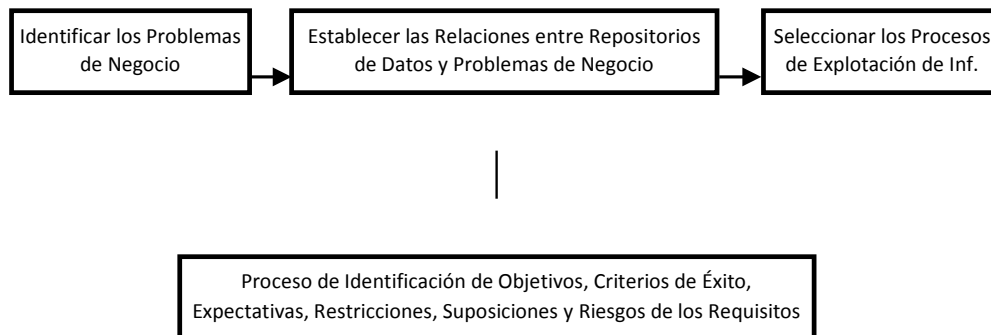


Figura 5. Actividades y Procesos de Formalización de la Fase de Identificación de Procesos de Explotación de Información.

El Analista Funcional se encargará de construir una lista que contiene los problemas que el negocio desea resolver (actividad “Identificar los Problemas de Negocio”), y en conjunto con el Analista de Explotación de Información establecerán las relaciones existentes entre los problemas identificados y los repositorios que contienen información útil para la resolución de los problemas (actividad “Establecer las Relaciones entre Repositorios de Datos y Problemas de Negocio) utilizando el Proceso de Proceso de “Identificación de Objetivos, Criterios de Éxito, Expectativas, Restricciones, Suposiciones y Riesgos de los Requisitos”. Con esta información, y utilizando técnicas propuestas en [16, 17] se podrá/n seleccionar el o los procesos de Explotación de Información que utilizará el proyecto (actividad “Seleccionar Procesos de Explotación de Información”).

Caso de estudio

En esta sección se presentan los resultados de aplicar el modelo de procesos propuesto en un caso de estudio denominado “Caracterización de la Educación Rural en Argentina a través de Minería de Datos con Sistemas Inteligentes” [20]. Este caso de estudio tiene como objetivo la aplicación de la metodología CRISP-DM para encontrar patrones referidos a la educación rural en Argentina entre los 2000 y 2004.

En la primera fase del modelo Definición del Proyecto, correspondiente a establecer el alcance y las personas interesadas en el proyecto, se pudieron definir los objetivos generales del proyecto que se muestran en la tabla 2 y los objetivos particulares que se identifican en la tabla 3. Asimismo, se pudieron detectar riesgos referidos a la información disponible para llevar a cabo el proyecto, y se pudieron definir los planes de contingencia correspondientes a estos riesgos, que se muestran en las tablas 4 y 5 respectivamente.

Tabla 2. Planilla de Objetivos del Proyecto.

Objetivos del proyecto			
Analista	María Florencia Pollo	Fecha	08/09/2013
ID #	COMPORTAMIENTO-EDUCACIÓN-RURAL		
ID Objetivo	Descripción	Referencia	
1	Caracterización de entornos rurales para alumnos de nivel superior no universitario.	Entrevista	
2	Relación entre ingresos, egresos, carreras, provincias y tipo de formación de los alumnos de nivel no universitario en entornos rurales.	Entrevista	

En la segunda fase del modelo Educación del Negocio, en la cual se relevan (elicitan) los diferentes procesos del negocio y los repositorios de datos, se identificaron dos procesos básicos y de alto nivel que el negocio realiza: (a) evaluación de la calidad educativa e (b) investigación educativa. El proceso (a) permite detectar deficiencias y supervisar si se cumplen los patrones de calidad establecidos por el Ministerio de Educación, mientras que el proceso (b) permite analizar las características de la educación en general, de forma de poder detectar problemas, mejorar la calidad e implantar mejoras. Por otro lado, se detectó una fuente de información que se identifica en la tabla 6.

En la tercera fase del modelo Conceptualización del Negocio, que permite construir el modelo de negocio e identificar el vocabulario común, se pudieron establecer las descripciones del vocabulario que se utilizará en el proyecto, tal como se muestra en las tabla 7a y 7b y los atributos referidos a los objetivos del proyecto, que se puede visualizar en la tabla 8.

Tabla 3. Planilla de Objetivos del Requisito.

Objetivos del requisito			
Analista	María Florencia Pollo	Fecha	08/09/2013
ID #	COMPORTAMIENTO-EDUCACIÓN-RURAL		
ID Objetivo del Requisito	Descripción	ID Objetivo del Proyecto	Referencia
1	Buscar la caracterización en entornos rurales en la nación para los alumnos de nivel superior no universitario en el año 2004.	1	Base de Datos
2	Buscar la caracterización en entornos rurales en la nación para los alumnos de nivel superior no universitario en el año 2002.	1	Base de Datos
3	Buscar la caracterización en entornos rurales en la nación para los alumnos de nivel superior no universitario en el año 2000.	1	Base de Datos
4	Buscar la relación entre ingresados, egresados, provincias, carreras y tipo de formación para los alumnos de nivel superior no universitario en el año 2004.	2	Base de Datos
5	Buscar la relación entre ingresados, egresados, provincias, carreras y tipo de formación para los alumnos de nivel superior no universitario en el año 2002.	2	Base de Datos
6	Buscar la relación entre ingresados, egresados, provincias, carreras y tipo de formación para los alumnos de nivel superior no universitario en el año 2000.	2	Base de Datos

Tabla 4. Planilla de Riesgos del Proyecto.

Riesgos del proyecto			
Analista	María Florencia Pollo	Fecha	08/09/2013
ID #	COMPORTAMIENTO-EDUCACIÓN-RURAL		
ID Riesgo	Descripción	Referencia	
1	La información disponible en las fuentes de datos no se encuentra bien estructurada y hay poca información para las zonas rurales estudiadas.	Base de Datos	

Tabla 5. Planilla de Plan de Contingencia.

Plan de contingencia			
Analista	María Florencia Pollo	Fecha	08/09/2013
ID #	COMPORTAMIENTO-EDUCACIÓN-RURAL		
ID Acción	Acción	Referencia	
1	Realizar entrevistas con el experto del negocio, de forma de entender la información contenida en la fuente de datos y realizar una interpretación adecuada.	Entrevista	

Tabla 6. Planilla de Fuente de Información para los Requerimientos.

Fuente de información para los requerimientos				
Analista	María Florencia Pollo	Fecha	08/09/2013	
ID #	COMPORTAMIENTO-EDUCACIÓN-RURAL			
Origen	Tipo	Descripción	Responsable	Referencia
Base de datos de educación nacional	Base de datos	Información vinculada a la educación nacional entre los años 2000 y 2004.	Ministerio de Educación, Ciencia y Tecnología.	Entrevista

Tabla 7a. Planilla de definiciones, acrónimos y abreviaturas.

Definiciones, acrónimos y abreviaturas			
Analista	María Florencia Pollo	Fecha	08/09/2013
ID #	COMPORTAMIENTO-EDUCACIÓN-RURAL		
Término	Descripción	Tipo	Referencia
ALU1	Cantidad de alumnos que ingresan en primer grado primario, primer grado EGB, primer año medio o primer año polimodal.	Valor	Entrevista
Ámbito	Características demográficas del espacio socio-geográfico donde se encuentra la unidad educativa, caracterizado por la cantidad de habitantes.	Concepto	Entrevista

Tabla 7b. Planilla de definiciones, acrónimos y abreviaturas.

Definiciones, acrónimos y abreviaturas			
Analista	María Florencia Pollo	Fecha	08/09/2013
ID #	COMPORTAMIENTO-EDUCACIÓN-RURAL		
Término	Descripción	Tipo	Referencia
ÁMBITO	Indica la zona en donde se encuentra el establecimiento educativo. Los valores posibles son: Urbano o Rural.	Valor	Entrevista
CARRERA	Nombre de la carrera.	Concepto	Entrevista
Carrera	Denominación que reciben los estudios que se realizan durante un determinado número de años, al final de los cuales se obtiene una titulación académica.	Concepto	Entrevista
Carrera de Grado	Carrera que se dicta en los institutos de nivel superior no universitarios.	Atributo	Entrevista
ED17	Número de matrícula de 17 años de edad.	Valor	Entrevista

ED18	Número de matrícula de 18 años de edad.	Valor	Entrevista
ED19	Número de matrícula de 19 años de edad.	Valor	Entrevista
ED20	Número de matrícula de 20 años de edad.	Valor	Entrevista
ED21	Número de matrícula de 21 años de edad.	Valor	Entrevista
ED22	Número de matrícula de 22 años de edad.	Valor	Entrevista
ED23	Número de matrícula de 23 años de edad.	Valor	Entrevista
ED24	Número de matrícula de 24 años de edad.	Valor	Entrevista
ED25	Número de matrícula de 25 años de edad.	Valor	Entrevista
ED26	Número de matrícula de 26 años de edad.	Valor	Entrevista
ED27	Número de matrícula de 27 años de edad.	Valor	Entrevista
ED28	Número de matrícula de 28 años de edad.	Valor	Entrevista
ED29	Número de matrícula de 29 años de edad.	Valor	Entrevista
ED30-34	Número de matrícula de 30 a 34 años de edad.	Valor	Entrevista
ED35_39	Número de matrícula de 35 a 39 años de edad.	Valor	Entrevista
ED40	Número de matrícula de 40 años de edad.	Valor	Entrevista
Egresado	Alumno que ha acreditado satisfactoriamente todas las asignaturas y actividades que integran el plan de estudios de un determinado nivel educativo.	Concepto	Entrevista
EGRESADOS	Cantidad de egresados de un establecimiento educativo.	Valor	Entrevista

Establecimiento Educativo	Unidad institucional donde se organiza la oferta educativa, cuya creación o autorización se registra bajo un acto administrativo (ley, decreto, resolución o disposición).	Concepto	Entrevista
ID_RA	Identificación del establecimiento educativo.	Valor	Entrevista
Matrícula	Cantidad de alumnos registrados en una unidad educativa a una fecha determinada.	Concepto	Entrevista
MATRÍCULA	Cantidad de alumnos matriculados totales por establecimiento educativo.	Valor	Entrevista
NIVEL	Nivel de Enseñanza	Valor	Entrevista
Nivel de Enseñanza	Tiene por objeto profundizar el conocimiento en un conjunto de saberes, habilidades y valores según modalidades y orientaciones científicas, técnicas, humanísticas, etc.	Concepto	Entrevista
REP1	Cantidad de alumnos repitentes de primer grado primario, primer grado EGB, primer año medio o polimodal.	Valor	Entrevista
PROVINCIA	Nombre de la jurisdicción.	Valor	Entrevista
Rural Aglomerado	Núcleo poblacional entre 500 y 2000 habitantes.	Atributo	Entrevista
Rural Disperso	Núcleo poblacional de menos de 500 habitantes o en campo abierto.	Atributo	Entrevista
S.N.U.	Marca que indica si el establecimiento ofrece el nivel de enseñanza superior no universitario.	Valor	Entrevista
TIPOFORMAC	Tipo de Formación.	Valor	Entrevista
Urbano	Núcleo poblacional de 2000 o más habitantes.	Atributo	Entrevista

Tabla 8. Planilla de Atributos relacionados con los Requisitos.

Atributos relacionados con los requisitos				
Analista	María Florencia Pollo		Fecha	08/09/2013
ID #	COMPORTAMIENTO-EDUCACIÓN-RURAL			
Atributo	Origen	Tipo de Atributo	Objetivo del Requisito	Referencia
ID_RA	Base de datos	Numérico	1,2,3,4,5,6	Entrevista
ED17	Base de datos	Numérico	1,2,3	Entrevista
ED18	Base de datos	Numérico	1,2,3	Entrevista
ED19	Base de datos	Numérico	1,2,3	Entrevista
ED20	Base de datos	Numérico	1,2,3	Entrevista
ED21	Base de datos	Numérico	1,2,3	Entrevista
ED22	Base de datos	Numérico	1,2,3	Entrevista
ED23	Base de datos	Numérico	1,2,3	Entrevista
ED24	Base de datos	Numérico	1,2,3	Entrevista
ED25	Base de datos	Numérico	1,2,3	Entrevista
ED26	Base de datos	Numérico	1,2,3	Entrevista
ED27	Base de datos	Numérico	1,2,3	Entrevista
ED28	Base de datos	Numérico	1,2,3	Entrevista
ED29	Base de datos	Numérico	1,2,3	Entrevista
ED30_34	Base de datos	Numérico	1,2,3	Entrevista
ED35_39	Base de datos	Numérico	1,2,3	Entrevista
ED40	Base de datos	Numérico	1,2,3	Entrevista
EGRESADOS	Base de datos	Numérico	1,2,3,4,5,6	Entrevista

AMBITO	Base de datos	Alfabético	1,2,3,4,5,6	Entrevista
S.N.U.	Base de datos	Booleano	1,2,3,4,5,6	Entrevista
ALU1	Base de datos	Numérico	1,2,3	Entrevista
REP1	Base de datos	Numérico	1,2,3	Entrevista
NIVEL	Base de datos	Alfabético	1,2,3,4,5,6	Entrevista
PROVINCIA	Base de datos	Alfabético	4,5,6	Entrevista
CARRERA	Base de datos	Alfabético	4,5,6	Entrevista
TIPOFORMAC	Base de datos	Alfabético	4,5,6	Entrevista
MATRÍCULA	Base de datos	Numérico	4,5,6	Entrevista

Por último, en la cuarta fase del modelo Identificación de Procesos de Explotación de Información, se pudieron detectar las restricciones correspondientes a los objetivos particulares del proyecto que se muestran en la tabla 9 y a partir de estos objetivos se analizan los procesos de explotación de información acordes a cada objetivo a cumplir, que se muestra en la tabla 10.

Tabla 9. Planilla de Restricciones del Requisito.

Restricciones del requisito					
Analista		María Florencia Pollo		Fecha	08/09/2013
ID #		COMPORTAMIENTO-EDUCACIÓN-RURAL			
ID Restricción	Tipo	Descripción	Objetivo del Requisito	Referencia	
1	Datos	El estudio realizado se encuentra acotado al año lectivo 2004.	1,4	Base de Datos	
2	Datos	El estudio realizado se encuentra acotado al año lectivo 2002.	2,5	Base de Datos	
2	Datos	El estudio realizado se encuentra acotado al año lectivo 2000.	3,6	Base de Datos	

Tabla 10. Planilla de Procesos de Explotación de Información.

Procesos de explotación de la información			
Analista	María Florencia Pollo	Fecha	08/09/2013
ID #	COMPORTAMIENTO-EDUCACIÓN-RURAL		
ID Obj. Req.	Descripción del Proceso		
1	Descubrimiento de reglas de pertenencia a grupos.		
2	Descubrimiento de reglas de pertenencia a grupos.		
3	Descubrimiento de reglas de pertenencia a grupos.		
4	Descubrimiento de reglas de comportamiento usando “provincia” como atributo objetivo. Descubrimiento de reglas de comportamiento usando “carrera” como atributo objetivo		
5	Descubrimiento de reglas de comportamiento usando “provincia” como atributo objetivo. Descubrimiento de reglas de comportamiento usando “carrera” como atributo objetivo.		
6	Descubrimiento de reglas de comportamiento usando “provincia” como atributo objetivo. Descubrimiento de reglas de comportamiento usando “carrera” como atributo objetivo.		

Una vez definidos los procesos correspondientes se continúa con el proyecto de Explotación de Información, pudiendo utilizar cualquier metodología de explotación de información existente.

Conclusiones

A partir del modelo de procesos propuesto se puede realizar una gestión integral de los requisitos en proyectos de Explotación de Información, mejorando de esta forma, las metodologías existentes que descuidan estos aspectos. El proceso contempla cuatro fases, en las cuales se gestiona de forma completa el alcance y los interesados del proyecto, así como se modelan los procesos y datos del negocio, para poder proponer los procesos de Explotación de Información que solucionan los problemas de negocio encontrados.


El proceso propuesto fue aplicado a un caso de estudio que utilizaba la metodología CRISP-DM para resolver un problema de negocio. Mediante la aplicación de este proceso se pudieron obtener los requerimientos del cliente y los objetivos del proyecto relacionados a estos requerimientos. Mediante estos objetivos se pudieron identificar los procesos de Explotación de Información que se deben aplicar para cumplirlos en forma exitosa.

Como futura de línea de trabajo, se propone buscar otros casos en donde no se aplique la metodología CRISP-DM de forma de obtener una validación completa del proceso.

Referencias

1. Schiefer J., Jeng J., Kapoor S & Chowdhary P. (2004). Process Information Factory: A Data Management Approach for Enhancing Business Process Intelligence. Proceedings 2004 IEEE International Conference on E-Commerce Technology. Pág. 162-169.
2. Thomsen, E. (2003). BI's Promised Land. *Intelligent Enterprise*, 6(4): 21-25.
3. Chapman P., Clinton J., Kerber R., Khabaza T., Reinartz T., Shearer C. & Wirth, R. (2000). CRISP-DM 1.0 Step-by-step Data Mining Guide. <http://tinyurl.com/crispdm>. Último acceso Enero de 2013.
4. Pyle, D. (2003). *Business Modeling and Business intelligence*. Morgan Kaufmann Publishers.
5. SAS (2008). SAS Enterprise Miner: SEMMA. <http://tinyurl.com/semmaSAS>. Último acceso Enero de 2013.
6. Vanrell J., Bertone R. & García-Martínez R. (2010). Modelo de Proceso de Operación para Proyectos de Explotación de Información. *Anales del XVI Congreso Argentino de Ciencias de la Computación*, Pág. 674-682. ISBN 978-950-9474-49-9.
7. Pollo-Cattaneo, M., García-Martínez, R., Britos, P., Pesado, P., Bertone, R., Rodríguez, D., Merlino, H., Pytel, P., Vanrell, J. (2012). Elementos para una Ingeniería de Explotación de Información. *Proyecciones* 10(1): 67-84. ISSN 1667-8400.
8. Wiegers, K. (2003). *Software Requirements*. Microsoft Press.
9. García Martínez, R. & Britos, P. (2004). *Ingeniería de Sistemas Expertos*. Editorial Nueva Librería.
10. Britos, P., Dieste, O., García-Martínez, R. (2008). Requirements Elicitation in Data Mining for Business Intelligence Projects. In *Advances in Information Systems Research, Education and Practice*. David Avison, George M. Kasper, Barbara Pernici, Isabel Ramos, Dewald Roode Eds. (Boston: Springer), IFIP International Federation for Information, 274: 139-150.
11. García-Martínez, R., Britos, P., Pollo-Cattaneo, F., Rodríguez, D., Pytel, P. (2011). Information Mining Processes Based on Intelligent Systems. *Proceedings of II International Congress on Computer Science and Informatics (INFONOR-CHILE 2011)*. Pág. 87-94. ISBN 978-956-7701-03-2.

12. García-Martínez, R., Britos, P., Pesado, P., Bertone, R., Pollo-Cattaneo, F., Rodríguez, D., Pytel, P., Vanrell, J. (2011). Towards an Information Mining Engineering. En Software Engineering, Methods, Modeling and Teaching. Sello Editorial Universidad de Medellín. ISBN 978-958-8692-32-6. Páginas 83-99.
13. William, R. 1996. A Guide To The Project Management Body Of Knowledge. PMI Pub-lishing.
14. Sommerville, Ian, Y Peter Sawyer. 1997. Requirements Engineering: A Good Practice Guide. Chichester, England: John Wiley & Sons.
15. Vegega, C., Pytel, P., Ramón, H., Rodríguez, D., Pollo-Cattaneo, F., Britos, P., García-Martínez, R. (2012). Formalización de Dominios de Negocio para Proyectos de Explotación de Información basada en Técnicas de Ingeniería del Conocimiento. Proceedings del XVIII Congreso Argentino de Ciencias de la Computación. Pag. 1049-1058. ISBN 978-987-1648-34-4.
16. Mansilla, D., Pollo-Cattaneo, F., Britos, P., García-Martínez, R. (2012). Modelo de Proceso para Elicitación de Requerimientos en Proyectos de Explotación de Información. Proceedings Latin American Congress on Requirements Engineering and Software Testing. Pág. 38-45. ISBN 978-958-46-0577-1.
17. Mansilla, D., Pollo, F., Britos, P., García-Martínez, R. (2013). A Proposal of a Process Model for Requirements Elicitation in Information Mining Projects. Lecture Notes in Business Information Processing, 139: 165-173. ISBN 978-3-642-36610-9.
18. Deroche, A. & Pollo-Cattaneo, M. F. (2011) Guía de Buenas Prácticas para Completar las Plantillas de Requerimientos para Proyectos de Explotacion de Información. Reporte Técnico GEMIS-TD-2011-01-RT-2012-01. Grupo de Estudio de Metodologías para Ingeniería en Software, UTN-FRBA.
19. Kimball, R., Ross, M., Thornthwaite, W., Mundy, J., & Becker, B. (2011). The data warehouse lifecycle toolkit. Wiley & Sons.
20. J.M. Lázaro Castillo & R. Nuñez López (2007). Caracterización de la Educación Rural en Argentina a través de Minería de Datos con Sistemas Inteligentes. Proyecto Final de Carrera. Instituto Tecnológico de Buenos Aires. Universidad Politécnica de Madrid.



Estudio del estado actual del proceso de ingeniería de requisitos en las empresas antioqueñas de software

Luz Janette Vélez Mejía; correo electrónico: luzjvelez@gmail.com

Alberto Restrepo Velásquez; correo electrónico: arestrep@eafit.edu.co

Resumen

Este estudio tiene como finalidad, encontrar las mejores prácticas, retos y desafíos que enfrentan las empresas que desarrollan productos de software a medida en el departamento de Antioquia-Colombia, considerando específicamente del proceso de desarrollo software la IR, y es una replicación de un estudio realizado en Brasil, Este estudio analiza aspectos como la metodología utilizada, las herramientas de apoyo, el proceso como tal y las mejores prácticas, fue realizado sobre doce (12) empresas a nivel regional. Entre las principales conclusiones del estudio se puede destacar: 1) Se requiere que durante la IR se logre mayor conocimiento del negocio y se ofrezcan soluciones que orienten al cliente y le generen valor, sin esperar que sea necesariamente este quien las encuentre. 2) La IR requiere de una documentación clara, corta y entendible para todo el equipo de desarrollo. 3) Se deben establecer procesos que permitan gestionar los cambios y estos deben ser claros para los desarrolladores y conocidos por el cliente.

Palabras clave: desafíos, documentación y gestión del cambio, elicitación, Ingeniería de Requisitos, mejores prácticas, negocio, retos,

Abstract

This study aims to find the best practices and challenges that the Antioquia-Colombia's enterprises who sell tailor made software have to face, considering specifically the IR development process. This document is a replica of a study that was made in Brazil. This study analyzes some aspects such as the methodology, the support tools, the process and best practices. And it was realized about twelve (12) regional enterprises. The principal conclusions are: 1) It's necessary to get during the IR the best business knowledge and it's also necessary to offer solutions that mean some value. 2) The Requirements Engineering needs a clear documentation and understandable for all development group. 3) It's necessary make enabled process to the change management.

Key words: business, challenges, documentation and Change management, elicitation, requirements engineering.

Introducción

A través de los diferentes experiencias ha quedado demostrada la importancia que tiene para el desarrollo de software con calidad, la Ingeniería de Requisitos (IR) [3], además, estudios como los de Standish Group International y Chaos Report [4] reportan para el año 2011, que el 42% de los proyectos de software deben ser mejorados y el 21% fracasaron debido a causas como: requisitos incompletos (13,1%) y requisitos cambiantes (8,7%). Por su parte, el Software Engineering Institute afirma que el 40% de los proyectos requieren sobreesfuerzo para reparar defectos causados por la mala calidad de los requisitos y el 60% de estos se deben a la ineficiente implementación de los requisitos en los procesos. En este sentido, el estudio realizado por National Institute of Standards and Technology, muestra que el 42% de los proyectos cambian debido a los requisitos, afirmación que complementa las ya mencionada The Standish Group.y Chaos Report, al sostener que en los procesos solo se implementa el 54% de los requisitos originales y el 55% de los requisitos no son usados por los usuarios o clientes[18], por lo tanto aproximadamente el 60-70% de los fracasos ocurridos en los proyectos de desarrollo de software se deben a las insuficiencias de la elicitación, gestión y análisis de requisitos.

Los estudios empíricos son uno de los mecanismos que permite obtener evidencias del estado de la práctica de la ingeniería de software con el propósito de identificar brechas entre la práctica y la teoría. El presente artículo, muestra los resultados de replicación de "Un estudio empírico sobre IR de Empresas Productos de software" realizado en Pernambuco Brasil [2]. Aplicado en Antioquia-Colombia, tiene como motivación presentar un conjunto de prácticas actualmente realizadas por la industria del desarrollo software, en materia de IR y que son llevadas a cabo por las organizaciones que desarrollan software a medida con el fin de apalancar la calidad, brindando soluciones tendientes a satisfacer las necesidades y expectativas de los clientes en los diferentes contextos de uso. [3], [5], [6]. Bajo este argumento este estudio genera la posibilidad de conocer el estado de la IR hoy en las organizaciones de software a

medida, a nivel de las mejores prácticas, experiencias y procesos de apoyo a las actividades. Se tiene pertinencia considerando la concentración que tiene la industria de software en el departamento de Antioquia y que este se ubica en segundo lugar a nivel de economía del país tras la capital que es Bogotá.

Definición del estudio

Objetivo General

Encontrar las principales prácticas retos y desafíos que enfrentan durante la Ingeniería de Requisitos, las organizaciones que desarrollan software a la medida en el departamento de Antioquia Colombia.

Objetivos específicos

- Identificar las estrategias de ingeniería de requisitos utilizadas en la elaboración de los productos software.
- Relacionar las estrategias y destrezas utilizadas en la adquisición de los requisitos con el conocimiento de las necesidades del cliente y su satisfacción.
- Identificar las técnicas de ingeniería de requisitos utilizadas por las organizaciones antioqueñas, en cada una de las fases del proceso.

Contextualización del entorno y alcance la muestra

Dado que no hay información específica del tema en el contexto, esta investigación es de naturaleza exploratoria, siguiendo un enfoque de tipo cualitativo, puesto que dentro de la recolección de los datos se ha tratado la percepción de las personas. Sin embargo para lograr analizar los resultados y garantizar precisión, este estudio se apoya en el método cuantitativo. [2].

Para publicar los resultados de la muestra las empresas se clasificaron con letras a partir de la letra A, y se analizaron teniendo en cuenta aspectos como: las características de las organizaciones, sus empleados, sus clientes, el proceso de desarrollo software y la Ingeniería de Requisitos.

Visto desde una perspectiva local, se eligió Antioquia para el muestreo, porque existen medianas empresas que son altamente exitosas y cuentan con abundante conocimiento acumulado, gracias a sus fuertes trayectorias tecnológicas, así mismo el software antioqueño presenta potencialidades que ofrecen alcanzar el catch-up tecnológico, lo que se ha convertido en estrategia de desarrollo y hace parte de las agendas económicas del país como motor de crecimiento e innovación [7].

Respecto a la muestra, participaron en todo el proceso doce (12) organizaciones de las cuales ocho (8) son desarrolladoras de software y cuatro (4) son empresas catalogadas como clientes significativos en

el mercado y que además poseen áreas de desarrollo propio. El período de recolección de los datos fue iniciado en el año 2011 y los instrumentos utilizados fueron: encuestas con el fin de contextualizar a las organizaciones y entrevista para generar datos específicamente relacionados con las experiencias y prácticas alrededor de la IR.

Hipótesis

Las hipótesis (H) sobre las que se fundamenta este estudio son las siguientes:

- H1: La competencia en el mercado es un objetivo estratégico clave en las empresas que desarrollan productos de software.
- H2: Los mayores desafíos para el crecimiento y gestión del mercado, no son los aspectos relacionadas con problemas técnicos.
- H3: Los requisitos suelen ser inventados por los desarrolladores internos de la compañía.
- H4: Los requisitos son pobremente documentados.
- H5: La priorización de las necesidades y la planificación de la nueva versión son procesos cruciales para la ventaja competitiva
- H6: La relación entre los desarrolladores y los clientes suele ser larga, pero con la proximidad limitada.
- H7: El fracaso del lanzamiento del producto a menudo se produce debido a que el producto no cumple con las necesidades de los clientes.
- H8: Los requisitos son evaluados sólo después de su lanzamiento en el mercado.
- H9: Los desarrolladores de productos de software por lo general tienen un proceso de ad hoc, para definir y gestionar los requisitos.
- H10: El proceso utilizado para IR en el desarrollo de software a medida difiere del utilizado para el desarrollo de paquetes de software, a tal grado que las prácticas de IR tradicionales usadas en el desarrollo a la medida no pueden emplearse para desarrollar paquetes software.

Análisis de los datos recolectados

Se recolectaron datos referentes a caracterizar las organizaciones, los empleados adscritos al proceso de desarrollo software, los clientes, el proceso de desarrollo y la IR.

Información general

Caracterización de las Organizaciones: Las organizaciones que participaron en el estudio fueron identificadas como medianas según los estándares de MinCIT a nivel nacional y como Pymes a nivel mundial, aspecto que genera importancia por la creencia de que las buenas prácticas y soluciones consumen tiempo y están dirigidas a apalancar las grandes organizaciones; en este sentido se desvirtúa esa afirmación cuando se encuentra en este estudio que las organizaciones medianas que además valga el comentario, cumplen un importante papel en el tejido industrial del país, tienen metas tendientes a medir sus procesos para generar mejora continua y poseen los estándares de calidad para la conservación de las buenas prácticas que orientaron sus certificaciones en ISO y CMMI, junto con la adopción de prácticas como ITIL. Se encontró además que las empresas tienen el convencimiento que la calidad de una software comienza con una fuerte y sólida implementación de los requisitos en el proceso de desarrollo software. [7, 8, 9].

Caracterización de los empleados adscritos al proceso de desarrollo software: La base de la estructura de una organización competitiva que desarrolla software, debe ser el recurso humano calificado y de alto nivel de formación, puesto que esto permitiría el desarrollo de las capacidades creativas e innovadoras [11, 12]. Se sabe que uno de los pilares fundamentales para adquirir dicha capacitación es el bilingüismo, a este respecto, este estudio concluye que los niveles de inglés de los que los empleados son bajos en un 53% y calificados como altos en un 13%, factor que afecta el acceso de los empleados a la capacitación en tecnologías de punta y a las organizaciones al momento de asumir los retos propios de la globalización (Figura 1a).

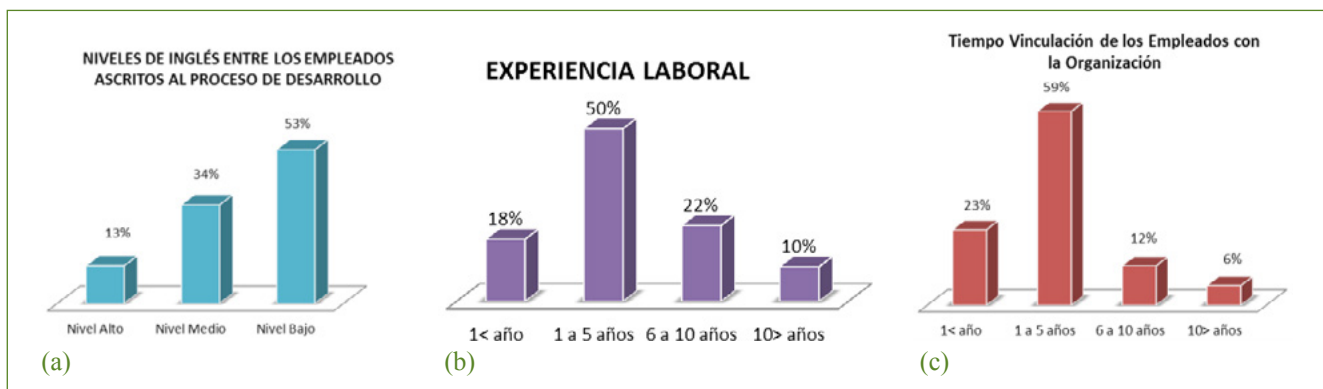


Figura 1. Caracterización de la población encuestada.

Por otro lado, se sabe que un proceso de desarrollo software, esta soportado en un buen equipo de trabajo que debe tener constancia, conocimiento y apropiación de los procesos que llevaran al funcionamiento estable de la organización. De esta manera, este estudio señala que el 32% de los involucrados en el proceso de desarrollo software, tienen experiencia mayor a cinco (5) años, (Figura 1b), esto acompañado de indicadores que muestran que el 59% de los empleados laboran entre uno (1) y cinco (5) años con la

misma organización, (Figura 1c) esta falta de continuidad conlleva a dificultades en la construcción del conocimiento, la apropiación del concepto y el desarrollo de las propuestas estratégicas de la organización a largo plazo [10, 13].

Este estudio ahora contrasta las certificaciones habilidades y conocimientos específicos que poseen los empleados del medio, con las requeridas por el mercado los resultados son:

- Empleados: JAVA, Microsoft .NET, 5, Microsoft, CMMI, Oracle, PMP, Sql Server, UML, RUP, Programmer, Itil.
- Organizaciones: Tecnologías Java, Microsoft .NET, CMMI, Oracle, PMP, PMO para gerencia de proyectos, SAP, Arquitecturas SOA, Especialización en Desarrollo de Software, Maestría en Ingeniería de Software.

Referente a los métodos o estrategias que utilizan en la actualidad las organizaciones para actualizar a sus empleados se encontraron: Programas de certificación de los fabricantes, e-learning y cursos presenciales, capacitaciones internas y apoyo en el horario para que la gente estudie, capacitaciones externas, capacitaciones internas, grupos de estudio internos, grupos de práctica (comité de analistas funcionales, comité de arquitectura), grupos para certificaciones, entrenamientos externos, utilización de planes de formación integral y planes orientados a las carreras específicas.

Caracterización de los clientes. Los nichos de mercado encontrados en las organizaciones estudiadas estaban relacionados con el desarrollo a la medida para compañías de Servicios y finanzas, Outsourcing para desarrollo de productos genéricos orientados a las comunicaciones y la optimización de procesos (CEBP), Retail, Sector financiero y bancario, Sector petrolero, Sector de la construcción, Empresas en el exterior en sectores financieros. En cuanto a la ubicación geográfica de los clientes se observa la necesidad de tener mayor incursión en el comercio extranjero donde los clientes ascienden al 24% (Figura 2a) y donde los desarrollos contratados son 52% nacionales (Figura 2b). Otro resultado conllevó a notar que los clientes son clasificados como: grandes, medianos y pequeños, basados en criterios como la cantidad de puestos de trabajo, de tecnología que usan y el número de empleados, de esta manera el mayor porcentaje de clientes atendidos son clientes grandes con 66% (Figura 2c).



Figura 2. Caracterización de los clientes.

Referente a los niveles de satisfacción de los clientes, el 80% de las organizaciones encuestadas respondieron que estos oscilan entre el 81% y 90%. Y las causas de insatisfacción de los clientes están centradas en los retrasos en la entrega, generados por: Inicio tardío de los proyectos a causa del desconocimiento de las necesidades del cliente, arreglos de última hora a los productos, demoras en la construcción por retrasos en tener los requisitos iniciales listos, discusiones sobre la claridad en requisitos (lo que el cliente esperaba vs lo entregado), dificultades de comunicación entre gestores del proyecto Respecto a la claridad de la especificación de los requisitos del proyecto. Lo anterior permite concluir que la insatisfacción del cliente se da por defectos en IR.

Este estudio investiga los tipos de contratación que utilizan las empresas con el fin de tener una idea general de la relación comercial establecidas con los clientes, ellas fueron: Fábrica de Software, Application services y staff augmentation en formato de insourcing y outsourcing, proyecto valor cerrado y outsourcing, contrato marco, tiempo fijo y costo fijo, contrato por etapas, contrataciones por tiempo y materiales o a costo fijo dependiendo del nivel de conocimiento del proyecto y del riesgo que este implique, por outsourcing in situ, por outsourcing en TF, proyecto llave en mano. No obstante se sabe que la satisfacción del cliente está directamente ligada a la correcta interpretación de las necesidades y a su satisfacción.

Con respecto al uso de herramientas, este estudio destaca a CRM como la herramienta de gestión del cliente utilizada en el 50% de las organizaciones. (Figura 3a). Y las estrategias de comunicación usadas con los clientes son: Tener un modelo comercial de ejecutivos de cuenta y especialistas de producto, nombrar siempre en los proyectos un gerente que esté haciendo planeación, realizar reuniones de seguimiento, contacto cercano y reiterativo en todo momento y tener ejecutivos de cuenta. Como estrategia para captar nuevos clientes están la realización de marketing con 60% [6] muy por encima de varias estrategias nombradas.

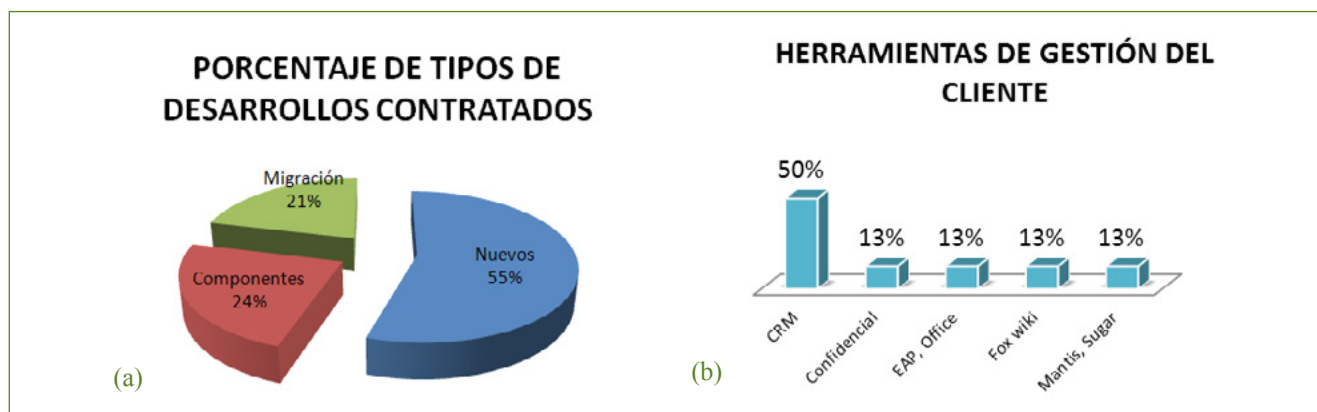


Figura 3. Herramientas y Tipos de Desarrollo

Caracterización del Proceso de Desarrollo Software en la Organización: Se caracterizaron los roles que cumplen los empleados y se encontraron: Administrador de Base de Datos, Administrador de configuración, Administrador de Redes, Comunicaciones y Sistemas Operativos, Administrador de Seguridad, Analista de Procesos, Analista de Sistemas/Analista Técnico Funcional, Analista QA (Quality Assurance), Arquitecto, Aseguradores de calidad, Consultor Funcional Junior, Consultor Funcional Sénior, Desarrollador de Aplicaciones Cliente Servidor, Desarrollador de Aplicaciones Web, Diseñador de Soluciones, Ejecutivo Comercial, Especialista en Seguridad de Aplicaciones, Especialistas en Tecnología, Ingeniero de validación y verificación, Líder / Gerente/ Responsable de Calidad, Líder de Proyecto, Programador, Project Manager / Director de Proyectos, Tester. [16].

Con relación a los lenguajes de desarrollo software utilizados, llama la atención cómo lenguajes como Java alcanzan gran versatilidad según, la tabla 1. Respecto al porcentaje de desarrollos se encontró que el 55% corresponde a la elaboración de productos nuevos. (Figura 3b). En referencias a las metodologías de desarrollo software utilizadas los datos fueron: Scrum 75%, seguida de RUP, ágil, cascada y extrema al mismo nivel, 50%, las más usadas (Figura 4).

METODOLOGÍA DE DESARROLLO UTILIZADA POR LA ORGANIZACIÓN

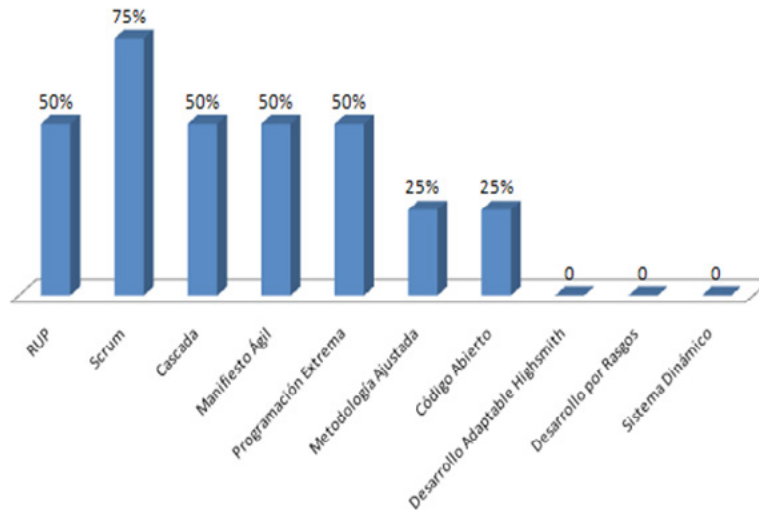


Figura 4. Metodologías de desarrollo utilizadas.

Tabla 1. Lenguajes de desarrollo software utilizados por las organizaciones.

LENGUAJE DE DESARROLLO SOFTWARE UTILIZADOS POR LAS ORGANIZACIONES		
TIPO DE DESARROLLO	LENGUAJES	Uso
Web	J2EE, DotNet	35%
	Java con Oracle, Java con SqlServer, Java con Postgress, Java con MySQL, PuntoNet con SqlServer	80%
	JAVA Microsoft .NET – PB – PHP	80%
Sistemas embebidos	JAVA - Microsoft .NET – PB – PHP	10%
Sistemas de información, ambiente y lenguajes de desarrollo utilizados	PuntoNET, Java, VB, Powebuilder, RPG y Cobol	100%
	ABAP/SAP	10%
Sistemas en tiempo real	JAVA - Microsoft .NET – PB – PHP	10%
Desarrollos móviles	PhoneGap, J2ME, - Microsoft .NET, Mobile	20%
	J2EE, DotNet	20%
SmartClient	DotNet	10%
Integración de aplicaciones	BizTalk, WespHERE Integration Developer	10%
Inteligencia de negocios	SQL SERVER, Oracle BI	10%
Consultoría en arquitectura de software	Enterprise Architect	5%

Caracterización de la IR. Para el 100% de las organizaciones del estudio es importante la IR y dicen tener normalizado el proceso. La tabla 2 cita las experiencias del cómo se realiza y los insumos utilizados, [6, 19, 20]. Referente al uso de metodologías de apoyo a IR se encontró que el 36 % usan metodología propia, el 24% utilizan RUP, y el resto (40%) utilizan otras metodologías. [6]. Las organizaciones además esperan de parte de la IR: Capacitar al grupo de trabajo, afinar las prácticas existentes, que sea la base de estimación, mejorar el diseño, mejorar el análisis de los casos de uso, lograr que el analista funcional entregue productos desarrollables, hacer un mejor modelado de procesos de negocio, simplificar el proceso, mejorar el tiempo dedicado al proceso de requisitos, mitigar los riesgos del alcance, mejorar la interpretación de las necesidades del cliente, cerrar la brecha entre lo elicitado y lo entregado. En complemento la gráfica 5 resume el estado de la documentación, la elicitación y el tiempo dedicado al proceso en la IR.

Tabla 2. El cómo del proceso de IR y los insumos.

EL CÓMO DEL PROCESO DE INGENIERÍA DE REQUISITOS REALIZADO Y LOS INSUMOS	
Experiencia	Insumos
Basados en RUP. Los pasos son: Identificar necesidades, entender el problema, definir CU, priorizar, afinar y transversalmente realizar control cambios.	Work shop, documento de especificación, documento CU, documento de atributos de requisitos, documento de visión, Trazabilidad
Los pasos son: Entrevista, reunión grabada y bien documentada, vaciado a herramienta de requisitos y (filtrado), reconocimiento de requisitos funcionales y no nacionales, etc. recepción de documentos con requisitos, elaboración de prototipos.	Entrevista, encuesta, documentos usuario
Se realiza entrevista, se refina, se completa la información, se analiza, se vacía en herramienta de requisitos y se regresa al cliente para validarla.	Plantillas propias preestablecidas en la herramienta de Ingeniería de Requisitos
Partiendo de las necesidades del negocio se detallan características, requisitos funcionales, no funcionales y se realizan prototipos que se validan.	Insumos: WBS, dentro se tiene para cada proceso tenemos la lista de los entregables, hay instructivos para llevar a cabo cada proceso, listas de chequeo, participación de arquitectos para dar vía libre al requisitos.
Se hace Elicitación, validación, CU	Cuestionarios. EAP. Balsamia (para prototipos), Excel
Reuniones de contextualización, para conocer la técnica de elicitación, entrevista, lectura de documentos previos, depuración de requisitos.	Software Oasis, Documentación del cliente
Dada la especificación del negocio, hay catálogo de requisitos donde el cliente construye un perfil y determina el método y hacen la prueba que desea.	Insumos: Proceso de mercadeo y ventas, Catálogo de requisitos, (preventiva) documenta en oportunidad comercial, y cada servicio tiene un costo basado en una encuesta se sabe cuántos campos (una tabla). En la prueba se toman los requisitos de seguridad y con base en la elección del cliente en el catálogo se llama prueba de más o menos rigurosidad y se presta el servicio al cliente.
Se estructura todo el proyecto basado en el concepto de arquitectura empresarial que luego se divide en 4 niveles de arquitectura: procesos, sistemas, componentes y física. Hay interpretación de requisitos en las 4 arquitecturas. El mayor apoyo de la Ingeniería de Requisitos está en la parte de arquitectura de sistemas (parte de productos software) requisitos no funcionales se involucran en la vista de componentes a través del diseño de componentes. Es un modelo iterativo incremental definido basado en requisitos	Insumos: En la venta y la visión el insumo es el conocimiento de que hay en la empresa, y que en el medio, normas gubernamentales, antes de ir al cliente para que cuente sus necesidades. En la especificación funcional el insumo es la visión construida en los 4 niveles. En diseño se usan más el insumo de componentes y el de infraestructura y en la implantación es totalmente el tema de infraestructura (poner en marcha el sistema).

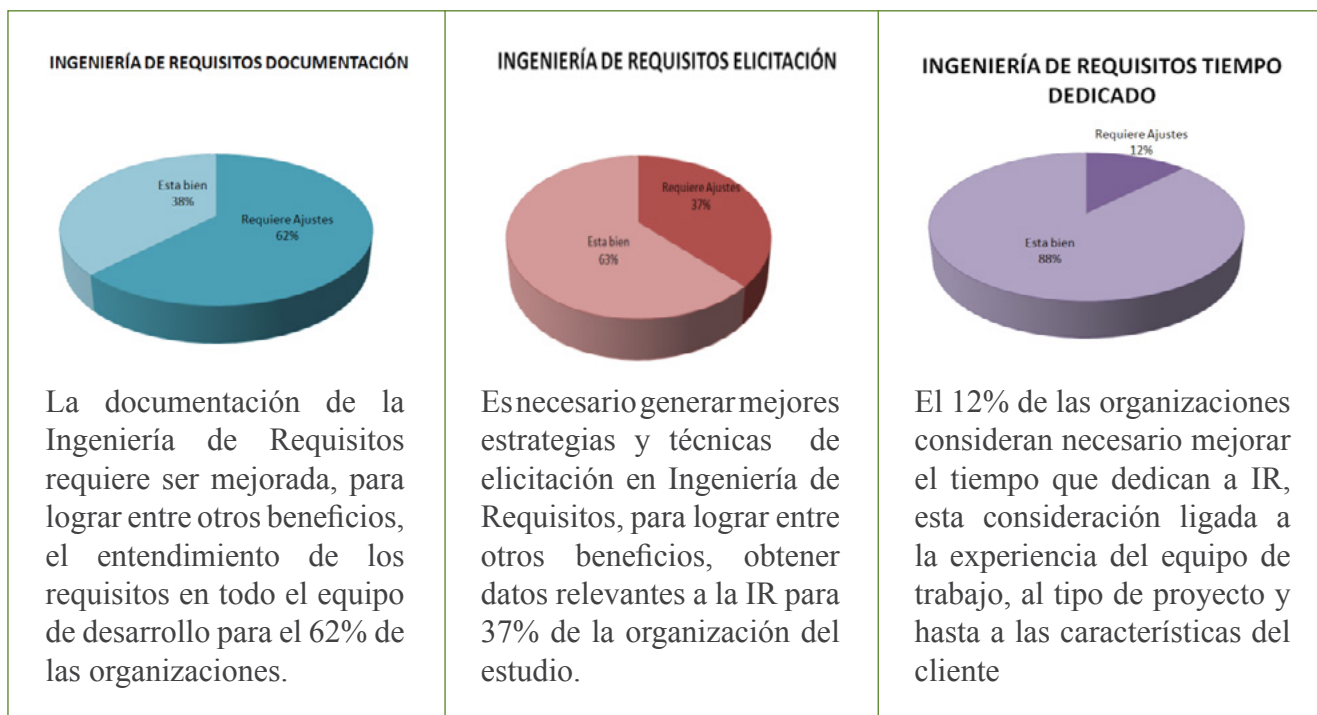


Figura 5. Resumen de resultados mejoras en ingeniería de requisitos.

Referente a las oportunidades de mejora encontradas en el proceso de IR, se encontraron las siguientes: En documentación: los resultados mostraron debilidades en: Productividad, consistencia, claridad, completitud, rastreabilidad, enfoque a los resultados. En la elicitación: de los requisitos tienen que ver con una marcada tendencia a utilizar la entrevista como única técnica de elicitación, por tanto es necesario preparar a los ingenieros de requisitos para variar la técnica, utilizando métodos pedagógicos, que se amolden a los diferentes tipos de organizaciones y clientes generando información más valiosa, utilizando técnicas que lleven a conocer los diferentes puntos de vista de los stakeholders y posterior consenso. A este respecto las organizaciones afirman que la elicitación utiliza términos muy vagos que propician la desinformación porque no alcanzan a tratar el tema a profundidad debido al poco tiempo destinado para ello y a la mezcla de diferentes temas en la misma sección. En el tiempo dedicado al proceso: Este depende en gran medida de factores que tienen que ver con la experiencia del ingeniero, quien debe tener la capacidad de encontrar los stakeholders de mayor aportación, incluirlos en el proceso y encontrar estrategias que generen información valiosa; en este sentido es importante tener en cuenta el entrenamiento que se le suministre al equipo para realizar el proceso. De la misma manera, otras opiniones afirman que el tiempo de la IR depende de su duración del proyecto, a su vez se encontró que como factor de demora está en el hecho de que cuando el cliente entrega documentos de requisitos, se debe dedicar mucho más tiempo a esclarecer lo que realmente quiere decir. La figura 6a donde se muestra a la documentación como la etapa que requiere ajustarse en el 56%.

La IR debe apoyar aspectos como: el conocimiento del alcance, la destinación de los recursos, la estimación del proyecto, la figura 6b muestra que la IR apoya hoy la estimación del proyecto en 30% siendo este el principal factor que debería ser apoyado por la IR y que de cara al cliente representa costos y tiempo, otro aspecto que se considera pobremente apalancado en el 100% de las organizaciones es el apoyo desde la IR al conocimiento del negocio.

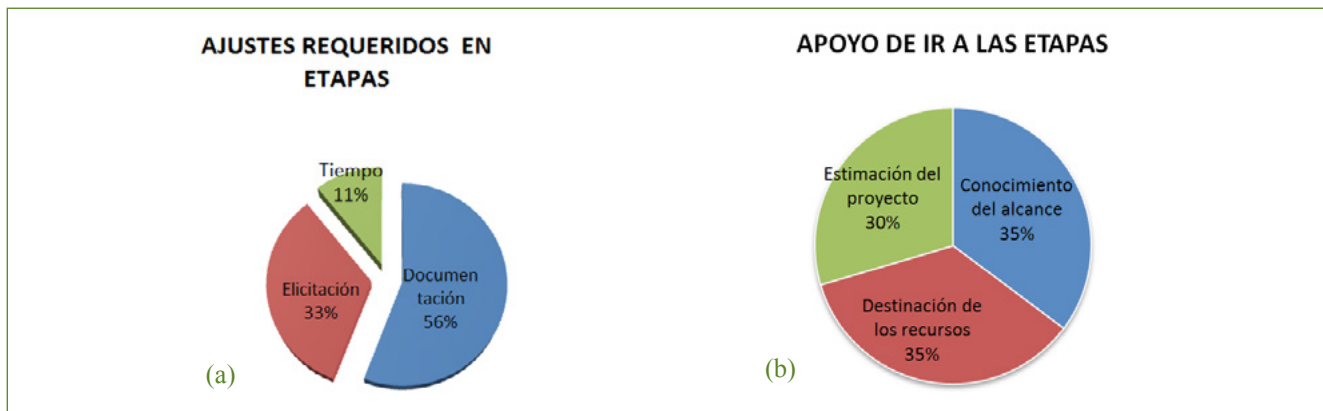


Figura 6. Ajustes y apoyo de la IR en las etapas del desarrollo Software.

Referente a las herramientas para realizar el proceso de IR, este estudio encontró que EAP se utiliza en un 70%, a si mismo se encontró que aunque el 100% de las organizaciones usan repositorios, el uso de fuente de consulta para estructurar requisitos es del 60% y la consulta al repositorio se da entre el 51% y el 60% de los empleados.

Con relación a las estrategias de versionamiento de requisitos ellas fueron: Uso de herramienta Microsoft Visual SourceSafe partiendo de línea base, tener proceso de gestión de cambio y control de versiones. Uso de la herramienta EAP. Por cada fase se versiona (fase de elicitación, fase diseño...). Un equipo se reúne para determinar el impacto del cambio y lo aprueba, luego lo negocia con el cliente. Se hace todo el proceso de gestión de la configuración, más herramientas para manejo de versiones. Se hace gestión de la configuración en Tim System y por cada cambio se genera una nueva versión una vez validado el cambio. Se tiene catálogo de fox wiki. Se hace gestión de configuración de CMMI, hay versionamientos y rastreabilidad. Por otro lado se respecto a los proyectos en los que las organizaciones presentan mayor destreza para la elicitación de requisitos están: Software a la medida, sector financiero, proyectos muy grandes, sector de seguros, desarrollo de servicios, consultoría SAP, automatización de procesos, criterio de seguridad, migración de legassi a web, comunicación con sistemas, (SAP con otro sistema), relacionamientos entre sistemas y la organización.

Trabajos futuros

- Replicar el estudio empírico en otras regiones con el fin de obtener conocimiento de las actividades, problemas y desafíos enfrentan las empresas de otros lugares.
- Se requiere realizar un estudio que compendie las buenas prácticas en Ingeniería de Requisitos encontradas en los varios estudios realizados.
- Es importante conocer las experiencias uso y contenidos de los repositorios de requisitos.

Resultados y buenas prácticas

Resultados de las Hipótesis (H)

H1. La competencia en el mercado es un objetivo estratégico clave en las empresas que desarrollan productos de software: La competencia lleva a las organizaciones a: comprimir los plazos de entrega, generar nuevas funcionalidades al producto, sacrificando en muchos casos la calidad. Además aunque las organizaciones reconocen la importancia del proceso en muchos casos no lo tienen institucionalizado, aunque en la mayoría de los casos si lo tienen descrito.

H2. Los mayores desafíos para el crecimiento y gestión del mercado, no son los aspectos relacionadas con problemas técnicos: Se demostró que los niveles de estudio de los empleados son competentes para sus funciones y que además que se realizan capacitaciones frecuentes en temas técnicos, sin embargo se presentan fallas en aspectos relacionados con el conocimiento del negocio que brinda elementos para la generación de ideas que representen valor. Y en falta de pericia al elicitar y guiar la obtención de los requisitos.

H3. Los requisitos suelen ser inventados por los desarrolladores internos de la compañía: Los problemas de definición de alcance, volatilidad en requisitos y comprensión de los mismos, lleva a que en determinados casos estos sean inventados por los desarrolladores internos de la compañía, por tanto se hace necesario tener una técnica de IR que brinde claridad en cuanto a la gestión del cambio, y a estándares de documentación claros donde se logre de la identificación clara del alcance para cada entrega.

H4. Los requisitos son pobremente documentados: La documentación es considerada en el medio uno de los factores críticos, puesto que los requisitos al documentarse deben ser entendibles y claros para varios públicos dentro del equipo de desarrollo y en consecuencia, es mediante los requisitos cumplidos y documentados que se debe medir el alcance del proyecto y sus recursos, en el medio este tema se considera débil, se recurre a utilizar técnicas ad hoc o juicio experto para la estimación y a dejar el documento de requisitos como un apoyo. Cabe anotar que la calidad en los requisitos no necesariamente es igual la cantidad, más bien se requieren datos concretos, por el contrario el exceso de datos es equivalente

a la mala documentación en requisitos pues se hace difícil de manejar y se torna improductiva para la persona que la crea y un obstáculo para las personas que buscan la información para hacer su trabajo. Mientras que la documentación incompleta de los requisitos, impide la rastreabilidad en los mismos, actividad que es particularmente importante porque una de las características principales del proceso de desarrollo software es el manejo de requisitos cambiantes, que de no hacerse de manera adecuada causa demoras, sobrecostos y desarrollo no acordes con las necesidades del cliente, e impide el desarrollo del proceso de gestión de cambio que como consecuencia afecta directamente la calidad del producto. En general, las empresas participantes en el estudio no realizan una documentación formal de los requisitos esto a consecuencia de la presión de los mercados en cuanto al tiempo y el hecho de que el documento no tiene carácter contractual entre el cliente y la compañía de desarrollo.

H5. La priorización de las necesidades y la planificación de la nueva versión son procesos cruciales para la ventaja competitiva: La correcta selección de los requisitos en la planificación de cada entrega de un producto implica el análisis su importancia para el cliente, las estimaciones de costos y las interdependencias de los requisitos. Además teniendo en cuenta que un proceso de estimación requiere el conocimiento claro de los requisitos que cubren cada versión para que se pueden estimar el tiempo de desarrollo y los recursos humanos y técnicos involucrados, lo que garantiza cumplimiento en tiempo y satisfacción de las necesidades y expectativas reales del usuario y por tanto calidad del software y la competitividad en el mercado.

H6. La relación entre los desarrolladores y los clientes suele ser larga, pero con la proximidad limitada: Se logró establecer que el mercado de desarrollo de software antioqueño se enfrenta a varias situaciones que demuestran proximidad limitada con los clientes, entre estas está: los clientes que ocultan información por no considerarla relevante o bien por considerarla secreto del negocio, esta situación obstaculiza en gran medida el desarrollo del proyecto y evidencia proximidad limitada entre clientes y equipo de desarrollo.. Otra situación se da cuando los clientes no tienen claridad en sus necesidades y presentan en cada acercamiento, un cumulo de ideas encontradas y cambiantes que se entremezclan entre necesidades de la organización y los deseos de los usuarios. Ante las anteriores situaciones, las organizaciones han generado diversas estrategias y documentos de confidencialidad con el fin de generar confianza y claridad entre los usuarios, además han tratado de establecer nichos de mercado donde la experiencia le brinda a la empresa de desarrollo elementos de conocimiento del negocio. En caso de no lograr compromiso y proximidad con el cliente se prefiere no realizar el negocio.

H7. El fracaso del lanzamiento del producto a menudo se produce debido a que el producto no cumple con las necesidades de los clientes: Las fallas en el proceso de IR espacialmente durante la elicitación y la documentación hacen que los productos iniciales presenten defectos en su primera puesta en producción. Las causas apuntan a falta de comunicación clara y frecuente con el cliente, a información oculta.

H8. Los requisitos son evaluados sólo después de su lanzamiento en el mercado: Factores como la falta de proximidad con el cliente, la aceptación de requisitos fuera del alcance hacen que en muchos casos los requisitos sean evaluados después del lanzamiento.

H9. Los desarrolladores de productos de software por lo general tienen un proceso de ad hoc, para definir y gestionar los requisitos: las organizaciones tienen en muchos casos documentado el proceso IR sin embargo este carece de institucionalización y o de completitud frente a situaciones como la gestión del cambio lo que lleva a aplicar soluciones ad hoc. Por otro lado, hoy las organizaciones han creado estrategias tales como: tener el proceso formalizando, utilizar metodologías que lo apoyen, realizar seguimientos que midan los niveles de satisfacción de los usuarios durante el proceso, etc. No obstante, aún existen factores de ambigüedad en los requisitos, falta documentación clara y de habilidad en la elicitación que hace que los procesos sean ad hoc.

H10. El proceso utilizado para IR en el desarrollo de software a medida difiere del utilizado para el desarrollo de paquetes de software, a tal grado que las prácticas de IR tradicionales usadas en el desarrollo a la medida no pueden emplearse para desarrollar paquetes software: La diferencia radica en la necesidad de conocer las necesidades puntuales, es una realidad que entre el 40% y el 60% de los errores y defectos del software son el resultado de una pobre gestión y definición de requisitos, aproximadamente la mitad de los problemas encontrados se podrían haber evitado, simplemente con dejar claro desde el principio, lo que el cliente espera del proyecto, bajo este contexto se deja ver la necesidad manifiesta de tener contacto con el cliente y conocer sus necesidades para lograr calidad en el software a medida, mientras que a esta tarea en el software COTS no se le puede dar cumplimiento detallado dado que no se tienen clientes específico sino consumidores varios y estrategias de marketing para apoyarse [6].

Buenas prácticas

En elicitación:

- El analista debe tener conocimiento del negocio para lograr el 100% del entendimiento del cliente.
- Se deben institucionalizar paquetes con recursos pedagógicos alternativos de tal manera que el elicitador pueda refenciarse.
- Es necesario verificar que el elicitador conozca diferentes técnicas y sepa cómo, cuándo y a quienes aplicarlas.
- La investigación de mercado debe formar parte de la elicitación
- Tener en una actividad grupal de elicitación es decir, a más de una persona tomando nota de los requisitos evita que la información se pierda y garantiza mayor grado de validez.
- El ingeniero de requisitos debe entender la necesidad, de plantear soluciones y orientar, en lugar de pretender que el cliente lo haga, por lo tanto la ingeniería debe tener buenos niveles de abstracción de los requisitos de negocio para lograr soluciones viables y con valor.

- Los requisitos no funcionales deben tener un nivel alto de importancia para la IR dado que deben ser acordes con el tipo de organización y por tanto se requiere hacer lectura de ella para obtenerlos.

Verificación y validación:

- Involucrar al cliente demostrándole la importancia de su participación en el proceso garantiza buenos resultados en el momento de la validación.
- Se recomienda usar como insumo de IR como plantillas.
- La organización debe tener claramente definida la gestión del cambio.
- El cliente debe conocer detalladamente los costos e implicaciones de cada cambio.

Documentación:

- Además del uso de documentos de confidencialidad que brinden al cliente seguridad para depositar su información, es importante hacer la demostración de la trayectoria de la empresa de desarrollo software con otras organizaciones para generar confianza.
- La documentación debe ser clara y concreta tener adecuadas técnicas de lectoescritura, con el fin de no confundir ni generar lecturas parciales en el equipo de desarrollo.

Tiempo:

- La falta de coordinación con el tiempo del cliente lleva al ingeniero a realizar interpretaciones a requisitos vagos, o imprecisos carentes de descomposición, ante esto las estrategias pueden ser el uso de prototipos o modelos que generen mayor detalle y especificidad.
- Para evitar que el tiempo afecte el proyecto una estrategia es fragmentar las entregas así, el modelo de desarrollo utilizado no lo imponga.

Referencias

1. Restrepo, Alberto; Henao, Mónica; Anaya, Raquel. Aplicación de una metodología de Ingeniería de Requisitos a un caso real. VII Workshop Iberoamericano de Ingeniería de Requisitos y Desarrollo de Ambientes de Software, IDEAS2004. Arequipa – Perú, Mayo 2004
2. Cássia Pereira S. Um Estudo Empírico sobre Engenharia de Requisitos em Empresas de Produtos de Software. [posgrado en Ciencias de la computación] Recife: Universidad Federal de Pernambuco, Centro de informática; 2007.

3. Mendoza L, Pérez M, Grimán A. Análisis del Impacto del Proceso de Desarrollo en las Características de Calidad del Software. [Internet] [consultado 2013 Ago 25] Disponible en: http://www.researchgate.net/publication/228556854_Analisis_del_Impacto_del_Proceso_de_Desarrollo_en_las_Caracteristicas_de_Calidad_del_Software.
4. Standish Group International (2011) – www.standishgroup.com. Last Access Engineering”. 3rd Conference for the Promotion of Research in IT at New Universities and University Colleges in Sweden.
5. Sawyer P, Kontoya G. Chapter 2 Software Requeriments [internet]. [consultado 2013 Ago 25]. Disponible en: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.135.6794&rep=rep1&type=pdf>
6. Wiegers KE. When Telepathy Won't Do: Requirements Engineering Key Pratices [consultado 2013 Ago 25] Disponible en: <http://www.processimpact.com/articles/telepathy.html>
7. Castañeda Zamora. Sistema regional de innovación para potenciar la industria del software en Antioquia. [internet]. [consultado 2013 Ago 25]. Disponible en: http://www.dane.gov.co/candane/images/DT_DANE/wp_sistema_innovacion_software_antioquia.pdf
8. IBM. ALM trends: requirements and the role or business anlysts. [internet] [consultado 2013 Ago 25] Disponible en: <http://searchsoftwarequality.techtarget.com/ebook/Requirements-management-tools-and-the-changing-role-of-business-analysts>
9. Pino FJ, García F, Piattini M, Oktaba H. Revisión sistemática de mejora se procesos Software en pequeñas y medianas empresas de software. COMPETISOFT [internet]. 2006; 0.2 [consultado 2013 Ago 25]. Disponible en: http://alarcos.inf-cr.uclm.es/competisoft/publico/downloads/Inf_T%C3%A9cnicos/COMPETISOFT_IT_1.pdf
10. Londoño Londoño LF. Recomendaciones para la Formación de una Empresa de Desarrollo de Software Competitiva en un país como Colombia. [internet]. [consultado 2013 Ago 25]. Disponible en: http://biblioteca.universia.net/html_bura/ficha/params/title/recomendaciones-formacion-empresa-desarrollo-software-competitiva-pais-como-colombia/id/54667899.html
11. Palomino KC. Estudio del comportamiento de la industria del software en Colombia ante escenarios de capacidades de innovación y ventajas comparativas por medio de dinámica de sistemas. [Trabajo de grado como requisito parcial para optar al título de Magíster en ingeniería de sistemas] (Medellín) Universidad Nacional de Colombia, Facultad de Minas, Escuela de Sistemas; 2011.


12. Botero NE. Faltan ingenieros y programadores bilingües. [internet]. [consultado 2013 Ago 25]. Disponible en:
http://www.elcolombiano.com/BancoConocimiento/F/faltan_ingenieros_y_programadores_biling%C3%BCes/faltan_ingenieros_y_programadores_biling%C3%BCes.asp
13. Krüger K. El concepto de sociedad del conocimiento.(Internet) (Consultado 18 Ago 2013) Disponible en: <http://www.ub.edu/geocrit/b3w-683.htm>
14. Pereira Davila R. 10 formas de motivar a sus empleados. [internet]. [consultado 2013 Ago 25]. Disponible en: <http://www.microsoft.com/business/es-es/Content/Paginas/article.aspx?cbcid=266>.
15. Cheng B, Atlee J. Research directions in requirements engineering. , Perú [internet]. [consultado 2013 Ago 25]. Disponible en: <http://www-users.cselabs.umn.edu/classes/Fall-2010/seng5801/readings/future-of-RE.pdf>
16. Mendoza L, Pérez M, Grimán A. Análisis del Impacto del Proceso de Desarrollo en las Características de Calidad del Software. [Internet] [consultado 2013 Ago 25] Disponible en:
http://www.researchgate.net/publication/228556854_Analisis_del_Impacto_del_Proceso_de_Desarrollo_en_las_Caractersticas_de_Calidad_del_Software
17. Pressman, R, Ingeniería del Software: Un enfoque práctico, McGraw Hill 1997. Pressman, R. S. (2006) "Engenharia de Software". Makron Books. 6ª Edición capítulo 7.
18. Gartner IT Key Metrics Data 2011 Summary Reports. [Internet] [consultado 2013 Ago 25] Disponible en: <http://www.slideshare.net/SalmanJameel/gartner-it-enterprise-key-metrics-data-2011>
19. Ian Sommerville y Sawyer Pete, Ingeniería de Requisitos: Una Guía de Buenas Prácticas (Wiley, 1997).
20. Guide to the software engineering body of knowledge. Swebok. IEEE (version 2004)
21. An Introductory Overview of ITIL V3. ITSMF [internet] 2007 [consultado 2013 Ago 25]. Disponible en: http://www.best-management-practice.com/gempdf/itsmf_an_introduutory_overview_of_itsil_v3.pdf.
22. ISO 2001. Guía sobre los requisitos de la documentación ISO 9000:2000. [consultado 2013 Ago 25]. Disponible en: <http://www.iie.org.mx/bolISO02/tecni2.pdf>.
23. Martínez Carod N. Priorización de requerimientos de software utilizando una estrategia cognitiva. [consultado 2013 Ago 25]. Disponible en: http://sedici.unlp.edu.ar/bitstream/handle/10915/20793/Documento_completo.pdf?sequence=1.

24. INTECO – Instituto Nacional de Tecnologías de la Comunicación. Guía avanzada de gestión de requisitos. [consultado 2013 Ago 25]. Disponible en: http://www.pymesonline.com/uploads/tx_icticontent/R02638_gestioncontratos.pdf.

Parte 2

Verificación y Validación





I

ntegración de pruebas automáticas para la optimización de los procesos de producción de software en un estudio de caso real

Johanna Patricia Vinasco Cano¹; María José Roca Valverde¹; César Pardo²

¹ Grupo de Investigación LIDIS, Universidad de San Buenaventura Avenida 10 de Mayo, La Umbría, Vía a Pance. Cali, Colombia johannavinasco@gmail.com, mj_rocav@hotmail.com

² Grupo de Investigación GITI, Universidad Autónoma de Occidente, Cl. 25 # 115- 85, Km 2 Vía Cali – Jamundí, Colombia cjparado@uao.edu.co

Resumen

Las actividades relacionadas con el aseguramiento de la calidad juegan un papel importante en la industria de software debido a su participación activa a través de todo el ciclo productivo. El control de la calidad de los productos o servicios desde etapas tempranas de los proyectos, permite minimizar los costos asociados con esfuerzo y tiempo, lo que implica para el negocio un aumento de la calidad y satisfacción de los clientes. Sin embargo y dado que su implementación no llega a ser clasificada como una actividad netamente productiva, representa un alto costo para las organizaciones que producen software, es por ello que surge la necesidad de implementar procesos de prueba automáticos que optimicen el ciclo productivo y minimicen los costos. Con la inclusión de pruebas funcionales y no funcionales automatizadas en el proceso de software, el enfoque del área se ve positivamente alterado, pues los tiempos del procesos de control de calidad serán significativamente más cortos y los resultados más precisos, por ende, las organizaciones podrán invertir mejor sus recursos y fortalecer actividades como análisis de resultados, mejoramiento de procesos, productos y servicios. En este artículo se presenta un estudio de caso de una

empresa productora de software en la cual se implementó el proceso de pruebas automáticas generando resultados positivos en sus proyectos.

Palabras clave: automatización, calidad, herramientas, mejora, optimización, pruebas, software, testing.

Abstract

The activities related to the Quality Assurance play such an important role in Software Industry due to their active participation throughout the entire productive cycle. The quality control of products and/or services since early project stages allows to minimize costs related to effort and time, what implies for the business a quality rise and customer satisfaction. Nevertheless, since its implementation is not classified as a merely productive activity, it represents a high cost to software producing companies. That is why there is the necessity to implement automated testing processes which optimize the productive cycle and minimize the costs. With the inclusion of functional and non-functional automated test in the software process, the area approach is positively altered, so the control quality process times will be significantly shorter and the results more accurate. For instance, the organizations could invest their resources better, and enhance activities such as result analysis, process, product and service improvement. This article shows a study case of a software production company in which the automated testing process was implemented, getting positive results in its projects.

Key words: automation, improvement, optimization, quality, software, test, testing, tools.

Introducción

La tecnología alrededor del mundo avanza cada vez más rápido, por lo que se hace preciso reinventarse y contar con el manejo de procesos y herramientas que permitan estar a la vanguardia. Hoy en día muchas organizaciones se han esforzado por romper sus paradigmas de administración y han optado por darle una oportunidad a la administración basada en procesos, donde cuentan con conjunto de actividades, roles y artefactos sobre los cuales basan su trabajo diario; sin embargo, esos procesos siguen siendo, en su mayoría, manuales.

En la industria del software hay una amplia variedad de especialidades y enfoques, en este artículo se profundizará en un área transversal al ciclo productivo de software: el área de aseguramiento de calidad de software (SQA) (Piattini Velthuis, 2011), la cual tiene como foco principal evaluar la calidad de los procesos y productos software y detectar el mayor número de defectos antes de llevar un producto a manos del usuario final (G.).

El área de aseguramiento de calidad tiene participación desde la concepción del proyecto hasta el cierre del mismo, esta participación se puede observar en un reconocido modelo de desarrollo de software

expuesto en 1986 por Paul Rook (Rook, 1986); el Modelo en V que presenta los diferentes niveles del desarrollo de software y su correspondiente etapa de pruebas (Figura 1).

En este artículo se profundizará en las pruebas de sistema que involucran tanto pruebas funcionales como no funcionales. En las pruebas funcionales se comprueba que el producto o servicio cumple con los requisitos funcionales (Thomas Müller (chair) / Debra Friedenberg, 2011) y opera según las necesidades del usuario final, en las pruebas no funcionales no sólo se verifica que el producto o servicio opere de forma correcta, sino que lo haga cumpliendo los requerimientos no funcionales especificados, por ejemplo: uso correcto de los recursos hardware, tiempos de respuesta, usabilidad, rendimiento, estrés, entre otros; en este sentido, son entonces las pruebas funcionales la base principal para iniciar las pruebas no funcionales.

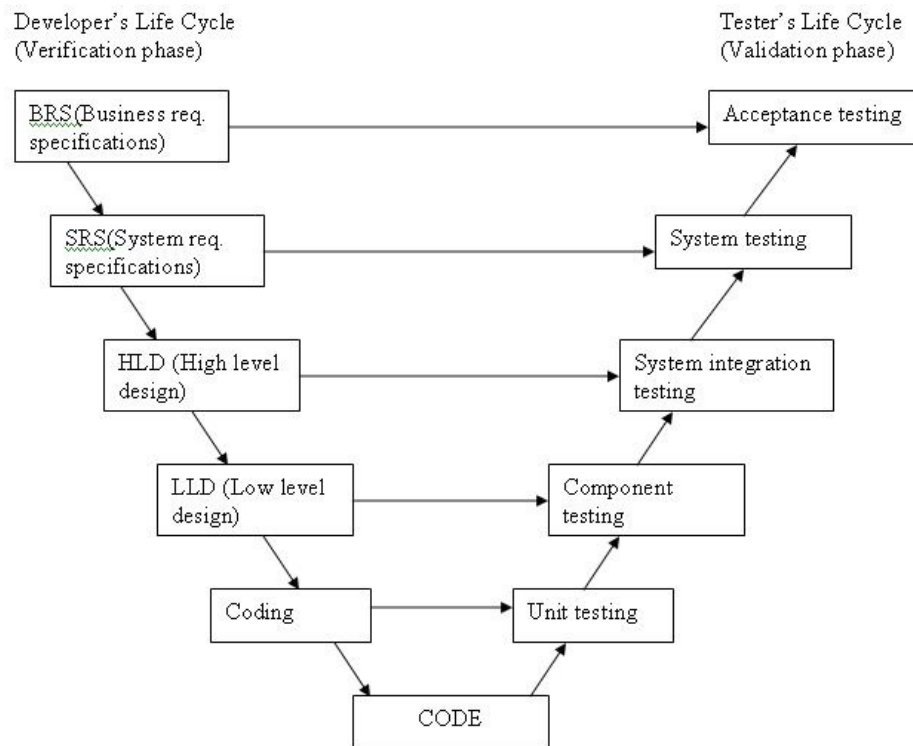


Figura 1. Modelo V (Guide).

La automatización de pruebas tanto funcionales como no funcionales, optimiza los procesos de calidad y reduce significativamente el esfuerzo invertido en la actividad de Testing, gracias a ello, no sólo se cuenta con resultados oportunos y precisos del comportamiento de un producto software con base en requisitos funcionales y no funcionales, si no también es posible construir un framework de pruebas automáticas, que permitan reutilizar en un futuro los componentes generados en los proyectos.

Además de la presente introducción, el artículo está organizado de la siguiente manera: en la sección 2 se presenta un marco teórico general para ubicar al lector en las temáticas relacionadas al desarrollo de este artículo, la importancia de automatizar las pruebas y los fundamentos acerca de: por qué automatizar y qué se debe automatizar. La sección 3 presenta los resultados y lecciones aprendidas a través de la realización de un estudio de caso. La sección 4 presenta un conjunto de lecciones aprendidas generales. Finalmente, la sección 5 presenta las conclusiones y trabajos futuros.

Automatización de pruebas

¿Qué es la automatización de pruebas?

La automatización de pruebas es posible clasificarla en dos grupos, la automatización de pruebas Funcionales y automatización de pruebas No Funcionales. Para la ejecución de las pruebas automáticas funcionales, se deben generar un conjunto de programas o scripts¹, que se realizan generalmente por medio de la herramienta seleccionada para la ejecución de pruebas. Los scripts generados y estabilizados son la base para crear las pruebas no funcionales automáticas, idealmente antes de ejecutar una prueba no funcional, la funcionalidad debe estar operando de forma correcta.

La automatización de pruebas atiende la necesidad de tener resultados más precisos de procesos que no pueden simularse manualmente, como las pruebas con altas cantidades de usuarios, pruebas sobre múltiples plataformas, con alto volumen de datos o pruebas en escenarios con especificaciones particulares según las necesidades del cliente, entre otras características. Si no hay automatización de pruebas es muy complejo detectar los fallos de tipo no funcional dentro de una aplicación, es probable entonces, que estos fallos sean detectadas en producción, lo cual genera altos costos, no sólo por la corrección del error en una etapa muy avanzada de un proyecto, si no porque se tiende a perder credibilidad ante los clientes debido a que los productos no tienen un debido control de calidad en aspectos no funcionales, principalmente.

La automatización de pruebas generalmente no se enfoca en la construcción de herramientas de automatización, sino en sacar el mayor provecho a las herramientas existentes y generar estrategias con base en ellas, también es importante reconocer que automatizar va mucho más allá que realizar grabaciones. Asimismo, es necesario tener en cuenta que para generar soluciones de alta calidad y que se ajusten a nuestras necesidades, se requiere de algunas habilidades de programación que permitan complementar los scripts, dado que las herramientas tienen algunas restricciones y en ocasiones las grabaciones no se realizan correctamente.

¹*Script: Es un conjunto de instrucciones almacenadas en un archivo de texto, que es ejecutado por una aplicación externa y permite automatizar tareas.*

¿Por qué automatizar?

Al hablar de pruebas automáticas nos enfocamos en cómo mejorar nuestra forma de trabajar, no sólo se trata de realizar pruebas no funcionales, sino también funcionales, de tal forma que nuestros procesos dependan cada vez menos de las personas, lo que permite enfocar el recurso especializado más en proyectos estratégicos y menos en procesos operativos.

Con respecto a las pruebas automáticas funcionales, el objetivo es poder reutilizar los scripts generados con el fin de involucrar una menor cantidad de recursos (ingenieros y especialistas de calidad) en procesos de pruebas que sean repetitivos, como por ejemplo: pruebas de regresión² (Sosa, 2008), reproceso, pruebas de una misma funcionalidad en diferentes sistemas operativos, diferentes navegadores, entre otros.

En cuanto a las pruebas no funcionales, algunos de los tipos que se manejan en la fase de aseguramiento de calidad, como las de rendimiento, carga, estrés y volumen, son difíciles y algunas veces imposible de simular manualmente, y en el caso que se hagan de esta manera, los resultados obtenidos no son lo suficientemente confiables.

La automatización de pruebas permite replicar errores sin cambiar las condiciones y almacena un registro exacto de los tiempos de respuesta, transacciones por segundo, código defectuoso, componentes de aplicación no encontrados, entre otros (Kumar, 2008). Adicionalmente, minimiza la cantidad de recursos necesarios para la ejecución de pruebas, los tiempos de reproceso y de ejecución de tal forma que los cronogramas de los proyectos no se ven afectados y se obtiene un producto con mayor calidad.

¿Qué se debe automatizar?

Las principales etapas que comprenden el proceso de pruebas de software son: i) Planeación y Control, ii) Análisis y Diseño, iii) Implementación y ejecución, iv) Evaluación de los criterios de finalización y realizar reportes y v) Realizar actividades de cierre (Thomas Müller (chair) / Debra Friedenber, 2011). La definición de qué procesos se automatizarán en la fase de ejecución de pruebas de un proyecto se debe realizar de forma temprana, especialmente en la etapa de “Análisis y Diseño” y basados en la arquitectura de la aplicación. Para definir esto, se deben tener en cuenta, entre otros, los siguientes criterios (Elfriede Dustin, 1999): procesos o funcionalidades que sean repetitivos, procesos que requieran pruebas con distintos datos, procesos con alto consumo de recursos, funcionalidades complejas, aplicaciones que deben correr en múltiples sistemas operativos, motores de bases de datos o navegadores y procesos críticos dentro de una aplicación.

²*Pruebas de regresión: Pruebas de un programa previamente probado que ha sufrido modificaciones, para asegurarse que no se han introducido o descubierto defectos en áreas del software que no han sido modificadas como resultado de los cambios realizados. Se realiza cuando el software o su entorno han sido modificados.*

Estudio de caso

El estudio de caso se llevó a cabo en el área de Aseguramiento de Calidad de Software de una empresa proveedora de servicios software en la ciudad de Cali. En el área se generó la iniciativa de investigación sobre la automatización de pruebas y las diferentes herramientas que están disponibles para ello, tanto comerciales como de libre distribución, esto, con el objetivo de mejorar la calidad y oportunidad de los procesos mediante la automatización de pruebas funcionales y no funcionales y su aplicación al desarrollo de aplicaciones web y de escritorio.

La implementación de este cambio implicó realizar ajustes en la metodología de aseguramiento de calidad, para que desde el proceso de diseño se realizara un análisis detallado de qué se puede o se debe automatizar. Para definir esto y con base en la experiencia, se tuvieron en cuenta los siguientes criterios: procesos repetitivos, procesos de pruebas imposibles de ejecutar manualmente, procesos que requerían pruebas con múltiples conjuntos de datos ó en múltiples plataformas, entre otros.

Proyecto de Automatización: Fase I

Esta fase del proyecto fue principalmente investigativa, el objetivo era encontrar una herramienta de automatización de pruebas no funcionales para aplicaciones en entornos web. Durante la realización de esta fase, se identificó la necesidad de obtener resultados con mayor exactitud de procesos de pruebas no funcionales como las de rendimiento, carga, estrés y volumen, que son difíciles de implementar y algunas veces imposibles de simular manualmente. Por experiencia, se ha comprobado que aún ejecutándolas manualmente, los resultados obtenidos no son suficientemente confiables, adicional a esto, si no hay automatización de pruebas, se torna complejo detectar si hay fallas de tipo no funcional dentro de una aplicación, es muy posible entonces que: si estas fallas existen, sean detectadas en producción, lo cual genera altos costos, no sólo por la corrección del error en una etapa muy avanzada de los proyectos, sino también por la mala imagen generada ante los clientes y posterior a esto, su insatisfacción.

Luego de investigar diversas herramientas, se realizó una categorización donde se detallaron sus principales características y variables como el tipo de licenciamiento, tipo de aplicación, tipos de pruebas soportadas y restricciones, asimismo se realizó un mapeo respecto a las necesidades del negocio basadas puntualmente en la optimización de su proceso de pruebas. Posteriormente, se dio inicio a la etapa de pruebas con aplicaciones web, donde iniciaron proyectos pilotos con cada una de las herramientas, comprobando cuál se adaptaba mejor a los requisitos planteados. Durante las pruebas se presentaron inconvenientes de compatibilidad entre los componentes de las aplicaciones desarrolladas y las herramientas para pruebas, por lo que fue necesario recurrir a herramientas Sniffer³ para monitorear el tráfico de la red y así poder ajustar manualmente los script.

³*Sniffer: Es una aplicación que permite monitorear el tráfico de la red.*

Proyecto de Automatización: Fase II

La segunda fase del proyecto fue principalmente de aplicación. Durante esta fase, se llevaron a cabo diferentes proyectos piloto, asimismo, se puso en marcha la recolección de lecciones aprendidas y mejoras previas a la campaña de capacitación y sensibilización. Por otra parte, se lanzó una nueva versión de la herramienta usada para la ejecución de pruebas, con la cual muchas de las restricciones técnicas que existían en el primer alcance del proyecto fueron superadas, lo que se traduce como un aporte importante para fortalecer los procesos de pruebas automáticas para aplicaciones web.

Un punto importante a tener en cuenta para las pruebas no funcionales, es el monitoreo de los recursos físicos durante la ejecución de las pruebas, por ejemplo: el uso del procesador, memoria, disco duro y la red permiten evaluar las capacidades de infraestructura que se tienen de acuerdo a las aplicaciones y definir si se cumple o no con el requisito no funcional especificado.

La primera fase tuvo un enfoque bastante fuerte en aplicaciones web, sin embargo, estaba latente la necesidad de automatizar procesos con aplicaciones de escritorio y con la nueva versión de la herramienta se pudo lograr. En ese sentido, lo que antes consumía demasiado tiempo y esfuerzo elaborando manualmente un código bastante robusto para automatizar una prueba en una aplicación de escritorio, ahora se puede realizar automáticamente grabando las acciones del mouse y el teclado y generando el código con un sólo clic. Es importante aclarar que las pruebas funcionales de aplicaciones de escritorio se pueden integrar con las pruebas funcionales y no funcionales de las aplicaciones web en una misma solución, y de esta manera, realizar una prueba completa y consolidada de los productos y servicios ofrecidos por la empresa.

Cuando los servicios se interrumpen por algún motivo, muchos integrantes del equipo de aseguramiento de calidad de software se deben desplazar hasta las instalaciones de la empresa para realizar pruebas manualmente de todos los servicios, por lo tanto, uno de los principales pilotos fue la automatización del “plan de pruebas de servicios”, el cual consiste en realizar pruebas válidas automáticas de todos los servicios prestados por la empresa. Al finalizar el piloto, se generó un plan de pruebas automatizado y listo para ejecutarse en cualquier momento, el cual sólo requiere una o dos personas monitoreando su ejecución.

Con base en los resultados del proyecto de automatización del “plan de pruebas de servicios”, en la figura 2 se presenta un cuadro comparativo de los tiempos planeados (con base en históricos) para ejecución manual contra los ejecutados de forma automática, donde se puede observar que un proceso que tomaba 6 horas de ejecución manual, pasó a tomar sólo 1,4 horas de ejecución automática, ese mismo proceso requería la intervención de 8 ingenieros del área de calidad realizando la ejecución de pruebas, mientras que con las pruebas automáticas sólo fueron necesarios dos ingenieros, respecto al cubrimiento de automatización de pruebas se evidencia que no está al 100% pues siempre hay procesos puntuales que no son susceptibles de automatización, en este caso sólo cinco pruebas entre 40 tuvieron que realizarse de forma manual. Adicionalmente, el esfuerzo en horario no laboral se redujo, por lo

tanto si antes se debían compensar aproximadamente 64 horas extras, con los procesos automatizados ahora son sólo 16 horas extras. La tabla 1 presenta el porcentaje de ahorro alcanzado al implementar los procesos de pruebas de forma automática.

Al finalizar el proyecto se llevó a cabo una campaña de sensibilización y capacitación sobre pruebas automáticas y divulgación del proyecto de automatización, además, se trabajó en el fortalecimiento del conocimiento de la nueva versión de la metodología de aseguramiento de calidad de software y el uso de herramientas de software para la ejecución de pruebas funcionales y no funcionales para aplicaciones web y de escritorio.

Comparativo pruebas manuales vs. automáticas

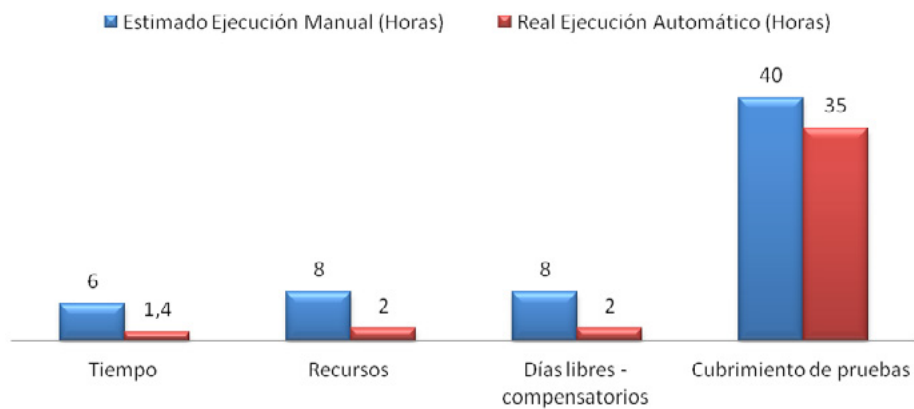


Figura 2. Comparativo pruebas manuales vs. automáticas.

Tabla 1. Estadísticas de mejoramiento.

Diferencia: Estimado manual vs Real Automático	
Tiempo	76%
Recursos	75%
Compensatorios	75%
Cubrimiento de Automatización	87,5%

Luego de llevar a cabo el desarrollo de este proyecto en la empresa, actualmente, el equipo de aseguramiento de calidad de software ha fortalecido sus fundamentos sobre automatización de pruebas funcionales y no funcionales. Gracias a la inserción de pruebas automáticas, se han obtenido resultados más precisos del comportamiento de las aplicaciones bajo diferentes ambientes. Asimismo, se cuenta con un repositorio de scripts que permitirán minimizar tiempos de ejecución en futuros proyectos

Lecciones aprendidas

Existen programas de capacitación sobre pruebas automáticas y herramientas de automatización, que tienden a ser altamente costosos y de difícil acceso, por lo tanto, si se requiere involucrar la automatización de pruebas en los procesos de control de calidad de software de forma autónoma y empírica, es importante tener en cuenta los siguientes aspectos los cuales son presentados a manera de lecciones aprendidas:

1. Idealmente se deben tener claros los objetivos del negocio y el alcance del proyecto, antes de iniciar una investigación herramientas para implementar y ejecutar las pruebas automáticas.
2. Es fundamental contar con el aval y el apoyo de la alta gerencia al iniciar un proyecto de mejoramiento de esta envergadura.
3. Se debe definir un equipo de investigación o automatización y disponer de proyectos pequeños que sirvan como piloto para probar las herramientas, metodologías, artefactos, entre otros y generar conclusiones.
4. Se debe tener un acompañamiento constante del área de arquitectura y desarrollo y contar con el apoyo adecuado para realizar la correcta configuración de los ambientes requeridos para las pruebas.
5. Es importante definir contadores de rendimiento para medir los recursos sobre los cuales se realizan las pruebas, por ejemplo: servidores de base de datos, servidores de aplicaciones, ftp, etc.
6. Es importante involucrar la automatización de pruebas funcionales y no funcionales en el mayor número posible de proyectos en la organización, con el objetivo de crear una cultura de calidad y optimizar recursos.
7. Se debe realizar un acompañamiento entre el área de ingeniería de requisitos y aseguramiento de calidad, con el objetivo de mejorar el levantamiento de requisitos no funcionales.
8. Para el análisis de los datos es necesario contar con bases estadísticas y conocimiento sobre contadores de rendimiento.
9. Se debe ajustar el proceso de aseguramiento de calidad de software e incluir detalladamente la fase de pruebas automáticas.
10. Se debe definir un repositorio donde de forma organizada y estandarizada se empiece a construir el Framework de pruebas automáticas. Además se recomienda tener un repositorio dedicado para las lecciones aprendidas.

11. Si bien es claro que sería muy costoso automatizar absolutamente todo, es importante definir criterios genéricos que sirvan como apoyo a la hora de seleccionar los casos de prueba susceptibles a automatización.
12. Se deben definir métricas que permitan evaluar la mejora de las pruebas automáticas respecto a las pruebas manuales.

Conclusiones y trabajos futuros

A partir de los resultados obtenidos y la experiencia ganada a través del estudio de caso, es posible establecer que las pruebas automáticas son un factor clave en la optimización de procesos de aseguramiento de calidad de software, pues aportan mejoras en la calidad, costo y tiempos de los proyectos. Actualmente existen diversas herramientas tanto de software libre como de pago que las empresas pueden usar para mejorar sus procesos. Con las pruebas automáticas, la fase de aseguramiento de calidad se convierte en algo menos costoso y más estratégico, y es inherente a la mejora de la organización respecto a calidad, tiempos, satisfacción de los clientes, competitividad y productividad, entre otros.

Como trabajo futuro se espera llevar a cabo una investigación detallada sobre las técnicas avanzadas de pruebas automáticas y su aplicación en los procesos de certificación de software, con estas técnicas se incrementaría la efectividad de la automatización dado que es mucho más fácil identificar qué se debe automatizar y de qué forma.

Referencias

1. Elfriede Dustin, J.R, John Paul. (1999). Automated software testing.
2. G., M. A. B. Calidad de Software Extracto del libro en formato digital “calidad tradicional y de software”: Universidad Técnica Federico Santa María. Industrias, Campus Santiago de Chile.
3. Guide, I. What is V-model- advantages, disadvantages and when to use it? Retrieved Noviembre de 2013, 2013, from:
<http://istqbexamcertification.com/what-is-v-model-advantages-disadvantages-and-when-to-use-it/>
4. Kumar, N. S. (2008). Software Testing with Visual Studio Team System 2008.
5. Piattini Velthuis, M. G. G., F.O. / García Rodríguez de Guzmán, Ignacio/ Pino, Francisco. (2011). Calidad de sistemas de información.
6. Rook, P. (1986). Controlling software projects. [Journal]. Software Engineering Journal, 1(1), 16.

7. Sosa, G. M. (Ed.) (2008) Glosario estándar de términos utilizados en pruebas software. . SSTQB: Spanish Software Testing Qualification Board.
8. Thomas Müller (chair) / Debra Friedenber, a. t. I. W. F. L. (2011). Certified Tester Foundation Level Syllabus (Vol. Versión 2011, pp. 78): International Software Testing Qualifications Board.

Parte 3

Aplicaciones de la Ingeniería de Software





Desarrollo ágil usando XRX: un caso práctico

Velásquez William David¹; Jaime Alberto Echeverri²; Miguel Aristizábal³, Mauricio González⁴

¹ Ingeniero de Sistemas de la Universidad de Medellín y Director de investigación y Desarrollo de Visión Tecnológica S.A.S. de la cuál es cofundador en el año 2000. Es co-creador de la herramienta de Inteligencia de Negocios BIABLE y desde hace más de siete años se ha dedicado a la investigación alrededor de las tecnologías XML. Desde hace dos años comparte su conocimiento desde la cátedra en la Universidad de Medellín. wvelasquez@udem.edu.co

² Magíster de la Universidad Nacional de Colombia, actualmente realiza estudios de doctorado en la Universidad de Antioquia, desde hace más de siete años es docente investigador del programa ingeniería de sistemas de la Universidad de Medellín y ha sido líder del grupo de investigación Arkadius. jaecheverri@udem.edu.co

³ Magíster en Ingeniería de la Universidad de Antioquia, actualmente adelanta estudios de doctorado en la Universidad de Antioquia, tiene amplia experiencia en el sector de telecomunicaciones, trabaja actualmente en UNE EPM Telecomunicaciones y pertenece al grupo de investigación ITOS de la Universidad de Antioquia. miguel.aristizabal@une.com.co

⁴ Ingeniero electrónico y actualmente cursa quinto semestre de Maestría en Ingeniería con énfasis en Informática en la Universidad de Antioquia. Sus áreas de acción suman dos años en la docencia y la investigación (Universidad de Medellín y Universidad de Antioquia) y nueve años en el desarrollo de proyectos de instrumentación, control y automatización industrial. mgonzapa@gmail.com,

Resumen

Recientemente la búsqueda de la agilidad en los procesos de desarrollo se ha convertido en un objetivo común a quienes crean aplicaciones, que se ha abordado casi siempre como un problema de cambio en las metodologías. Sin embargo, poco se han cuestionado las herramientas y plataformas utilizadas. La mayoría de las aplicaciones actuales tienen almacenada su información en sistemas de bases de datos relacionales que utilizan SQL como el lenguaje para recuperar y manipular información; Lenguajes Orientados a Objetos para representar la lógica del negocio; y HTML para la Interfaz de Usuario. Estos exigen un extenso trabajo de codificación para transformar los datos entre las tres capas. Nuevos estándares para la manipulación y la representación de datos han demostrado estar mejor adaptados para crear aplicaciones Web escribiendo menos código. Particularmente, lenguajes declarativos usados para consultas (XQuery) y para la interfaz de usuario (XForms) que usan XML como modelo común de representación de la información, agilizan el proceso de desarrollo al eliminar las transformaciones entre capas. El presente trabajo presenta los aspectos relevantes del desarrollo de una plataforma de colaboración usando la arquitectura XRX (XForms-REST-XQuery) y que puede extenderse a muchos otros tipos de aplicaciones, obteniendo beneficios de agilidad sobre las arquitecturas populares en uso actualmente.

Palabras clave: agilidad, desarrollo basado en XML, NoSQL, XRX, XML.

Abstract

Nowadays, the search for agility in software development is a common goal for those who create applications; that is treated almost as a problem of change of methodologies. However, little has been questioned about the tools and platforms used. Most of applications today store information in relational databases systems that use SQL as query language for retrieval and management of the information; object oriented languages for business logic representation; and HTML for the user interface. This needs a big work of coding for transforming the data between the three layers. New standards for data management and representation have proven to be better suited for creating web applications writing less code. Particularly, declarative languages used for queries (XQuery) and user interfaces (XForms) that use XML like common model for information representation, streamline the development process, eliminating the need for transformations between layers. This work presents the most relevant aspects of the building of a collaboration platform using XRX (XForms-REST-XQuery) that can be applied to many other kinds of software, obtaining an advantage in agility over other architectures widely used these days.

Key words: agility, development based on XML, NoSQL, XRX, XML.

Introducción

Si bien el título de este escrito podría sugerir que este es un ensayo más sobre las ya bien conocidas y ampliamente adoptadas metodologías de desarrollo ágil, la realidad es que poco trata sobre el método y sí mucho sobre las herramientas, y sobre todo las arquitecturas que usamos hoy en día para crear aplicaciones. Pero una cosa tiene en común con todo el objeto de investigación de las metodologías ágiles: crear mejor software, en menor tiempo y con menos defectos. Sólo que aquí se aborda desde un punto de vista que pocas veces se ha cuestionado y con la intención de demostrar que puede ser tan benéfico como lo que se ha conseguido al crear nuevas metodologías de desarrollo.

Al igual que la mayoría de propuestas en materia de desarrollo de software, una demostración con todo el rigor del positivismo científico tal vez no sea posible, pero esperamos si aportar ideas valiosas a quienes tienen la tarea diaria de crear software, o por lo menos sembrar la conciencia de que son posibles otros caminos diferentes a los que normalmente se aceptan a la hora de afrontar la tarea de construir aplicaciones y sistemas informáticos.

Desde la óptica de las metodologías, la pregunta que siempre se ha formulado ha sido: ¿Podemos hacer mejor las cosas? Sin embargo, últimamente poco se formula la pregunta desde el punto de vista de las herramientas: ¿Son estas las herramientas correctas para hacer las cosas? Muy seguramente por el amplio despliegue de características de nuestras actuales herramientas, la amplia difusión de su uso y los intereses comerciales detrás de ellas, no hay mucho interés en cuestionarlas, y en opinión de los autores, las actuales herramientas son en verdad altamente capaces, pero solo en las áreas a las que están dirigidas, ya sea almacenamiento (bases de datos relacionales), representación del dominio del problema (lenguajes orientados a objetos) o la interfaz de usuario (HTML).

Pero los problemas asociados a estas herramientas – o mejor, a los paradigmas sobre los que se apoyan estos tipos de herramientas – aparecen al momento de usar los tres juntos en una misma arquitectura, pues con esa combinación aparece la necesidad de realizar un trabajo adicional de diseño y codificación para transformar los datos cuando se comunican entre ellas.

Algunos esfuerzos se han realizado para cuantificar el impacto de estas transformaciones sobre el proceso de desarrollo (Law, 2010) y solamente tal medición ocuparía todo un escrito como este, sin embargo, el sentido común y la experiencia práctica en la creación de aplicaciones nos permite aventurar una afirmación tan simple como que el proceso de desarrollo sería más ágil si tales transformaciones no tuvieran que hacerse.

Y precisamente desde 2006 algunos académicos, pero también muchos empíricos (Wikipedia, 2010), han comenzado a buscar y probar nuevas herramientas donde el trabajo de transformación entre capas arquitectónicas de la aplicación no exista, es más, ni siquiera se realice automáticamente como en las herramientas ORM (Object Relational Mapping) o los generadores de aplicaciones tipo MDD (Model Driven Development).

Precisamente el conjunto de herramientas más maduras que cumplen esta característica son las basadas en XML: las Bases de Datos XML, el protocolo HTTP (particularmente el estilo arquitectónico REST que se basa en HTTP) y los navegadores con soporte XForms. En todos tres el paradigma común es el documento XML, por lo que no es necesario escribir código de transformación entre las capas y adicionalmente el modelo de datos XML (conocido como XDM, por XML Data Model) tiene capacidades de representación del conocimiento superiores a las del Modelo Relacional, pues permite encapsular en un solo documento estructuras complejas que de otra manera sería necesario separar en múltiples tablas relacionales.

La combinación de estas herramientas en una sola arquitectura se ha presentado como XRX (XForms-REST-XQuery) (XRX, 2011) y cuenta ya con una amplia comunidad de desarrolladores y casos de éxito.

Somos conscientes de que los cambios de paradigmas toman tiempo en implementarse, y que van a ser muchos los argumentos a favor del estado actual de las cosas, pero también creemos firmemente en que luego de este aparecerán muchos otros trabajos que los derrumbarán y que en cuanto comiencen a ser visibles los ahorros en esfuerzo y los beneficios en calidad y costos, el cambio sucederá.

Arquitecturas de software

Las Arquitecturas de software constituyen las guías generales, con base a las cuales se pueden resolver problemas específicos. Estas cumplen el papel de los planos en la construcción de un edificio, detallan la estructura, funcionamiento e interacción entre las partes del software. La Arquitectura de software se enfoca en aspectos que van más allá de los algoritmos y de las estructuras de datos (Somerville, 2010).

La mayoría de arquitecturas de software incluyen en su estructura el componente de datos, casi siempre soportado en sistemas de gestión de bases de datos relacionales. Tecnología predominante para almacenar datos estructurados en aplicaciones web y empresariales.

Los sistemas relacionales han demostrado su utilidad a través del tiempo, pero esta puede verse circunscrita a aplicaciones con determinadas características. Una vez se diseña y se desarrolla un sistema bajo el esquema relacional, aspectos como la funcionalidad y el rendimiento de la aplicación no ofrecen la libertad necesaria para actualizar y gestionar fácilmente cambios en los componentes del sistema. Si se definen nuevas relaciones o nuevos tipos de datos con el fin de extender los modelos, se evidencian mayores dificultades en el manejo de las tablas y de las relaciones entre estas. Igualmente ocurre, cuando hay necesidad de desplegar la información en entornos distribuidos con operaciones complejas. Los intentos por actualizar el sistema relacional conducen a una explosión de tablas y la necesidad de establecer muchas combinaciones, esto ocasiona bajo rendimiento, falta de escalabilidad y en muchas situaciones pérdida de integridad. (NoSQL, 2010)

La tecnología relacional está mostrando signos de envejecimiento. A pesar de que funciona muy bien cuando se trata de modelos simples, datos tabulares, con operaciones simples, almacenamiento y acceso a los datos de manera centralizada. Así pues, se registra en las arquitecturas de desarrollo clásicas un punto débil, el cual consiste en la transferencia de datos entre los diferentes niveles (o etapas), ya que diferentes niveles requieren diferentes formatos, diferentes lenguajes de programación y diferentes tipos de sistemas (Wade, 2010).

Actualmente los sistemas de información desarrollados mediante estándares para el intercambio de información estructurada entre diferentes plataformas son ampliamente reconocidos y utilizados. Diferentes Arquitecturas que utilizan como base archivos estructurados, se dan a conocer debido a una serie de características que permiten un desarrollo rápido y fácil (Technologies, 2008). El acelerado crecimiento de internet y la evolución de los lenguajes de etiquetado han exhibido diferentes posibilidades y oportunidades para el intercambio de información y han puesto en evidencia las capacidades para interoperar de una manera uniforme (Karunamu, 2012).

Por otra parte, las prácticas económicas actuales han dejado como legado procesos de negocios conformados por múltiples sistemas, que en la mayoría de los casos son incompatibles. La necesidad del intercambio de información y la integración a diferentes formatos son insuficiencias cotidianas en los procesos de negocio que afectan decisiones al interior de las empresas. El lenguaje XML provee una manera para regularizar el almacenamiento de datos estructurados, información histórica y otra información que requiere administración de contenido, aumentando además la capacidad para procesar documentos (Pemberton, 2006), (McCreary, 2008). De esta manera el navegador se convierte en una poderosa herramienta para el desarrollo de aplicaciones, que puede brindar amplio soporte, nuevas áreas de aplicación y una alta compatibilidad. Se evidencia pues, la necesidad del intercambio de información y la integración a diferentes formatos, que se hacen cotidianos en los procesos de negocio que afectan decisiones al interior de las empresas.

El lenguaje XML se convierte en una poderosa herramienta para regularizar el almacenamiento de datos estructurados, información histórica y otra información que requiere administración de contenido (XSLTForm, 2011).

La comunidad de desarrolladores XML ha propuesto la arquitectura XRX para dar solución a la problemática antes planteada. Su principal virtud consiste en manejar la misma unidad de representación de información – documentos XML – de extremo a extremo de la aplicación [11]. Desde que la información es manipulada en la interfaz de usuario basada en XForms, pasando por su procesamiento mediante XQuery, para ser finalmente almacenada en una base de datos nativas XML, la información permanece representada como un documento XML, en lugar de ser transformada entre diferentes representaciones como en las aplicaciones que usan HTML-OOP-SQL (XSLTForm, 2011).

En el presente trabajo se muestra un desarrollo basado en la en la arquitectura MVC (Modelo Vista Controlador) desarrollado con XRX, que utiliza eXist (eXist, 2011), la base de datos nativa XML

(XNDB), la cual utiliza como unidad fundamental de almacenamiento estructuras específicas para documentos XML, eXist tiene la facultad de almacenar documentos XML y ejecutar consultas en Xquery, se posiciona además como un servidor de aplicaciones Web. Permite a los usuarios utilizar XQuery para la creación de páginas de servidor, algo similar a JSP o ASP.NET, por lo que es muy fácil crear aplicaciones web con XQuery y datos de aplicaciones almacenados en formato XML (Kurt., 2008).

Las Bases de Datos (eXist) permiten adicionalmente almacenar documentos XML, archivos binarios, imágenes, archivos CSS, etc. lo que significa que eXist ha sido diseñado como un sistema administrador de contenido (Content Management System, CMS) (Kurt., 2008). Una ventaja adicional es que los scripts XQuery también pueden ser almacenados en las bases de datos eXist y pueden correr en ellas (o sobre ellas). Así todos los componentes de la aplicación web pueden ser almacenados en la base de datos y ninguno de ellos en el sistema de archivos. O los datos de aplicación pueden ser almacenados en bases de datos y scripts (y otros archivos) en el sistema de archivos.

El desarrollo que se presenta en este artículo utiliza un conjunto básico de las tecnologías XML para implementar XMLStarways: primero se utilizó XML para representar datos, XML Schema para restringir y estructurar los datos, XQuery (Boag, 2010) para consultar y procesar datos y XForms para definir las interfaces de usuario, describir cualquier interfaz y para realizar tareas de manipulación de datos. Todo esto inmerso en XHTML y con CSS para la interfaz de usuario.

Este artículo pretende sembrar entre los constructores y arquitectos de software la necesidad de analizar en los problemas que pronto tendrán que enfrentar debido a las herramientas que usan, y que lejos de ser cuestionadas, gozan del beneplácito colectivo y alabanzas autocomplacientes de quienes las elijen; pero al mismo tiempo, mostrar herramientas de las que pueden valerse para superar estos problemas.

2 XForms-REST-XQuery (XRX)

XRX es una arquitectura de software que se basa en el uso de XML para almacenar la información en el servidor y procesarla en el lado del cliente. Aunque la primera y la última X en la sigla se asocia inicialmente con XQuery para el servidor y XForms para el cliente, últimamente se ha aceptado que estas pueden ser substituidas por cualquier otra tecnología XML que cumpla la misma intención (como por ejemplo XSLT). Lo que sí es una constante es el uso de REST (Representational State Transfer), como la capa que une al cliente con el servidor. La figura 1 detalla las características de la comunicación entre cliente y servidor en las aplicaciones XRX .

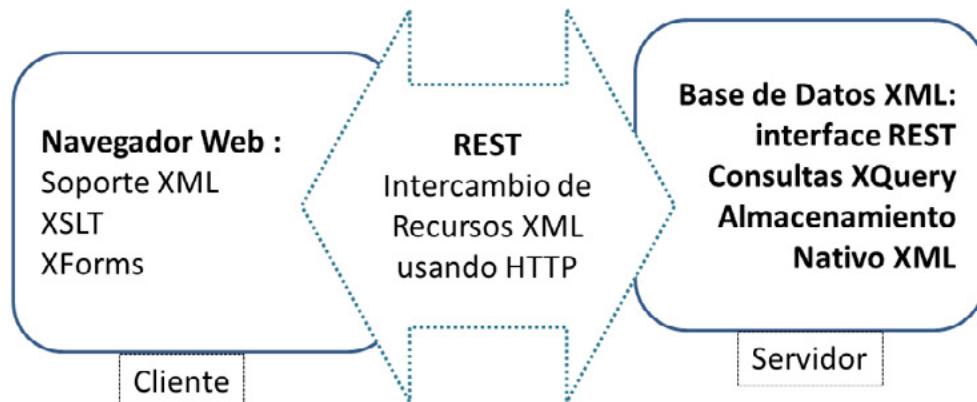


Figura 1. Arquitectura general de una aplicación XRX

La figura 1 muestra la arquitectura (Cliente- Servidor) de la aplicación Web XRX, en la cual se pueden reconocer los siguientes elementos. En el bloque del cliente, se tiene el navegador web, el cual debe soportar XML, XSLT y Xforms, XForms es el formato XML para la especificación de formularios web, diseñado por el W3C para poder definir interfaces de usuario (Boyer, 2009). La comunicación entre el cliente y el servidor se realiza por medio de REST, que permite describir diferentes interfaces web que utilizan XML y HTTP, sin las abstracciones agregadas de los protocolos construidos sobre patrones.

Una característica común a la mayoría de bases de datos XML es que soportan directamente el protocolo HTTP y se puede acceder a ellas como servidores Web que sirven directamente los datos como recursos identificables por medio de una dirección URL. De la misma forma permiten acceder a los programas que gestionan los datos, sin necesidad de plataformas aparte de ejecución del lenguaje de desarrollo elegido.

Este mecanismo donde cada elemento de una aplicación puede accederse a través de un identificador URI usando el protocolo HTTP, se conoce como Representational State Transfer y es la base para creación de aplicaciones Web y especialmente las basadas en XML.

En las bases de datos XML, REST es la interface preferida de comunicación entre los clientes y el servidor, entre otras cosas por su amplia adopción con la popularización de la World Wide Web (WWW) y su simplicidad que va acompañada de profundos fundamentos, como la separación del contenido de su representación.

Dentro de una aplicación XRX cada recurso puede ser: Un documento XML directamente, un Formulario XForms que edita un documento XML, una página (X)HTML obtenida de consultas sobre documentos XML mediante XQuery u otro lenguaje de consulta XML (como XSLT).

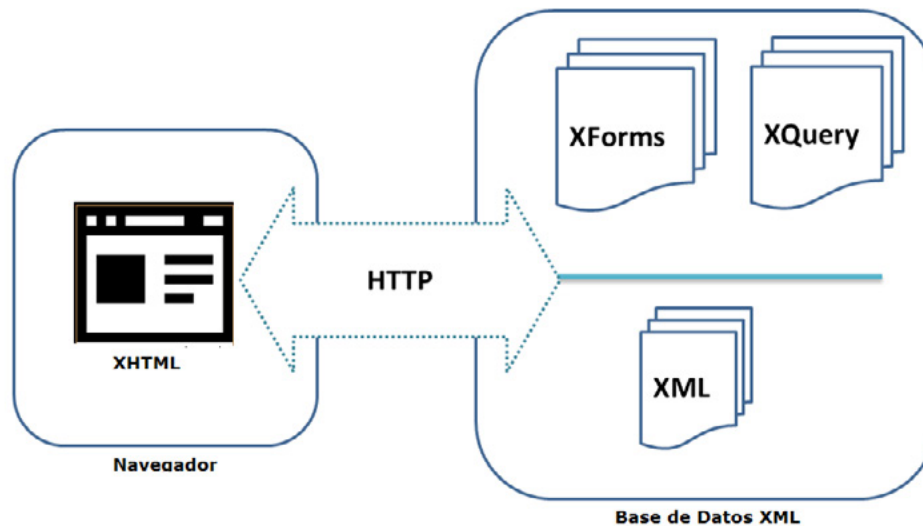


Figura 2. Interacción cliente servidor.

La figura 2 se presenta la interacción, entre el cliente y el servidor. En esta, el Cliente navega a través de la aplicación mediante vínculos a los URLs (Uniform Resource Locators) de los programas y datos que son representados como XHTML. En el Servidor, tanto datos como programas se almacenan por igual como documentos XML, cada uno identificado por un URL único.

Los requerimientos de XML stairways

Este artículo describe la creación de una plataforma de colaboración que atiende a una necesidad real en toda comunidad de desarrolladores: encontrar ayuda para la resolución de problemas, y en particular para la comunidad de desarrolladores XML.

El objetivo de esta plataforma es permitir a quienes tienen problemas o necesitan como se hace algo, preguntar a la comunidad de desarrolladores y obtener respuestas de otros más experimentados. Estas preguntas y respuestas que se convertirán con el tiempo en una base de conocimiento para desarrolladores XML.

La Visión de XML Stairways es “que cualquiera que no sepa cómo hacer algo con XML, sea capaz de hacerlo hoy, basado en problemas previamente resueltos, o mañana, preguntando a otros, si aún no está disponible una solución”.

Los tres factores de éxito de XML stairways son:

1. Está dedicado a la comunidad XML.
2. Está desarrollado 100% usando tecnología XML, específicamente, la arquitectura XRX.
3. Está basado en el concepto de “Escaleras de Aprendizaje”: para aprender una técnica, se aborda una serie de problemas que son presentados en orden incremental de complejidad, con su respectiva solución, a manera de guías paso a paso.

El proceso de construcción de una aplicación XRX

Para la construcción de XML Stairways seguimos una secuencia de pasos, que es común en la creación de aplicaciones XRX y que aquí presentamos a manera de referencia para quienes creen nuevas aplicaciones XRX (Figura 3).

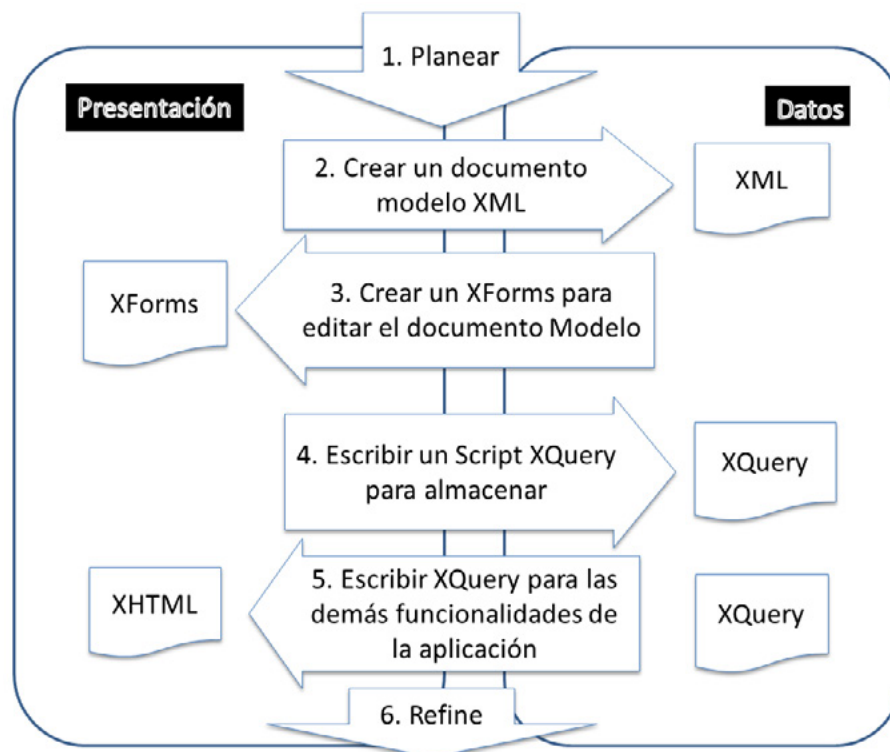


Figura 3. Proceso de creación de una aplicación XRX.

Paso 1: planeación y diseño

Planee su aplicación para que los datos puedan ser representados en XML. Al ocuparse del diseño de una aplicación basada en XML, se debe decidir qué contendrá cada documento almacenado en él. Hay cuatro aproximaciones comunes que los desarrolladores suelen aplicar y que aquí ilustraremos usando ejemplos de las tiras cómicas:

Libro de Mafalda: incluye todo, chistes, historia, fotos. Un diseño de aplicación estilo Libro de Mafalda, es aquel donde se pone absolutamente toda la información que almacenará el sistema en un solo documento XML. En este caso, se crea un elemento raíz que pueda contener a cualquier otro elemento de información de la aplicación.

Si bien puede ser suficiente para aplicaciones pequeñas que manejen pocos datos, tendrán problemas cuando la cantidad de datos se incrementa, especialmente de concurrencia. En nuestra aplicación ejemplo, sería equivalente a poner todos los requerimientos, los objetivos, los roles, las cuentas de usuario, los menús y la configuración en un solo documento XML.

Revista de Condorito: solo contiene Chistes, pero no necesariamente relacionados entre sí. Este es el diseño donde se almacenan en un solo documento, todos los elementos similares. En este caso se crea un elemento raíz que contenga infinitos (unbounded) elementos iguales. Ayuda a separar la funcionalidad de la aplicación, pero sigue teniendo problemas de concurrencia. Si aplicáramos este modelo en nuestra aplicación, tendríamos un solo documento XML que contenga a todos los requerimientos, otro que contenga a las cuentas de usuario, otro los objetivos, otro las cuentas de usuario y así sucesivamente.

Calvin & Hobbes: para entender una historieta, hay que haber leído las anteriores.

En este diseño, se almacenan cada elemento complejo en un documento diferente, pero de tal modo que una sola transacción, puede requerir varios documentos para ser representada. Es equivalente al proceso de normalizar una base de datos.

Si bien representa una gran mejora para la concurrencia, tiene la desventaja de que necesitará manipular varios documentos en cada caso de uso, y la información podría perder su integridad. Si usáramos este modelo, cada requerimiento se dividiría en varios documentos, al igual que los objetivos, los roles y demás.

Olafo: un chiste completo en cada historieta. En este diseño, cada documento contiene toda la información de una transacción, de tal modo que cumple estas dos características: Cohesión: Todos los datos contenidos en el documento están relacionados entre sí y no pueden existir por separado. Integridad: Todos los datos contenidos en el documento están completos y no hacen falta datos de otros documentos para representar la unidad de información.

Este es el modelo recomendado en aplicaciones XRX y que usaremos en XML Stairways: cada problema será un documento independiente que contenga toda la información que lo compone, incluidas las respuestas y comentarios de los usuarios, como se observa en el esquema presentado como figura 4:

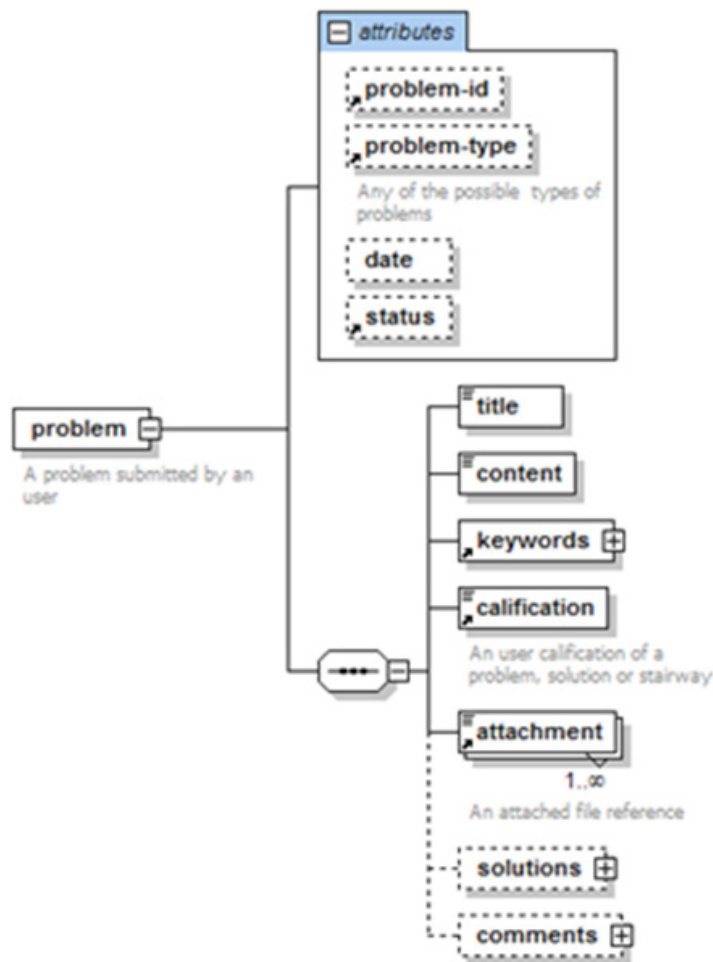


Figura 4. Esquema problema.

Paso 2: creación de un documento modelo

Una vez elegido cuál (o cuáles) serán los esquemas de documento que almacenará nuestra aplicación, el segundo paso es crear un documento en blanco para cada uno de ellos y que servirá como la plantilla a partir de la cual se crearán los recursos que el usuario manipule en el sistema.

Estos documentos modelo pueden ser diseñados desde cero por los creadores de la aplicación usando herramientas de creación de Esquemas XML, pero también podrían ser tomados a partir de esquemas públicos de uso común. En el caso de XML Stairways, diseñamos un esquema XML que representara la

estructura de los problemas y las escaleras de aprendizaje, y a partir de él creamos documentos en blanco que servirán como modelo para los problemas que reporten los usuarios.



Figura 5. Modelo de un problema nuevo.

Paso 3: Construcción del XForms

Construya el XForms para editar el documento. XForms está basado en el principio de separar Contenido y Presentación, tal como se describe en el patrón MVC. El contenido del XForms se representa usando el elemento `<xf:model>`. Este a su vez contiene elementos `<xf:instance>` que permiten indicar los documentos que se editarán en el formulario y el destino donde se enviarán los documentos editados se indica usando el elemento `<xf:submission>`. Igualmente las reglas edición se representan usando elementos `<xf:bind>`.

En el ejemplo el elemento `<xf:model>` quedaría escrito así:

```
<xf:model>
  <xf:instance xmlns="" src="new-problem.xml" id="main-instance"/>
  <xf:submission id="save" method="post" action="save-problem.xq"
    instance="main-instance" replace="instance">
  </xf:submission>
</xf:model>
```

Figura 6. Código del modelo para XForms.

La instancia apunta al documento modelo en blanco que creamos en el paso anterior, y el destino del formulario es un script XQuery que construiremos en el siguiente paso. La Presentación del formulario se hace mediante una combinación de controles visuales XForms y HTML. Cada uno de los controles visuales hace referencia. Para nuestro ejemplo, los controles visuales que permiten editar los datos del problema se representan mediante el siguiente código XForms:

```
<xf.group>
  <xf.input ref="title" class="full-input">
    <xf.label>Problem:</xf.label>
  </xf.input>
  <xf.textarea ref="content" class="multiline"
    mediatype="application/xhtml+xml">
    <xf.label>Describe your Problem:</xf.label>
  </xf.textarea>
  <xf.repeat ref="keywords">
    <xf.label>Keywords</xf.label>
    <xf.input ref="keyword" class="small-input" />
  </xf.repeat>

  <xf.submit submission="save">
    <xf.label>Submit your problem</xf.label>
  </xf.submit>
</xf.group>
```

Figura 7. Código XForms para la edición de problemas.

El resultado que se presentará en el navegador será el siguiente:

The rendered form consists of the following elements from top to bottom:

- A label "Problem:" followed by a single-line text input field.
- A label "Describe your Problem:" followed by a multi-line text area.
- A label "Keywords" followed by a single-line text input field.
- A button labeled "Submit your problem".

Figura 8. Presentación formulario.

Una última anotación: la mayoría de navegadores no soportan XForms directamente. En nuestro proyecto utilizaremos XSLTForms, un plug-in desarrollado en Javascript que permite correr XForms en cualquier navegador moderno. Para utilizarlo, solo basta incluir una instrucción de procesamiento `<?xml-stylesheet ...?>` al inicio de nuestro XForms que pone en marcha una transformación XSLT para convertir el XForms en una página HTML+Javascript.

Paso 4: Crear un punto de llegada para guardar los XForms

Al momento de guardar cada documento, será necesario asignar un identificador único con el cual se guardará en nuestra base de datos XML y con el que se identificará inequívocamente.

Para hacerlo, necesitamos un script XQuery que cumple la tarea de guardado. Si bien el código XQuery que se usa para realizar consultas, es prácticamente igual para todas las bases de datos XML, la tarea de actualización es una característica relativamente nueva y aún hay pequeñas diferencias entre las diferentes implementaciones. Aquí se presenta el código que se utilizaría en eXist-db para guardar nuestro problema:

```
let $doc := request:get-data()
let $collection := '/db/xmlstairways/data'

let $ret := xmldb:login($collection, 'admin', '123456')
let $sid := if ( $doc/*/@problem-id = "" )
  || || || || then rules:next-problem-id()
  else $doc/*/@problem-id
let $ret := xmldb:store($collection, concat($sid, '.xml'), $doc )

let $doc := doc( concat( $collection, '/', $sid, '.xml' ) )

let $ret := update value $doc/*/@problem-id with ($sid)
let $ret := update value $doc/*/@user-id with (usession:user-id())
let $ret := update value $doc/*/@date with (current-dateTime())

return $doc
```

Figura 9. Script XQuery Guardar.

Este fragmento de código añade un identificador tipo UUID al atributo id de cada problema y este valor será el que usaremos para identificarlo en cualquier lugar dentro de la aplicación donde se haga referencia al problema.

Paso 5: conectar las partes

Una vez construidos los formularios de edición de los documentos que componen la aplicación, será necesario incluir una página de inicio con una lista de los problemas, que permita al usuario navegar por XML Stairways. Utilizando un poco de XQuery podemos conseguir un listado de los problemas que el usuario haya creado, con vínculos a los XForms para verlos y crear nuevos problemas.

```

<div class="4u">
  <section class="box box-feature">
    <div class="inner">
      <header>
        <h2>Open problems</h2>
        <span class="byline">
          Waiting for your solutions
        </span>
      </header>
      <ul>
        { for $p in collection(/db/xmlstairways/data)/problem return
          <li>
            <a href="view-problem.xq?id={$p/@problem-id}">
              {$p/title/text()}
            </a>
          </li>
        }
      </ul>
    </div>
  </section>
</div>

```

Figura 10. Código Xquery para listar problemas en StairWyas.

Lo que produce el siguiente resultado en el navegador:

Open problems

Waiting for your solutions

[Use XSLTForms in a page](#)

[Change the size of an xf:input](#)

[Starting with XForms Styling](#)

[Validation email on register](#)

Figura 11. Vista en el navegador.

Paso 6: refinar la aplicación

A partir de aquí el proceso de desarrollo sigue un curso muy similar al de una aplicación web desarrollada con otras herramientas, repitiendo los mismos pasos anteriores, para las demás opciones de captura de datos, y añadiendo características comunes, como:

- Hojas de Estilos
- Páginas informativas (en HTML plano)
- Menús de navegación
- Búsqueda

Es deseable en toda aplicación moderna, tener un diseño que se adapte a diferentes formatos de presentación en diferentes dispositivos (escritorio, tabletas, teléfonos), una característica popularmente conocida como responsive design, y que cumpla de una vez con el estándar HTML5. En XML Stairways nos basamos en un template HTML5 gratuito tomado de <http://html5up.net> que ya incluye las funcionalidades de responsive design y lo adaptamos para que se use en todas las páginas del sitio. Para ello creamos un módulo XQuery llamado style.xqm que contenga funciones para agregar la apariencia sin mezclarla con la lógica de la aplicación.

Paso 7: Ir más allá

Al usar XML como formato de almacenamiento de la información, se pueden incluir con mayor facilidad características muy apreciadas por los usuarios y que usando los caminos tradicionales llevaría bastante esfuerzo conseguir, como por ejemplo:

- Control de Versiones
- Comparación de diferencias
- Replicación
- Conversión a otros formatos
- Interface Web Service tipo REST
- Caching

Los detalles de cómo implementar estas características, así como de otras funcionalidades comunes, pueden ser encontrados en el WikiBook de XRX y de XForms.

Resultados

Este proyecto es la base sobre la que nos apoyaremos para divulgar el desarrollo basado en XRX, no solo por su despliegue como una comunidad de interés alrededor de XML, sino también por ser su creación una muestra de cómo puede aplicarse XRX. El portal www.xmlstairways.com ahora permite a quienes se involucren en el desarrollo basado en XML, encontrar una comunidad dispuesta a ayudarles a resolver

sus problemas, desde los más sencillos hasta los más complejos y al mismo tiempo ir construyendo una base de conocimientos pública que sirva a quienes apenas comiencen a crear aplicaciones donde usen XML y todas sus tecnologías relacionadas.

El proyecto Open Source XML Stairways, alojado de manera pública en SourceForge, es un punto de contacto con desarrolladores experimentados de todo el mundo que realizan aportes al proyecto y al hacerlo comparten su experiencia con todos los que se involucran en el proyecto, también ha servido para que grupos de estudiantes de la Universidad de Medellín aprendan a partir de una aplicación ya creada, y al mismo tiempo pongan en práctica sus conocimientos haciendo aportes al proyecto.

Aunque el portal XML Stairways está dirigido a una comunidad de usuarios XML, el proyecto pueden adaptarse fácilmente para ser usado en otros portales del mismo formato, pero enfocados a diferentes audiencias, y no solo relacionadas con las tecnologías de información, sino de cualquier área del conocimiento, donde se pueda aprovechar el conocimiento generado al resolver problemas como plataforma de aprendizaje.

Conclusiones y trabajo futuro

Los paradigmas solo son visibles cuando están siendo reemplazados, luego, cuando se los nuevos se hacen ampliamente aceptados damos, por ciertos todos los supuestos en que se basan y creamos mecanismos para que sigan siendo aplicables aun cuando las contracciones empiezan a ser evidentes.

El actual paradigma de desarrollo que usamos, fundamentado en el supuesto de que las bases de datos relacionales y los lenguajes orientados a objetos son las herramientas idóneas para construir aplicaciones, sumado al supuesto de que la combinación de dos tecnologías exitosas crean otra tecnología exitosa, ha comenzado a evidenciar sus contradicciones. Ejemplo de ello ha sido el siempre presente Error de Impedancia Objeto-Relacional (Mismatch, 2008) y más recientemente los problemas de escalabilidad (et, 2009) con la llegada de la Web.

En ambos casos la comunidad del desarrollo de software ha formulado soluciones basadas en la construcción de cada vez más artefactos, que permitan el acople del modelo de objetos al relacional y permitan también que las aplicaciones escalen. La construcción de estos artefactos añade trabajo adicional al proceso de desarrollo de software que reduce la agilidad con que se lleva a cabo la construcción de la aplicación.

Este artículo ha abordado la creación de nuevas arquitecturas, donde no se usen bases de datos relacionales ni lenguajes orientados a objetos, y en su lugar se utilicen bases de datos XML, lenguajes funcionales (XQuery) y declarativos para la interfaz de usuario (XForms), con la característica de compartir un solo modelo común para la representación de datos (documentos XML) lo que garantiza que el trabajo de codificación que se lleve a cabo sea realmente el necesario para implementar la funcionalidad, y no se tengan que dedicar esfuerzos a escribir artefactos que no añaden valor al producto, sino que simplemente

permiten seguir usando las herramientas que tradicionalmente se usan, pero incompatibles entre sí.

El cambio de paradigma que representa XRX implica cuestionar y desechar tecnologías ampliamente utilizadas, con sus costos asociados:

- Des aprendizaje de las actuales técnicas de desarrollo.
- Adquisición de nuevas herramientas.
- Entrenamiento en el uso de las nuevas herramientas.
- Adopción de nuevas prácticas de modelamiento y diseño.
- Construcción de prototipos para ayudar al aprendizaje.

Pero de otro lado, estos costos se ven compensados por los siguientes beneficios:

- Menor esfuerzo de construcción requerido (menos tiempo y menos personal).
- Capacidad de manejar estructuras de datos de alta complejidad.
- Sistemas interoperables per-se, sin necesidad de construir artefactos de interfaz con otros sistemas, gracias a REST.
- Escalabilidad intrínseca gracias a XQuery y su naturaleza side-effect-free (effect, 2010) (Features, 2008).

La implementación de XRX hace fundamental la práctica de la Arquitectura de Información al inicio del proceso de desarrollo. Tradicionalmente solo se piensa en ella cuando se deben desarrollar interfaces entre sistemas, y muchas veces se hace cuando la construcción del sistema ya va bien adelantada, pero al adoptar XRX, la arquitectura de información se convierte en la pieza central sobre la que se construye toda la aplicación.

Como trabajo futuro se propone, trabajar en la construcción de un Framework que facilite la ejecución de las tareas comunes en la creación de aplicaciones XRX, como el diseño de documentos modelo y la generación automática de XForms a partir de esos documentos. En este sentido, ya estamos vinculados al proyecto Open Source Schema2XForms (<http://schema2xforms.sourceforge.net/>) iniciado por Steve Cameron y que persigue este objetivo, Parte de este trabajo conjunto ya fue expuesto en la conferencia Balisage el pasado mes de agosto y publicado en sus Proceedings (Cameron, 2013).

Pero sabemos que el mayor reto es la difusión en nuestro medio, por eso continuaremos trabajando para divulgar material en nuestro idioma, ayudar a quienes adopten XRX y crear una base de conocimiento y experiencia que ayude a minimizar la resistencia a su adopción.

Agradecimientos

A la Universidad de Medellín por facilitar la aplicación de estrategias de enseñanza basadas en juegos en los cursos de pregrado.

A Alianza Regional en TIC Aplicada (ARTICA), por su apoyo en la financiación de estudios de formación de alto nivel.

Stephen Cameron (www.collinta.com.au) por su apoyo al lanzamiento de este proyecto y todas sus ideas y aportes.

Dan McCreary y Alain Couthures por su trabajo en la construcción de las herramientas que hicieron posible este proyecto y su permanente disposición a resolver nuestras inquietudes.

Referencias

Boag, S.E. (2010). XQuery 1.0: An XML Query Language (Second Edition).

Boyer, J.M. (10 de 2009). XForms 1.1. Recuperado el 01 de 09 de 2013, de <http://www.w3.org/TR/xforms11/>.

Cameron, S.A. (6 de 08 de 2013). A Data-Driven Approach using XForms for Building a Web Forms Generation Framework: The Markup Conference 2013, Montréal, In Proceedings of Balisage. Recuperado el 01 de 09 de 2013, de Balisage 2013: <http://www.balisage.net/Pr>

Effect, S. (2010). Side effect (computer science) . Recuperado el 01 de 09 de 2013, de [http://en.wikipedia.org/wiki/Side_effect_\(computer_science\)](http://en.wikipedia.org/wiki/Side_effect_(computer_science))

Et, S.M. (2009). OLTP Through the Looking Glass, and What We Found There. Recuperado el 01 de 09 de 2013, de: <http://cs-www.cs.yale.edu/homes/dna/papers/oltpperf-sigmod08.pdf>

EXist. (2011). eXist. Obtenido de <http://exist-db.org>

Features, X. (2008). XQuery Features. Recuperado el 01 de 09 de 2013, de <http://en.wikipedia.org/wiki/XQuery#Features>

Karunamu, F.K. (2012). A novel architecture for Web service composition?. Journal of Network and Computer Applications, 787–802.

Kurt., C. (2008). Metaphorical Web and XRX. Obtenido de <http://broadcast.oreilly.com/2008/09/metaphoricalweb-and-xrx.html>

Law, M.C. (09 de 2010). Mc Crearys - Law. Recuperado el 01 de 09 de 2013, de <http://datadictionary.blogspot.com>

McCreary, D. (05 de 2008). XRX: Simple, Elegant, Disruptive. Recuperado el 01 de 09 de 2013, de: http://www.oreillynet.com/xml/blog/2008/05/xrx_a_simple_elegant_disruptiv_1.html.

Mismatch, O.R. (2008). Object Relational Impedance Mismatch . Recuperado el 01 de 09 de 2013, de: http://en.wikipedia.org/wiki/Object-relational_impedance_mismatch NoSQL. (2010). NoSQL Databases. See. Recuperado el 01 de 09 de 2013, de <http://nosql-database.org/>.

Pemberton, S. (2006). What do we want from the web?,. Recuperado el 01 de 09 de 2013, de <http://homepages.cwi.nl/~steven/Talks/2013/07-xx-web/>.

Somerville. (2010). INGENIERÍA DEL SOFTWARE GENERALIDADES. Ed MCGRAW-HIL.

Technologies, X. (2008). XML Technologies. Recuperado el 01 de 09 de 2013, de <https://community.emc.com/community/edn/xmltech>.

Wade, A. E. (2010). Hitting The Relational Wall, An Objectivity, Inc. Recuperado el 01 de 09 de 2013, de <http://odbms.org>

Wikipedia. (2010). History of XRX. [http://en.wikipedia.org/wiki/XRX_\(web_application_architecture\)#History_of_XRX](http://en.wikipedia.org/wiki/XRX_(web_application_architecture)#History_of_XRX). Recuperado el 01 de 09 de 2013, de: [http://en.wikipedia.org/wiki/XRX_\(web_application_architecture\)#History_of_XRX](http://en.wikipedia.org/wiki/XRX_(web_application_architecture)#History_of_XRX)

XRX. (2011). XRX (web application architecture). Recuperado el 01 de 09 de 2013, de <http://en.wikibooks.org/wiki/XRX>

XSLTForm. (2011). XSLTForms Web site, 2011. Recuperado el 01 de 09 de 2013, de <http://www.agencexml.com/xsltforms>



A

nálisis de exámenes en carreras de Sistemas mediante procesos de explotación de información

Pollo-Cattaneo, M.F¹; Deroche, A¹, Raus, N¹, Lujan, F¹, Vegega, C¹, Pytel, P^{1,2}

¹ Grupo de Estudio en Metodologías de Ingeniería de Software. Facultad Regional Buenos Aires. Universidad Tecnológica Nacional. Argentina. pollo@posgrado.frba.utn.edu.ar, ariel.deroche@gmail.com

² Grupo Investigación en Sistemas de Información. Departamento Desarrollo Productivo y Tecnológico. Universidad Nacional de Lanús. Argentina. ppytel@gmail.com

Resumen

Las evaluaciones de los alumnos pueden ser consideradas como una herramienta para medir la eficacia del proceso de enseñanza y aprendizaje de una asignatura. Para el análisis de estos exámenes, se suelen utilizar técnicas estadísticas tradicionales que resultan ineficientes para detectar los problemas encontrados en el proceso de enseñanza y la detección de relaciones implícitas entre los temas impartidos en clase. Teniendo en cuenta este contexto, el presente trabajo propone la aplicación de procesos de Explotación de Información sobre las evaluaciones de los alumnos para así extraer patrones de conocimiento que les permitan a los docentes, detectar problemas en la enseñanza y tomar decisiones de mejora en sus asignaturas. Para demostrar la utilidad de los resultados aportados por estos procesos se presentan dos casos de estudio realizados en el ámbito de la carrera 'Ingeniería en Sistemas de Información' de la Universidad Tecnológica Nacional de Argentina.

Palabras clave: análisis de proceso de evaluación, educación y tecnología, enseñanza, explotación de información.

Introducción

La Universidad Tecnológica Nacional (UTN) es una de las principales universidades estatales argentinas para la formación de ingenieros contando con facultades regionales en todo el país. Dentro de las carreras ofrecidas, se encuentra 'Ingeniería en Sistemas de Información', que tiene como objetivo formar profesionales de sólida capacidad analítica para la interpretación y resolución de problemas mediante el empleo de metodologías de sistemas y tecnologías de procesamiento de información [1]. Para ello, incluye tanto asignaturas sobre ciencias y tecnologías básicas (donde se pueden encontrar matemática, física, química y fundamentos de la informática) como también asignaturas especializadas sobre metodologías, técnicas y herramientas para planificar, dirigir, ejecutar y controlar el desarrollo de proyectos orientados a los sistemas de información en general y sistemas software en particular [2].

En Argentina, todas las carreras asociadas a las tecnologías de información, se encuentran reguladas por la resolución 786/2009 del Ministerio de Educación [3]. La misma especifica las actividades profesionales reservadas a cada tipo de carrera, sus contenidos curriculares básicos, la carga horaria mínima y los estándares para su acreditación. Estos procesos de acreditación buscan garantizar la calidad de la formación académica en las instituciones universitarias estatales y privadas para trabajar a favor de la equidad y la disminución de las asimetrías que afectan la calidad institucional y académica [4]. Uno de los elementos considerados en la acreditación son las evaluaciones de los alumnos. Estas deben ser congruentes con los objetivos y metodologías de enseñanza previamente establecidos por los docentes de cada asignatura.

Teniendo en cuenta este contexto, el presente trabajo tiene como objetivo demostrar la utilidad de la aplicación de procesos de Explotación de Información sobre las evaluaciones de los alumnos, para así extraer patrones de conocimiento que le permitan a los docentes detectar problemas en la enseñanza y, tomar decisiones de mejora en sus asignaturas. Para ello, primero se describe el problema detectado (sección 2), se define la solución propuesta (sección 3) junto con dos casos de estudio (sección 4). Finalmente se indican las conclusiones generales obtenidas y futuras líneas de trabajo (sección 5).

Definición del problema

La Declaración de la Conferencia Regional de Educación Superior de América Latina y el Caribe (CRES), también conocida como Declaración de Cartagena, identifica la necesidad de contar con herramientas positivas para controlar el proceso de enseñanza de las tecnologías de información [5]. Una de las herramientas que se pueden utilizar en este sentido son las evaluaciones parciales y finales que los alumnos deben rendir para aprobar una asignatura.

Según lo enunciado en [6], las evaluaciones pueden ser consideradas como métodos para adquirir y elaborar las comprobaciones que son necesarias a fin de perfeccionar tanto el aprendizaje del estudiante como la enseñanza. En otras palabras, los exámenes se pueden considerar como un instrumento para “medir” la eficacia del proceso de enseñanza y aprendizaje de los alumnos en la asignatura. De esta manera, el docente puede ajustar sus esfuerzos con miras de preservar y aumentar su eficiencia antes de que sea demasiado tarde [7].

Desde finales del siglo XIX se han utilizado técnicas estadísticas para analizar los resultados de los exámenes y evaluar el progreso de los alumnos. Sin embargo, estas técnicas suelen ser ineficientes para detectar los problemas encontrados en el proceso de enseñanza y la detección de relaciones implícitas entre los temas impartidos en clase. Este uso limitado se suele generar por una falta general de comprensión de los conceptos básicos de la medición (es decir, validez, confiabilidad, correlación, error típico, entre otros). Por lo tanto, las asignaturas interesadas en perfeccionar sus programas deben contar con una persona competente en dicha esfera que se encargue de organizar y aplicar las técnicas estadísticas.

Solución propuesta

A partir de varios estudios realizados en el ámbito de la educación [8-11], se ha comprobado la eficiencia positiva de la aplicación de Explotación de Información para resolver problemas en dicho ámbito. La Explotación de Información es una sub-disciplina de la Informática que aporta a la Inteligencia de Negocios [12], las herramientas para la transformación de grandes masas de datos en conocimiento [13] mediante la búsqueda de patrones no triviales, y de regularidades importantes, en grandes masas de información [14].

Para realizar dicha búsqueda se aplican los denominados Procesos de Explotación de Información [15]. A partir de la identificación del problema que se desea resolver, y las características de los datos disponibles, se selecciona el proceso que mejor se adecue. Cada proceso tiene asociado un conjunto de técnicas o algoritmos de Minería de Datos para obtener los resultados necesarios. Muchas de esas técnicas vienen del campo del Aprendizaje Automático [16], por lo que los patrones de conocimiento son obtenidos automáticamente. Esto significa que, en estos procesos, no existe la necesidad de formular hipótesis previas aunque es imprescindible interpretar con cuidado los resultados obtenidos. En este sentido, esto lo diferencia de las técnicas estadísticas tradicionales, donde se debe primero formular una hipótesis de trabajo de acuerdo al problema que se quiere resolver, luego se aplican las operaciones necesarias, y con los resultados obtenidos se confirman, o refutan, la hipótesis propuesta. Para ilustrar la diferencia entre las técnicas estadísticas tradicionales y la Minería de Datos, Pyle [17] indica que si para resolver un problema se debe navegar en un vasto océano de datos del cual se intentará pescar (o extraer) el conocimiento necesario para solucionarlo, la aplicación del análisis estadístico tradicional es como un niño pescando con una caña a la orilla del océano. En cambio, la Minería de Datos permite obtener el conocimiento necesario como si fuera un mediomundo o velo de pesca, ya que genera muchos y diversos patrones de conocimiento en forma directa y automática para luego ser analizados.

Casos de estudio

Para demostrar los resultados aportados de aplicar procesos de Explotación de Información sobre exámenes de asignaturas de grado, se utilizarán dos casos de estudio realizados en el ámbito de la Universidad Tecnológica Nacional (UTN). En el primer caso (sección 4.1) se ha hecho un análisis de la segmentación de exámenes parciales provenientes de dos regionales de dicha universidad, mientras que en el segundo (sección 4.2), se aplica la ponderación de temas de los exámenes finales de alumnos que han cursado con diferentes profesores.

Segmentación de la evaluación de exámenes parciales

El primer caso es una continuación del estudio realizado en [18], para la asignatura ‘Inteligencia Artificial’ [19] que es dictada por el mismo equipo docente en la Facultad Regional Buenos Aires (FRBA) y la Facultad Regional La Plata (FRLP) de la UTN. Aunque el plan de estudio, la planificación de las clases y los temas impartidos en clase son similares en ambas regionales, hay grandes diferencias en cuanto a los resultados de aprobación de los exámenes parciales. En FRLP el porcentaje de aprobados es menor al 15%, mientras que en FRBA es de casi el 57%. En ambas regionales se utilizan exámenes con 21 preguntas de opción múltiple donde sólo una de las opciones es la correcta. Para aprobar, el alumno debe tener por lo menos 13 preguntas bien respondidas. Los temas incluidos en el examen parcial son los siguientes:

- Teoría, evaluada mediante cuatro preguntas teóricas sobre conceptos relacionados a Inteligencia Artificial y la Ingeniería del Conocimiento.
- Modelos de Arquitecturas de Sistemas Inteligentes, evaluados mediante cuatro preguntas sobre conceptos y características de distintos tipos de sistemas inteligentes como Redes Neuronales Artificiales, Sistemas Expertos y Algoritmos Genéticos.
- Métodos de Búsqueda, mediante tres ejercicios prácticos sobre el recorrido de árboles de estados utilizando diferentes métodos.
- Emparrillado y Análisis de Protocolos, donde para cada método de educación de conocimiento el alumno debe resolver tres ejercicios.
- Deducción Natural y Traducción, incluyen cuatro ejercicios prácticos sobre lógica de primer orden.

En el estudio anteriormente mencionado, los exámenes parciales de la Facultad Regional Buenos Aires fueron analizados para identificar la relación entre los temas y sus consecuencias en la aprobación, o no, de los exámenes. Pero al considerarla discrepancia de aprobación con respecto a la Facultad Regional La Plata, los profesores de la asignatura han solicitado realizar un nuevo estudio para determinar el perfil de los alumnos evaluados de acuerdo a su desempeño en el examen. Esto permitiría identificar la forma

en que se pueden agrupar los alumnos, de acuerdo a sus respuestas en los parciales y, de esta manera, determinar mejoras en la enseñanza y la selección de los temas a ser evaluados en los recuperatorios.

Para dar solución a este problema, se aplica el proceso de Explotación de Información de “Descubrimiento de Reglas de Pertenencia a Grupos” [15]. Este proceso utiliza mapas auto-organizados para clasificar patrones de entrada en grupos utilizando la Red Neuronal Artificial Kohonen en su variante SOM [20]. Luego, se aplica el algoritmo de inducción de la familia TDIDT C4.5 [21] para determinar las reglas que describen a cada grupo. El esquema de este proceso se puede visualizar en la figura 1.

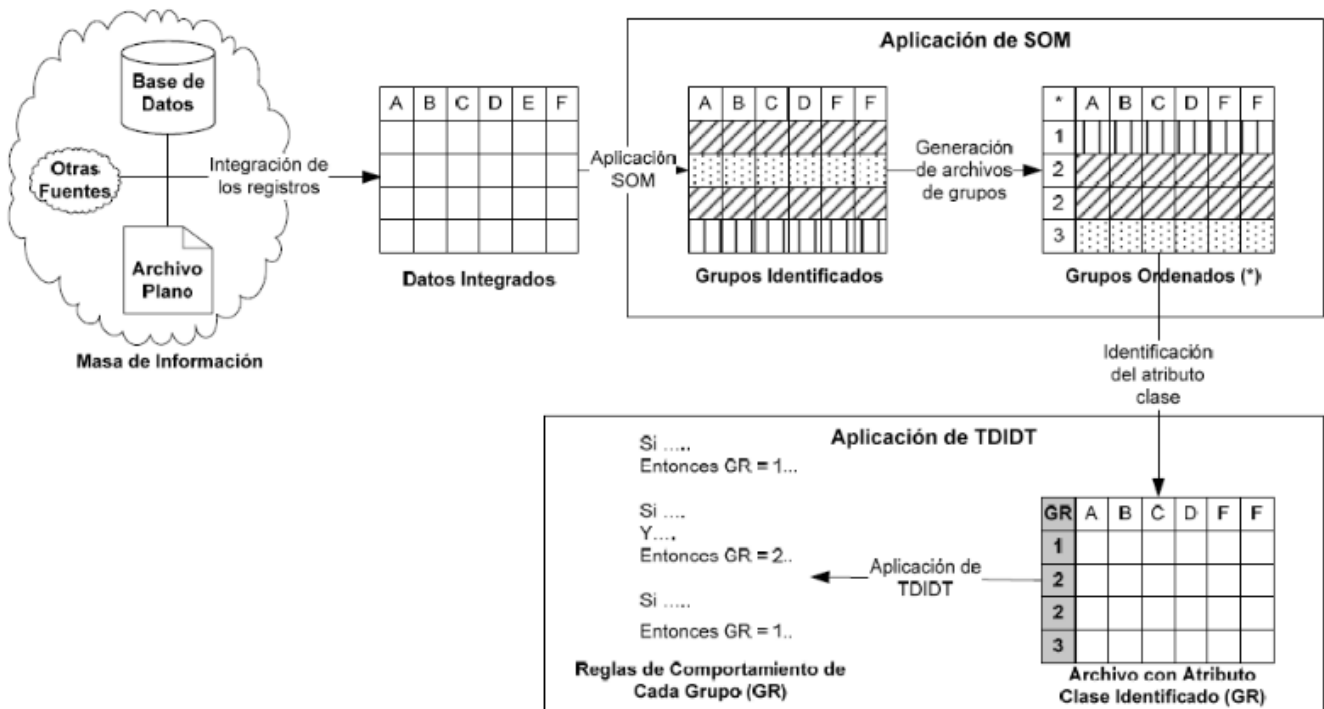


Figura 1. Esquema del proceso de descubrimiento de reglas de pertenencia a grupo.

Se cuenta con los datos de 308 exámenes parciales de la Facultad Buenos Aires (correspondientes al primer cuatrimestre del año 2011 y los dos cuatrimestres del 2012), y 43 exámenes de la Facultad La Plata (para sólo el segundo cuatrimestre del 2012). Estos 351 exámenes parciales han sido normalizados, determinando por tema la cantidad de preguntas correctas, incorrectas y sin contestar. Debido a la falta de espacio, el detalle sobre la preparación de los datos y resultados obtenidos se pueden encontrar en el reporte técnico [22].

Como resultado de aplicar el proceso con la herramienta Tanagra [23], se han generado 5 grupos que se pueden visualizar en la tabla 1 (indicando la cantidad de exámenes por cada grupo y porcentaje por regional).

Tabla 1. Distribución de grupos determinados por el proceso.

Grupo SOM	Cantidad de Exámenes	% Facultad Buenos Aires (FRBA)	% Facultad La Plata (FRLP)
Grupo 1	65	47,70	52,30
Grupo 2	17	100,00	0,00
Grupo 3	67	91,00	9,00
Grupo 4	132	97,70	2,30
Grupo 5	70	100,00	0,00

Una vez identificados los grupos, se aplica el algoritmo de inducción para establecer las reglas de pertenencia a cada uno de los mismos. Como resultado, se obtienen 25 reglas utilizando el atributo del grupo como atributo objetivo. Con estas reglas se identifican las características de los grupos obtenidos, teniendo en cuenta diferentes condiciones. A continuación, se agrupan las reglas por grupo y se analizan, a partir de ellas, las características de cada uno:

El Grupo 1: incluye a los alumnos que desaprobaron los parciales del año 2012 y que presentan algunas de las siguientes características:

- Cursaron el día lunes en FRBA, y respondieron una o más preguntas del tema Teoría correctamente, o
- cursaron en el 1er cuatrimestre los días miércoles(FRBA) o sábado (FRLP), no respondieron bien suficientes preguntas de los temas Emparrillado y todas mal del tema Deducción Natural, o
- cursaron los miércoles (FRBA) o los sábados (FRLP) en el 2do cuatrimestre, además de las condiciones anteriores, respondieron bien varias preguntas pero pocas del tema Arquitectura de Sistemas Inteligentes.

Este grupo lo conforman 65 parciales desaprobados, de los cuales 34 (el 52,3%) corresponden a la Regional La Plata. Debido a que, en total, se cuentan con 37 parciales desaprobados de esa regional, este grupo incluye casi la totalidad de los parciales desaprobados de esa regional (es decir casi el 92%).

El Grupo 2: incluyen los parciales de alumnos que cursaron en el 1er cuatrimestre del 2011 correspondientes en su totalidad a Buenos Aires. Pertenecen a este grupo:

- Por un lado, los parciales desaprobados que poseen una o más preguntas del tema Traducción y dos o más preguntas bien de Teoría, y
- por el otro lado, los parciales aprobados que poseen dos o más preguntas bien respondidas de Emparrillado, una o más preguntas bien de Traducción y ninguna bien respondida para los temas Deducción Natural y Métodos de Búsqueda.

El Grupo 3: lo conforman los parciales que en su gran mayoría pertenecen a Buenos Aires para el año 2012. Este grupo es bastante heterogéneo, ya que los parciales presentan características bastantes diferentes. Esto podría significar que dicho grupo incluye los parciales que no pudieron ser incluidos en los otros grupos. Entre sus características principales se destacan:

- Para los parciales aprobados incluidos en este grupo (aproximadamente el 59%) tienen la mitad o más de preguntas correctas en Análisis de Protocolos, Emparrillado, Arquitecturas de Sistemas Inteligentes y/o Traducción.
- del 41% de los parciales restantes, el alumno desaprueba a pesar de que algunas preguntas correctas en temas como Teoría, Emparrillado, Métodos de Búsqueda y/o Traducción.

El Grupo 4: contiene parciales aprobados de los años 2011 y 2012, siendo la mayoría de Buenos Aires. En todas las reglas de este grupo se pueden notar muchas semejanzas con las reglas de aprobación obtenidas en el estudio anterior realizado en [24]. Sus condiciones están asociadas a la cantidad de preguntas bien respondidas de los temas principales de la asignatura: Teoría, Análisis de Protocolos, Arquitecturas de Sistemas Inteligentes y Métodos de Búsqueda. La única condición que se puede observar en todas las reglas es sobre la cantidad de preguntas bien respondidas del tema Traducción.

El Grupo 5: contiene parciales del año 2011 de Buenos Aires. La mayoría de los parciales de este grupo cumplen las siguientes condiciones:

- Poseen una o más preguntas bien respondidas de Traducción, hasta una pregunta bien respondida de Emparrillado y el resultado es aprobado, o
- tienen hasta dos preguntas bien respondidas de Teoría y el resultado del mismo es desaprobado, o
- tienen entre dos y cuatro preguntas bien respondidas de Emparrillado, hasta dos preguntas bien de los temas Teoría y Arquitectura de Sistemas Inteligentes, dos o más bien de Métodos de Búsqueda, una o más de Traducción y el resultado del parcial resulta aprobado.

A partir de la segmentación de los exámenes parciales y su posterior descripción, ha sido posible identificar diferentes perfiles de los alumnos. Además, se ha descubierto que la aprobación de los alumnos no depende exclusivamente de la regional en que cursan los alumnos, sino de su capacidad

para responder las preguntas de los distintos temas. Esto significa que la cantidad de desaprobados no se genera por el lugar donde se dicta la asignatura, sino por problemas de estudio. Por otro lado, es interesante destacar como la existencia de diferentes temas de la asignatura que, a priori, no fueron catalogados como importantes en el estudio anterior (como era el caso de Traducción), ahora tiene mayor relevancia para caracterizar los grupos generados. Por otro lado, temas que influyen mucho en la aprobación (como Análisis de Protocolo), ahora no resultan tan importantes para la clasificación y división de grupos.

Ponderación de la evaluación de exámenes finales

El segundo caso corresponde al análisis de exámenes finales de la asignatura ‘Sistemas y Organizaciones’ [24] en la Facultad Regional Buenos Aires. Esta es la asignatura integradora troncal del primer año que introduce al alumno en las labores y competencias que deberá desempeñar como ingeniero.

Debido a la gran cantidad de alumnos que cursan esta asignatura (aproximadamente cada año se inscriben 950 alumnos), la cátedra se encuentra formada por 15 profesores. A pesar de que se han definido un conjunto de guías para impartir los temas en forma homogénea, se ha notado ciertas diferencias en los resultados de los alumnos al momento de rendir el examen final común (sin importar el profesor con el que cursaron la asignatura). En promedio, de los 550 alumnos que pasan el curso en forma regular, sólo el 82% aprueban los exámenes finales. Sin embargo este comportamiento general no es el mismo para todos los alumnos, ya que fluctúa entre el 50 y 95% dependiendo con qué profesor hayan cursado. En este contexto, la titular de la cátedra desea realizar un análisis detallado sobre los finales intentando identificar cuáles son los temas más relevantes a la hora de determinar si se aprueba, o no, el examen final. De esta forma, se esperan reconocer las relaciones implícitas existentes entre los temas considerados y, homogeneizar el dictado por todos los profesores de la cátedra. Los temas desarrollados en la asignatura que son evaluados en los exámenes finales son los siguientes:

- Metodología de Sistemas de Información, donde el alumno debe responder dos preguntas donde se le pide describir el objetivo, actividades principales, técnicas y herramientas de las etapas de la metodología.
- Teoría, incluyen cuatro preguntas teóricas de tipo verdadero o falso sobre todos los conceptos de la asignatura.
- Circuitos Administrativos, comprende cuatro preguntas teóricas sobre los circuitos básicos de una organización genérica (Compras, Ventas, Pagos, Cobranzas y Producción), y normas de control interno.
- Cursogramas, consiste en señalar cinco errores de un ejercicio resuelto mediante esta técnica gráfica.

Debe notarse que todos los exámenes finales incluyen preguntas de Metodología y Teoría pero los temas de Circuitos Administrativos y Corsogramas son excluyentes. Si se realizan preguntas sobre Circuitos Administrativos no se incluyen de Corsogramas, y viceversa. Es decir, existen dos tipos de final: los que evalúan Metodología, Teoría y Circuitos Administrativos, y los que evalúan Metodología, Teoría y Corsogramas.

Debido a que se quieren identificar las condiciones para obtener un resultado ya conocido junto con los factores que lo influyen en mayor medida, el proceso de Explotación de Información utilizado es el de “Ponderación de Reglas de Comportamiento” [15]. A partir de un conjunto de atributos se utiliza primero el algoritmo de inducción C4.5 [21], para descubrir cómo el atributo clase puede ser determinado a partir del resto de los atributos. El resultado de esta técnica es un conjunto de reglas donde el atributo clase aparece en el conculyente y, los otros atributos, en las condiciones del antecedente. Luego, se aplica el algoritmo Naive Bayes [25], para determinar la ponderación (medida) de cada uno de los atributos de sobre el atributo clase. El esquema general de este proceso se puede observar en la figura 2.

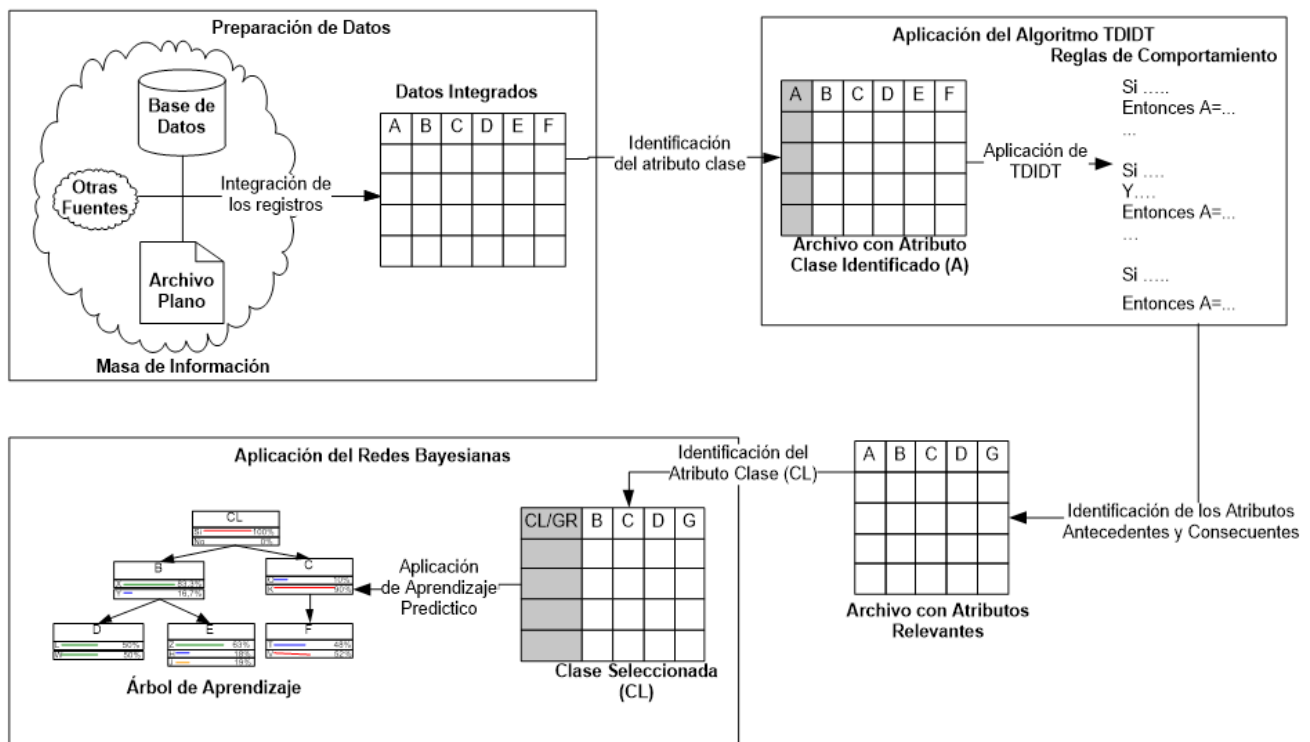


Figura 2. Esquema del proceso de ponderación de reglas de comportamiento.

Para aplicar este proceso primero se han utilizado los 449 exámenes finales correspondientes al período comprendido entre octubre del 2012 y febrero del 2013. Estos exámenes han sido preparados sumando por tema la cantidad de respuestas correctas (un punto) y regulares (medio punto). El detalle sobre la preparación de los datos y resultados obtenidos se puede encontrar en el reporte técnico [26].

A partir de las 12 reglas obtenidas y su posterior análisis se puede observar que:

- Las preguntas sobre Cursograma y Circuitos Administrativos no tienen, por sí mismas, mucho peso para determinar la aprobación del examen.
- Las preguntas decisivas para decidir la aprobación, o no, del final corresponden a las de Teoría. Si se tienen por lo menos tres de las cuatro preguntas incorrectas entonces se desapueba, sin importar cómo ha respondido en los otros temas.
- Por otro lado, aunque en el tema troncal de la asignatura (Metodología de Sistemas de Información) se posean las dos preguntas mal respondidas, o una regular y la otra mal, (es decir menos de la mitad bien respondidas), si se tienen por lo menos dos correctas (o cuatro regulares) en Teoría y en Circuitos Administrativos se puede aprobar el examen. Pero, esto no sucede cuando se tiene la misma cantidad de preguntas con Teoría y Cursograma (en este caso, el parcial es desaprobado).

Luego, se aplica el algoritmo de ponderación sobre los temas más importantes identificados por las reglas (es decir, Metodología de Sistemas de Información y Teoría), para determinar cómo éstos influyen sobre la aprobación, o no, según el tipo de examen final. En la tabla 2, se representa en forma gráfica, la influencia que cada uno de estos temas, sobre la aprobación del final en forma independiente del resto.

A partir de los gráficos de la tabla 2 se puede detectar que:

- En general, las preguntas de Metodología de Sistemas de Información tienen una influencia importante en la aprobación del examen. Al tener, por lo menos, una bien respondida y otra regular (es decir por lo menos 1,5 puntos), la probabilidad de aprobación supera el 50%. En los finales que tienen preguntas de Circuitos Administrativos, esta incidencia es mayor: teniendo 1,5 puntos la probabilidad de aprobación llega al 100%. Por otro lado, cuando en este tema se tiene menos de 1 punto (es decir sólo una pregunta regular o todas mal respondidas), la probabilidad de desaprobación es mayor al 85% para todo tipo de final.

Tabla 2: Resultados de la ponderación de los atributos por tipo de final.

	Preguntas de Metodología de Sistemas de Información	Preguntas de Teoría
Análisis de Finales con preguntas de Cursogramas		
Análisis de Finales con preguntas de Circuitos Administrativos		
Análisis General		
		<p>■ Aprobado</p> <p>■ Desaprobado</p>

- Con respecto a la Teoría, independientemente del tipo de final, cuando se tiene por lo menos tres respuestas mal respondidas (es decir con un valor menor o igual a 1 punto) de Teoría, se tiene una probabilidad de desaprobación superior al 90%. En cambio, para tener una probabilidad de aprobar cercana al 50%, se debe tener por lo menos 2,5 puntos en este tema (es decir por lo menos dos bien respondidas y otra regular). Es importante aclarar que, en los exámenes finales con Circuitos Administrativos considerados en este estudio, sólo dos exámenes tuvieron 1,5 puntos en Teoría y ambos aprobaron. Por esto, en el gráfico correspondiente no se representa ningún porcentaje de desaprobados para ese valor.

Finalmente, es interesante destacar que los docentes son más flexibles al corregir los exámenes finales que el criterio de aprobación fijado por la cátedra (el cual indica que se debe contar con al menos la mitad del puntaje de cada ejercicio para aprobar). Esto indicaría que, al momento de corregir, se considera el examen como un todo, y no cada ejercicio en particular.

Conclusiones

Las evaluaciones de los alumnos pueden ser consideradas como una herramienta importante para medir la eficacia del proceso de enseñanza y aprendizaje de una asignatura. Para el análisis de estos exámenes se suelen utilizar técnicas estadísticas tradicionales, sin embargo éstas suelen resultar ineficientes para detectar los problemas encontrados en el proceso de enseñanza y la detección de relaciones implícitas entre los temas impartidos en clase. Teniendo en cuenta este contexto, el presente trabajo propone la aplicación de procesos de Explotación de Información sobre las evaluaciones de los alumnos para poder extraer patrones de conocimiento que le permita, a los docentes, detectar problemas en la enseñanza y tomar decisiones de mejora en sus asignaturas.

Para demostrar esto, se han presentado los resultados de aplicar diferentes tipos de procesos en dos casos de estudio realizados en el ámbito de la carrera 'Ingeniería en Sistemas de Información' de la Universidad Tecnológica Nacional de Argentina. En el primero, se analizan los exámenes parciales de una asignatura de quinto año mientras que en el segundo se utilizan los exámenes finales de una asignatura troncal de primer año. A partir de la interpretación de las reglas y gráficos generados, es posible determinar comportamientos que serían muy difíciles de identificar a priori mediante técnicas estadísticas tradicionales.

Queda pendiente, como futuras líneas de trabajo, la definición de una guía que permita auxiliar a la interpretación de los resultados de la aplicación del proceso de Explotación de Información, de acuerdo a las características de los exámenes analizados.

Financiamiento

Las investigaciones que se reportan en el presente artículo se financiaron parcialmente por el proyecto “Prácticas y Aplicaciones de Ingeniería de Requisitos en Proyectos de Explotación de Información” (UTI1867) radicado en la Facultad Regional de Buenos Aires de la Universidad Tecnológica Nacional. Además uno de los autores posee el otorgamiento de una beca BINID por parte de la Secretaria de Ciencia y Tecnología de la Nación Argentina, y otro de una beca del Centro de Estudiantes de la Facultad Regional de Buenos Aires.

Referencias

1. UTN FRBA (2008). Perfil Profesional del Ingeniero en Sistemas de Información. <http://tinyurl.com/pqqefwe> Disponible online en septiembre de 2013.
2. UTN (S/A). Programa la carrera Ingeniería en Sistemas de Información (Plan 2008). <http://www.utn.edu.ar/download.aspx?idFile=584> Disponible online septiembre de 2013.
3. Ministerio de Educación Argentino (2009). Resolución 786/2009. Publicado en el Boletín Oficial N° 31.667 (04 de Junio del 2009), Pág. 91-99.
4. Comisión Nacional de Evaluación y Acreditación Universitaria (2012). La CONEAU y el sistema universitario argentino (Memoria 1996-2011). Con colaboración de Gabriela Chidichimo; edición literaria a cargo de Jorge Lafforgue(1a ed. - Buenos Aires). ISBN 978-987-27083-6-8. <http://tinyurl.com/omkvnfh> Disponible online en septiembre de 2013.
5. UNESCO (2009). Declaración de la Conferencia Regional de la Educación Superior en América Latina y el Caribe-CRES-2008. Unipluriversidad, 8(2).
6. Bloom, B.S., Hastings, J.T., Madaus, G. (1971) Handbook of Formative and Summative Evaluation of Student Learning. McGraw-Hill Book Company, Nueva York.
7. Hedges, W.D. (1982). Los exámenes y la Evaluación en la enseñanza de las ciencias. Educ. Med. Salud, 6, Pág430-41.
8. Chang C, Song X, Gao B, Kong F. (2008). Data mining-based engineering project grading technique. 7th World Congress In Intelligent Control and Automation (WCICA 2008), Pág. 3542-3546. IEEE.
9. Jiménez Rey E.M., Rodríguez D, Britos P.V, García Martínez, R. (2009). Caracterización de Problemas de Aprendizaje basada en Explotación de Información. In XI Workshop de Investigadores en Ciencias de la Computación.

10. Kuna H., García-Martínez R, Villatoro F. (2010). Identificación de Causales de Abandono de Estudios Universitarios. Uso de Procesos de Explotación de Información. Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología 5, Pág. 39-44.
11. Saavedra-Martinez P., Pollo-Cattaneo F., Britos P., Rodríguez D., García-Martínez R. (2012). Explotación de Información Aplicada a Identificación de Fallas de Apropiación de Conceptos. Proceedings IX Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento. Pág. 103-110. Sello Editorial de la Pontificia Universidad Católica del Perú.
12. Langseth, J, Vivatrat, N. (2003). Why Proactive Business Intelligence is a Hallmark of the Real-Time Enterprise: Outward Bound. Intelligent Enterprise 5(18), Pág. 34-41.
13. Fayad, U.M., Piatetsky-Shapiro G., Smyth P., Uhturudsamy R. (1996). Advances in KnowledgeDiscovery and Data Mining. AAAI Press.
14. Pollo-Cattaneo M., García-Martínez R., Britos P., Pesado P., Bertone R., Rodríguez D., Merlino H., Pytel P., Vanrell J. (2012). Elementos para una Ingeniería de Explotación de Información. Proyecciones 10(1), Pág. 67-84. ISSN 1667-8400.
15. Garcia-Martínez R., Britos P., Pollo-Cattaneo F., Rodriguez D., Pytel P. (2011). Information Mining Processes Based on Intelligent Systems. Proceedings of II International Congress on Computer Science and Informatics (INFONOR-Chile 2011). Pág. 87-94. ISBN 978-956-7701-03-2.
16. García Martínez R., Servente M., Pasquini D. (2003). Sistemas Inteligentes. Editorial Nueva Librería. ISBN 987-1104-05-7.
17. Pyle D. (2003). Business Modeling and Business Intelligence. Morgan Kaufmann.
18. Deroche A., Pytel P., Pollo-Cattaneo, F. (2013). Propuesta de mejora en asignatura de grado mediante Explotación de Información. Proceedings VIII Congreso de Tecnología en Educación y Educación en Tecnología. Artículo ID 5492. ISBN 978-987-1676-04-0.
19. UTN FRBA (2008). Programa Analítico de la asignatura 'Inteligencia Artificial'. <http://tinyurl.com/qxgqvwl> Disponible online en septiembre de 2013.
20. Kohonen, T. (1995). Self-Organizing Maps. SpringerVerlagPublishers
21. Quinlan J.R. (1993). C4.5: Programs for Machine Learning. Morgan KaufmannPublishers.

22. Deroche A., Vegega C., Pytel P., Pollo-Cattaneo M.F. (2013) Descubrimiento de perfiles de alumnos en exámenes parciales de la asignatura Inteligencia Artificial mediante Explotación de Información. Reporte Técnico GEMIS-TD-2013-01-RT-2013-01. UTN-FRBA. <http://grupogemis.com.ar/?p=605> Disponible online en septiembre de 2013.
23. Rakotomalala R. (2003) Software Tanagra. <http://tinyurl.com/tanagra>.
24. UTN FRBA (2008). Programa Analítico de la asignatura ‘Sistemas y Organizaciones’. <http://tinyurl.com/oh6mrk4> Disponible online en septiembre de 2013.
25. Langley P., Sage S. (1994). Induction of selective Bayesian classifiers. In Proceedings of the Tenth international conference on Uncertainty in artificial intelligence, Págs. 399-406. Morgan Kaufmann Publishers Inc.
26. Luján F., Raus N., Vegega C., Pytel P., Pollo-Cattaneo M.F. (2013) Ponderación de exámenes finales de la asignatura Sistemas y Organizaciones mediante Explotación de Información. Reporte Técnico GEMIS-TD-2013-02-RT-2013-02. UTN-FRBA. <http://grupogemis.com.ar/?p=608> Disponible online en septiembre de 2013.



Nuevas estrategias para realizar evaluaciones en cursos de ingeniería de software: caso Universidad de Medellín

Liliana González Palacio¹ ; Mauricio González Palacio¹; Jaime Echeverri Arias¹

¹ Universidad de Medellín, Medellín, Colombia. ligonzalez, magonzalez, jaecheverri @udem.edu.co

Resumen

La selección de técnicas adecuadas de evaluación incrementa la calidad del proceso educativo. Uno de los parámetros fundamentales es ofrecer un ambiente de confianza al estudiante donde se pueda concentrar en su aprendizaje evitando en lo posible situaciones de estrés. En este trabajo se presentan dos estrategias que incorporan las lúdicas y las redes sociales web 2.0 para realizar evaluaciones en cursos asociados a la línea de ingeniería de software del pregrado en ingeniería de sistemas de la Universidad de Medellín. Los experimentos se aplicaron en los cursos de ingeniería de software I y fundamentos de programación para el semestre 2013-01, en grupos de aproximadamente 20 estudiantes. Los resultados revelan que vincular este tipo de estrategias mejora sustancialmente el rendimiento de los estudiantes durante las evaluaciones, además de facilitar el trabajo colaborativo docente-alumno, donde se asumen roles más activos para propiciar el aprendizaje significativo.

Palabras clave: ingeniería de software, lúdicas, procesos evaluativos, redes sociales web 2.0.

Abstract

Selecting appropriate techniques to assess students increase the quality of educative processes. One of the fundamental parameters is the offering of a trusty environment to students where they can be focused in their learning, avoiding stress situations. In this work, two evaluation strategies are presented that involve social networks web 2.0 and playful activities, which are used to assess subjects related to software engineering for undergraduate students in the University of Medellín. Experiments were performed in “Software Engineering I” and “Programming Fundamentals” in the semester 2013-1 in groups of 20 students. Results show that those strategies improve the overall grades of students while evaluating, as well as easing collaborative work student-teacher, where significant roles are adopted, enhancing effective learning.

Key words: evaluation processes, playful activities, social networks web 2.0, Software Engineering.

Introducción

De acuerdo con [1-4], involucrar activamente a los estudiantes por medio de herramientas no tradicionales de aprendizaje incrementa su capacidad de abstracción, facilita la profundización y desarrolla el pensamiento crítico así como la habilidad de resolver problemas reales. El trabajo colaborativo entre los participantes por medio de juegos y redes sociales mejora el proceso educativo y es una herramienta poderosa para facilitar un aprendizaje significativo.

En este trabajo se presentan dos estrategias para afrontar eventos evaluativos: 1) redes sociales web 2.0 para la generación de ideas en el curso Ingeniería de software I y 2) desarrollo de juegos incorporando conocimientos de conceptos básicos en algoritmia para el curso de fundamentos de programación. Ambas asignaturas hacen parte del pensum de ingeniería de sistemas en la Universidad de Medellín.

El resto del artículo está organizado así: la sección 2 presenta conceptos básicos para el entendimiento de la propuesta; la sección 3 muestra trabajos previos sobre aplicación de lúdicas y herramientas web 2.0 en eventos evaluativos; la sección 4 muestra la propuesta y diseño de las estrategias junto con los resultados obtenidos; en la sección 5 se hace un análisis de los hallazgos. Finalmente se presentan las conclusiones, agradecimientos y referencias.

Marco teórico

A continuación se presentan los conceptos de proceso evaluativo, ingeniería de software y lúdica que serán usados a lo largo de este trabajo.

La evaluación es un elemento básico durante el proceso enseñanza-aprendizaje y parte integral del hecho educativo. Se ocupa de inferir y juzgar a partir de cierta información adquirida de forma sistemática, gradual y continua valorando cambios producidos en el conocimiento del estudiante sobre un tema específico [5].

Los procesos evaluativos en Ingeniería de Software (definida como la aplicación de un enfoque sistemático, ordenado y estructurado para el desarrollo, operación y mantenimiento de sistemas de software [6, 7]) no son triviales. Se hace necesario medir aprendizajes en diversos tópicos que van desde la capacidad de abstracción, hasta posibilidad de asumir cualquiera de las fases propias del desarrollo de software (modelado de negocio, ingeniería de requisitos, análisis, diseño, codificación, pruebas, mantenimiento, implantación), pasando por facilidad de redacción, escritura, síntesis, entre otros. Estas habilidades deberían ser adquiridas por los estudiantes usando estrategias que les permitan participar activamente tanto en el aprendizaje como en la evaluación [8].

Como alternativa a la enseñanza tradicional de la Ingeniería de Software se usan juegos o lúdicas cuyo objetivo es afianzar los conocimientos teóricos [9] además de introducir un componente adicional de motivación y generar mayor recordación de conceptos en el tiempo lo cual se traduce en mejores resultados durante los eventos evaluativos.

La selección de técnicas adecuadas de evaluación incrementa la calidad del proceso educativo. Uno de los parámetros fundamentales es ofrecer un ambiente de confianza al estudiante donde se pueda concentrar en su aprendizaje evitando en lo posible situaciones de estrés [10].

Trabajo relacionado

En este apartado se presentan clasificaciones de procesos evaluativos, y posteriormente se enuncian algunas estrategias no convencionales para realizar evaluaciones en diversas áreas de conocimiento, incluida la ingeniería de software.

En primera instancia, vale la pena enunciar que se reportan diversas investigaciones para identificar los tipos de evaluación existentes a nivel de ingeniería. González y otros proponen la siguiente taxonomía [10]:

- Enfoque tradicional: fundamentalmente cuantitativo y memorístico.
- Enfoque conductivista: se realizan pruebas de tipo repetitivo y cuantitativo (exámenes de selección múltiple, falso/verdadero, complementación) para sistematizar el conocimiento que asimila el estudiante y verificar el logro de objetivos.
- Enfoque contemporáneo: basado en procesos, donde el docente observa y analiza para comprobar, constatar, determinar, identificar, diferenciar, valorar, presentar alternativas y tomar decisiones.

- Enfoque social: dinámica en dos vías donde el profesor recopila información por múltiples vías para elaborar juicios, mientras que el estudiante es consciente de su proceso de formación. Es de tipo cualitativo en modalidad individual o grupal.

En este mismo sentido, Ortiz [5] propone diversos tipos de evaluación: unidimensional, multi-dimensional, diagnóstica, formativa, sumativa, reconstructiva, sistemática, basada en paradigmas socio-culturales, desde paradigmas complejos y comprensivos. Su clasificación involucra diversas dimensiones como: las técnicas usadas, el momento donde se aplican y las responsabilidades asignadas a cada rol involucrado.

En cuanto al uso de lúdicas y estrategias no convencionales para el acto educativo es común encontrar aportes enfocados hacia el proceso de enseñanza [1-4,10, 11], pero pocas propuestas se orientan hacia el mejoramiento de la evaluación. En este universo reducido existen iniciativas donde se incorporan simulacros, actividades previas de motivación y preparación normalmente realizadas en grupo, y otras propuestas más elaboradas tales como preparar evaluaciones de acuerdo al estilo cognitivo de los estudiantes, o aplicar pruebas de figuras enmascaradas (EFT Embedded Figures Test) [12].

Otros autores abordan la validación de conocimientos mediante la experimentación en el área de interés a través de reuniones presenciales y herramientas en línea tales como Web 2.0 [11]. Otra opción planteada es usar la Web 2.0 Peer Evaluation [13] donde se promueve también el uso de web 2.0 para desarrollar proyectos en equipo facilitando que cada compañero evalúe el conocimiento adquirido por los demás miembros del grupo.

El factor diferenciador de esta propuesta radica en la integración de varios conceptos usados de forma individual por los autores. Se aprovechará el uso de lúdicas junto con la incorporación de tecnologías web 2.0 para diseñar procesos evaluativos más efectivos.

Diseño y aplicación de estrategias de evaluación en el contexto de la ingeniería de software

En esta sección se explica la forma en que se estructuró la evaluación final de dos cursos asociados a la línea de ingeniería de software incorporando lúdicas y conceptos que facilitan un proceso colaborativo y multidimensional con enfoque social para la evaluación [10].

Contexto de aplicación

Las propuestas fueron desarrolladas previo a los exámenes finales de los cursos de fundamentos de programación (primer semestre) e Ingeniería de software I (cuarto semestre), ambos pertenecientes al programa Ingeniería de sistemas de la Universidad de Medellín.

En el curso de fundamentos se abordan temáticas como: programación secuencial, condicional y repetitiva, tipos de datos, variables, técnicas de representación de algoritmos, búsqueda y ordenamiento,

arreglos, programación modular, entre otros. Los estudiantes están en un rango de edades entre 17 - 22 años.

De otro lado, la asignatura ingeniería de software I inicia con una introducción a la Ingeniería de Software. Posteriormente se abordan los conceptos de metodología de desarrollo y modelo de ciclo de vida. Luego se trabaja en el modelado del negocio de una organización estudiada como parte de un proyecto de aula. Finalmente está el tema de Ingeniería de Requisitos con sus respectivas fases y técnicas para abordarla, y una práctica sobre la empresa previamente seleccionada.

En los demás apartados de esta sección se explica la forma en que se llevó a cabo cada lúdica.

Estrategia 1: Interacción vía Facebook para la generación de ideas con propósitos educativos y mediante el uso de dispositivos móviles.

Diseño de la estrategia

La iniciativa fue creada con el objetivo de afianzar los conocimientos adquiridos por los estudiantes en cuanto a modelos y metodologías de desarrollo software, además de reconocer los contextos donde es apropiado usar unos y otros modelos. También se buscó preparar a los estudiantes para el examen final. La estrategia se construyó para el curso de ingeniería de software I combinando lúdicas y redes sociales web 2.0 como herramientas de interacción social que facilitan un intercambio dinámico entre miembros de cada equipo con características y necesidades similares [14].

Consistió en la conformación de equipos de 3 o 4 integrantes que en un margen de 2 días y usando como medio de interacción la reconocida red social Facebook, generaban ideas, comentaban los aportes de los demás, votaban buscando constituir un banco de preguntas y respuestas para el examen final del curso. Las mejores ideas construidas de forma colaborativa tuvieron bonificación y fueron incluidas en el componente teórico del examen final, conformado por un total de 30 preguntas, donde el 50% fue extraído de aportes de los estudiantes.

El ejercicio se hizo usando dispositivos móviles para facilitar a los participantes el ingreso de aportes en cualquier momento y lugar, por lo tanto, antes de comenzar se les solicitó diligenciar un cuestionario verificando aspectos como la disponibilidad de dispositivos móviles y la conexión de la red, y la actividad fue desarrollada con aquellos que cumplían los parámetros. A continuación se explican los detalles de la estrategia:

- Temática para la generación de ideas: Preguntas y respuestas sobre modelos de ciclo de vida y metodologías de desarrollo de software.
- Materiales: Un dispositivo móvil (Tablet o smartphone), conexión continua a internet, cuenta en Facebook.

- Paso 1- Conformar grupos: Deben ser de 3 o 4 personas máximo. Miembros de un mismo grupo van a intercambiar opiniones y la mejor bonificación será para el grupo con mayores aportes de más alta calidad. Una vez constituido el grupo deben reportarlo al docente, además de crear un grupo en Facebook y adicionar al profesor como administrador
- Paso2- Leer las instrucciones con mucho cuidado y preguntar las inquietudes: se dispone de canales como Skype y correo electrónico, además de interacción cara a cara para resolver dudas.
- Paso 3- Entrar a Facebook y generar ideas: cada estudiante debe insertar Posts en el grupo de Facebook asociados a ideas de preguntas sin olvidar mantener un formato donde se indiquen datos como el dispositivo usado; actividad ejecutada cuando se ocurrió la idea (Estudiando, trabajando, labores del hogar, con mi novia, ocio, otra (Indicar cuál?)); Lugar/ubicación al momento de pensar en la idea (En mi casa, en la Universidad, en el trabajo, en la calle, en otra parte (Indicar cuál?)); Pregunta con un encabezado acompañado de las posibles respuestas indicando entre paréntesis cuál es correcta. Se presentaron ejemplos de preguntas buenas y regulares. En la figura 1 hay una muestra:

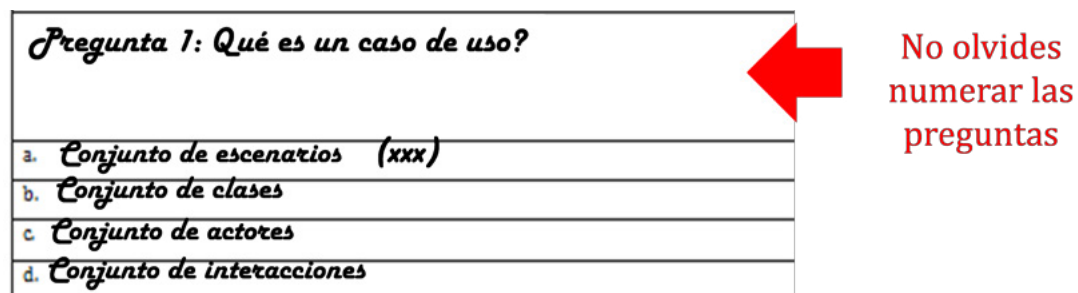



Figura 1. Ejemplo de pregunta/ respuesta estrategia 1

- Paso 4- Votar por las ideas y comentarlas: De forma simultánea cada participante puede incorporar votaciones y comentarios de las ideas generadas por otros miembros del grupo, con lo cual se genera colaboración para perfeccionar y complementar los aportes. Es ideal también tener registro de la actividad, sitio, y situación en la que se encontraba el participante cuando agregó el comentario.
- Paso 5- Consolidar resultados: una vez pasan dos días, los miembros del grupo deben hacer un conteo de las ideas generadas durante la interacción, y sus respectivos comentarios, además de hacer un informe completo, usando para ello un formato como el que se muestra en la figura 2. De otro lado también se genera un informe detallado de la interacción incorporando todos los post y comentarios generados por cada participante.


- Paso 6- Seleccionar las mejores preguntas: esta responsabilidad está en manos del docente que de acuerdo a su criterio de experto y al análisis de los resultados reportados por los equipos decide cuáles serán los aportes a incorporar en el examen final. En este punto también el profesor hace una retroalimentación a los estudiantes sobre los logros obtenidos y los puntos de mejora. De esta manera finaliza el ejercicio.



FORMATO PARA CONSOLIDAR RESULTADOS DEL EQUIPO

EVALUACIÓN DE HERRAMIENTAS PARA LA GENERACIÓN DE IDEAS:
FACEBOOK VS. ESQUEMAS FACE TO FACE O PRESENCIALES

Nombres miembros equipo *Fulanito, Peranito, Zutanita*



En este formato el equipo debe ordenar las preguntas que todos hicieron de acuerdo a las calificaciones obtenidas, ubicando primero las que obtuvieron más "Me gusta" hasta llegar a las que obtuvieron menos "me gusta". Si hay empates, el equipo resuelve cómo ubicar las preguntas de acuerdo al número de comentarios obtenidos y a un consenso. Entre más comentarios tenga mucho mejor es.

Orden de las preguntas En esta columna ubique el número de la pregunta	Quién hizo la pregunta?	Nro.votos obtenidos	Nro. comentarios
<i>Pregunta 2</i>	<i>Zutanito</i>	<i>4</i>	<i>10</i>
<i>Pregunta 5</i>	<i>Peranito</i>	<i>4</i>	<i>7</i>
<i>Pregunta 8</i>	<i>Zutanito</i>	<i>3</i>	<i>3</i>
<i>Pregunta 4</i>	<i>Fulanito</i>	<i>2</i>	

Orden de las preguntas En esta columna ubique el número de la pregunta	Quién hizo la pregunta?	Nro.votos obtenidos	Nro. comentarios
<i>Pregunta 8</i>	<i>Peranito</i>	<i>1</i>	

La pregunta 2 de Zutanito quedo empatada con la de Peranito pero tuvo más comentarios entonces gana

La pregunta 8 de Peranito fue la menos votada




Figura 2. Ejemplo formato para la consolidación de resultados.

- Anotaciones importantes: los participantes no tienen que estar todo el tiempo conectados a la red social, pero el hecho de hacerlo vía dispositivo móvil proporciona un ambiente de fácil acceso para integrar un aporte en cualquier momento y lugar.
- Variables y parámetros medidos durante la interacción: el número de ideas generadas por lapso de tiempo definido, calidad de los aportes (aportes coherentes vs “basura”), percepción de los estudiantes sobre estas estrategias no convencionales.

- Formatos fabricados: Diagnóstico inicial de accesibilidad a dispositivos móviles y redes; presentación de la actividad y guía de ejecución; generación de ideas sobre Preguntas/ respuestas; calificación de aportes; consolidación de resultados por equipo.

Reporte de resultados

La propuesta fue aplicada a 18 estudiantes del curso de ingeniería de software I durante el semestre 2013-01 que conformaron 5 equipos de aproximadamente 3 o 4 integrantes cada uno. En cuanto a las métricas asociadas al experimento se midió: cantidad de ideas por equipo (M1), cantidad de comentarios (M2) y número promedio de aportes individuales (M3). Los resultados se presentan en la tabla 1:

Tabla 1. Resultados de la estrategia 1.

	Equipo 1	Equipo 2	Equipo 3	Equipo 4	Equipo 5
M1	35	25	19	12	17
M2	4	3	7	3	2
M3	11,6	8,33	6,33	6	5,66

Una vez terminada la actividad, se aplicó un instrumento para capturar la percepción de los estudiantes con respecto a la inclusión de nuevas estrategias de evaluación:



Figura 3. Percepción de los estudiantes al aplicar la estrategia 1.

Por último se hizo un comparativo de los resultados obtenidos durante los exámenes finales de un grupo al cual no se le aplicó ninguna estrategia de preparación (Grupo 1) vs el curso que estudió usando esta lúdica (Grupo 2). Los resultados revelan que el promedio de nota del examen final para el grupo 1 fue de 3.16 (varianza=0.5) y en el grupo 2 fue de 3.83 (varianza=0.7).

Estrategia 2: Diseño de juegos para afianzar los conceptos de fundamentos de programación

Diseño de la estrategia

Esta propuesta se diseñó con el objetivo de medir en el estudiante su capacidad de abstraer técnicas y conceptos en un problema de aplicación real, documentar fielmente la solución propuesta, y argumentar el desarrollo acometido. Se buscó superar las técnicas tradicionales memorísticas (revisión de teoría y definiciones) e iterativas (diseño de un algoritmo y prueba de concepto), liberando adicionalmente al estudiante de la presión generada por exámenes.

El mecanismo de evaluación se basó en el planteamiento de un problema integral a resolver mediante los conceptos aprendidos en la asignatura (matrices, métodos de búsqueda/ordenamiento, programación cíclica, condicionales, operaciones de entrada y salida, temporización, números aleatorios) y las habilidades algorítmicas desarrolladas por los estudiantes. Para tal efecto, cada grupo de 2 estudiantes hizo la programación de un juego común, el cual se conoce en la idiosincrasia colombiana como “Concéntrese”.

En primera instancia, se retó al estudiante a crear dinámicamente arreglos bidimensionales, los cuales contendrían parejas de números distribuidos aleatoriamente dado que el número de filas y columnas debería ser un parámetro de entrada para el juego. Por ejemplo, si la matriz fuese de 4 filas por 4 columnas, el estudiante debería inferir que el número de parejas sería 8, y distribuir aleatoriamente pares de números desde el 1 hasta el 8.

Una vez el arreglo estuviese correctamente generado (lo cual se evidencia al tener los pares de números distribuidos aleatoriamente en cada ejecución del programa en dicho arreglo), el estudiante debía diseñar un algoritmo para mostrar al usuario la matriz por un tiempo específico para luego pedir las coordenadas de los números que coincidieran en la cuadrícula. Por ejemplo, si el par de números “uno” estaban en las posiciones (1,1) y (2,2), el usuario tendría que ingresar ambos pares ordenados. Se permitía errar las coordenadas por tres veces consecutivas para dar el juego por terminado. Para ganar el juego se debía encontrar todos los pares de números. El programa debía estar orientado a funciones y procedimientos además de ser documentado de forma exhaustiva sin perder de vista la modularidad del código, facilitando con esto el desarrollo de capacidades para distribuir responsabilidades y trabajar en equipo.

Para ejecutar la estrategia se requirieron entre otros: escritura de pseudocódigo de la solución a implementar, diagrama de entradas y salidas del juego, ambiente de programación MATLAB 2012, dada la usabilidad que este software presenta para estudiantes recién iniciados en algoritmia.

Mediante esta dinámica fue posible evaluar el 100% del currículo sugerido por la Universidad de Medellín en la asignatura Fundamentos de Programación.

Para medir los logros de la actividad se aplicaron tres métricas. Las primeras dos relacionadas con la revisión exhaustiva del pseudocódigo y el programa final, así como su documentación. La tercera mediante sustentación oral.

Reporte de resultados

La estrategia se aplicó a un grupo de 29 estudiantes del curso de fundamentos. Al finalizar se pidió a los alumnos diligenciar una encuesta de percepción. El instrumento se estructuró como se indica a continuación, y generó los siguientes resultados:

1. ¿Consideras que aprender mediante la utilización de un juego optimiza el tiempo para el entendimiento y asimilación de conceptos? Sí 100%, No 0%, No sé 0%.
2. Consideras que los conceptos aprendidos mediante la programación de un juego serán relevantes en algún área de tu disciplina ingenieril futura? Sí 79%, No 7%, No sé 14%.
3. Si respondiste SI a la pregunta anterior, indica por favor en qué áreas o materias consideras que aportará el haber desarrollado un juego? Lenguajes de programación 21%, Ingeniería de software 21%, Estructuras de datos 25%, Bases de datos 15%, otros 17%.
4. ¿Consideras que al programar el juego aumentaron tus habilidades técnicas? Sí 90%, No 7%, No sé 3%.
5. Dado que la programación del juego se realizó en equipos, ¿la comunicación con tus compañeros de equipo fue eficiente y/o fluida? Sí 83%, No 7%, No sé 10%.
6. ¿El hecho de programar un juego te motivó para resolver el problema complejo que se presentó? Sí 76%, No 10%, No sé 14%.
7. En tu concepto, la programación del juego afianzó cuáles de los siguientes conocimientos del curso: Programación secuencial 13%, Condicionales 17%, Ciclos 19%, Vectores 17%, Matrices 23%, Búsqueda y ordenamiento 10%, otro 2%.
8. ¿Consideras que la solución planteada para el juego fue creativa? Sí 93%, No 3%, No sé 3%.
9. Una vez finalizó la programación del juego, sentiste un grado de satisfacción: Bajo 3%, Medio 38%, Alto 59%.

Es evidente que para la mayoría de estudiantes la evaluación por medio de la programación de un juego afianza conocimientos en el área de algoritmia y estructuras de datos básicas, haciendo que haya una motivación adicional en el proceso de aprendizaje y que la presión asociada a la evaluación se disminuya.

El segundo criterio para verificar la efectividad de las lúdicas en los procesos de evaluación se basó en la comparación de las notas finales de dos grupos de estudiantes de la asignatura. El primer grupo se enfrentó a evaluaciones escritas con un enfoque tradicional, y el segundo grupo se evaluó por medio de la programación del juego.

El grupo 1 tuvo una media de 3.31 sobre 5 equivalente al 66.2%, con una varianza muestral de 0.625. El grupo 2 tuvo una media de 3.75 sobre 5 equivalente al 75%, con una varianza de 0.61. En vista que los resultados individuales de cada estudiante se encuentran relativamente concentrados en torno a la media para ambos grupos, es posible inferir que hubo una mejora cuando se empleó la evaluación por medio de lúdicas correspondiente a un 9% aproximadamente. Cabe aclarar que este es un piloto y para validar completamente los hallazgos sería necesario aplicar el mismo experimento a una muestra más representativa de la población de estudiantes de fundamentos de programación en la Universidad de Medellín.

Análisis de resultados

De acuerdo a los hallazgos es posible evidenciar una mejoría significativa en los resultados de los exámenes finales cuando se aplicaron lúdicas para preparar a los estudiantes.

En este resultado juega un papel muy importante la motivación y familiaridad con la tecnología empleada en las estrategias planteadas. En el caso de la programación de juegos, los estudiantes de primeros semestres demuestran interés al evidenciar casos prácticos donde es útil la teoría que imparte el docente. Para el escenario planteado en la estrategia 1 que usa las redes sociales como elemento esencial durante el intercambio de información académica resulta divertido usar Facebook en actividades diferentes al ocio.

Es interesante resaltar que actualmente los jóvenes están altamente influenciados y permeados por la tecnología, con lo cual no fue difícil encontrar un escenario apropiado y natural para aplicar las estrategias planteadas.

Los estudiantes actuales están inmersos en un entorno cargado de información y medios interactivos. Por esta razón resulta fundamental complementar las estrategias tradicionales de enseñanza con otras alternativas que le permitan al estudiante ser “actor principal” de su proceso y no sólo un espectador.

Conclusiones y trabajo futuro

Generar espacios de reflexión, debate y crítica convierte la evaluación en un elemento de retroalimentación en lugar de ser vista como un castigo, promoviendo una mejor actitud del estudiante hacia su proceso de formación.

El aprendizaje significativo de los estudiantes en el área de Ingeniería de Software es un verdadero reto para los docentes, de ahí la importancia de facilitar un alto nivel de interacción mediante estrategias novedosas donde además se desarrollan habilidades como la creatividad, toma de decisiones, capacidad de negociación, el liderazgo, buena comunicación y trabajo en equipo.

Como trabajo futuro se propone darle continuidad a esta iniciativa en los cursos siguientes de la línea de Ingeniería de Software, no buscando reemplazar las estrategias tradicionales de evaluación, sino complementarlas de tal manera que se pueda incrementar la motivación de los estudiantes y combinar lúdicas con aprendizaje significativo y desarrollo de habilidades sociales en su proceso de formación profesional. Además es necesario proponer otras escalas de evaluación, probablemente basadas en rúbricas, que permitan una evaluación objetiva sobre ítems de difícil medición en algunos proyectos de Ingeniería de Software.

Es claro que esta prueba es de concepto y para validar completamente los hallazgos sería necesario aplicar el mismo experimento a una muestra más representativa de la población de estudiantes de fundamentos de programación en la Universidad de Medellín.

También está pendiente revisar cuidadosamente el elemento pedagógico de estas prácticas evaluativas y hacer un proceso estadístico más riguroso con los datos extraídos durante la aplicación de cada lúdica. Se hace imprescindible además enlazar esta propuesta con otras de la universidad para disminuir niveles de deserción y otros indicadores que preocupan a las Instituciones de Educación Superior de Colombia.

Agradecimiento

A la Universidad de Medellín por facilitar la aplicación de estrategias de enseñanza basadas en juegos en los cursos de pregrado. A CICESE- México por orientar el diseño de la primera estrategia presentada a manera de experimento.

Referencias

1. Sweedyk E. et al. Computer games and CS education: why and how, in Proceedings of the 36th SIGCSE technical symposium on Computer science education. 2005, ACM: St. Louis, Missouri, USA. p. 256-257.

2. Callahan D, Pedigo B. Educating experienced IT professionals by addressing industry's needs. *Software, IEEE*, 2002. 19(5): p. 57-62.
3. Navarro E., Baker A., Van der Hoek A. *Teaching Software Engineering Using Simulation Games*. 2004.
4. Duarte M., Zapata C. El juego de la consistencia: Una estrategia didáctica para la ingeniería de software. *Revista Técnica Ingeniería Universidad de Zulia CLEI Electronic Journal*, 2008.
5. Ortiz, L., *Técnicas de evaluación*, ed. E. Eco. Vol. 1. 1987, Bogotá: Ediciones Eco.
6. IEEE, *IEEE Standard Glossary of Software Engineering Terminology*. IEEE Std 610.12-1990, 1990: p. 1-84.
7. ACM, *Computing Curricula 2005: The Overview Report*. 2005.
8. Morell, T. *¿Cómo podemos fomentar la participación en nuestras clases universitarias?* 2009. Universidad de Alicante. Departamento de Filología Inglesa.
9. Zapata C., Giraldo G. El juego del diálogo de educación de requisitos. *Avances en Sistemas e Informática*, 2009. 6(1): p. 10.
10. González E.M., D. M.I, Un modelo de evaluación curricular alternativo y pertinente con el diseño curricular basado en la solución de problemas para la formación de profesionales: Caso programa de Contaduría pública de la Universidad de Antioquia. *Contaduría Universidad de Antioquia*, 2007. 51: p. 105,130.
11. Staron, M. Using Experiments in Software Engineering as an Auxiliary Tool for Teaching--A Qualitative Evaluation from the Perspective of Students' Learning Process, in *Proceedings of the 29th international conference on Software Engineering*. 2007, IEEE Computer Society. p. 673-676.
12. Lizcano A. et al., *Comunidades de aprendizaje mediadas por redes informáticas*. *Educación Y Educadores*, 2008. 11(1): p. 199-224.
13. Watkins K.Z. Peer evaluation as a needed web 2.0 activity in project management for teaching practical software engineering, in *Proceedings of the 10th ACM conference on SIG-information technology education*. 2009, ACM: Fairfax, Virginia, USA. p. 173-177.
14. Caldevilla D. *Las Redes Sociales: Tipología, uso y consumo de las redes 2.0 en la sociedad digital actual*. *Documentación de las Ciencias de la Información*, 2010. 33: p. 45



