Avances en las Tecnologías de la Información y Comunicaciones

EDITORES:

- Dra. Alma Rosa García Gaona.
- Dr. Francisco Javier Álvarez Rodríguez.
- M. en C. Ma. de Lourdes Sánchez Guerrero.

EDITORIAL:

ALFA-OMEGA.

ISBN:

978-607-622-367-3

LUGAR:

MEXICO, D.F.

IX - Ingeniería de Software

Modeling technique for design software architectures with component-connector view (SOFTWARE ENGINEERING)

María Eugenia Cabello[1], Francisco Preciado[2], Sara Sandoval[3]
[1,2,3]University of Colima, Ave. Universidad 333, 28040 Colima, México
[1]ecabello@ucol.mx, [2]frpreciado@ucol.mx, [3]sary@ucol.mx
[1,2,3] Tel. 52 312 3161075

Abstract: This paper deals with a modeling technique for design architectural structures with component-connector view of software systems. We take into account the user functional requirements (at high abstraction level) modeled as a UML use case diagram. This modeling technique shows that a software system, as a component-connector architecture model, varies both in the configuration and in the architectural elements. This technique provides the design decisions of the authors of this paper. In order to illustrate our modeling technique, we present the step-by-step modeling of a bank account system using the automated teller machine (ATM) case study, and an expert system using the medical diagnosis case study.

Keywords: architectural structures, modeling technique, component-connector architecture.

1 Introduction

The development of software systems applications has acquired great importance in recent years and there is therefore a need to properly support them. But these systems are complex because their elements varient. In order to minimize this problem, we propose to construct the software system by means of a modeling technique for architectural structures of component-connector view, based on the Model-Driven Architecture (MDA) [4] approach.

To develop our technique, a field study has been done to learn about the structure of the rule-based expert systems that carry out tasks in a specific domain, and the bank account systems.

The structure of the paper is as follows: Section 2 presents the different technological spaces that our work integrates, and mentions some related works. Section 3 presents our modeling technique, which includes two examples. Section 4 presents our conclusions and proposals for future work.

2 Foundations

Our work integrates the following technological spaces in order to cope with the complexity of the problem.

a) Model-Driven Architecture (MDA) [4] is an initiative promoted by the Object Management Group (OMG) [5] for software system development. MDA is based on the separation of the description of the functionality of the system from its implementation on specific software platforms. MDA proposes defining and using models (as first class citizens) at different abstraction levels. From these models we can automatically generate code by means of transformation rules to obtain executable models.

b) Component-Based Software Development (CBSD) [2] approach that develop architectural models. It has, as first citizen, two types of architectural elements: components and connectors.

c) The Unified Modeling Language (UML) [8] defines the industry-standard notation and semantics for object-oriented and component-based systems. This language is an initiative promoted by the Object Management Group (OMG). It has undergone several revisions, the latest of which is UML 2.0. UML uses several diagrams in order to develop software systems, like Use Case diagram, Classes diagram, Sequence diagram, States diagram, and so on.

d) Expert Systems [1] capture the knowledge of experts and attempts to imitate their reasoning processes when the experts solve problems in a specific domain. Such systems usually have a generic architecture and its components are independent and separate units. Control is independent from the data. The input and output of the information is carried out through the user interface.

e) The Bank Account System allows the user to know your balance, withdraw money and make transfers, among other functions.

Several works had been done in order to make up software architectures with component-connector view. A closer research to us is presented in [7] where it is proposed a variability modeling method, which is specifically devised for the component and connector view of software architectures by using UML as the architecture modeling language. Our work proposes a modeling technique for design architectural models of view component-connector, taking into account the user functional requirements modeled as a UML use case diagram.

3 Our modeling technique

In this section we present the modeling technique that we have developed to configure software systems using component-connector (C-C) models. The architectural elements of the base architecture are obtained by identifying functional scenes of the system and assigning each element a functional scene. This modeling technique shows that a software system as a component-connector architecture model varies both in the configuration and in the architectural elements.

Our modeling technique follows three steps:

1) *Specify the user functional requirements (at high abstraction level) modeled as a UML use case diagram.* The number of use cases, the number of actors, and the number of use cases that an actor accesses are the relevant variants.

2) *Create an architectural model for each use case*. We specify the objects (class) that are involved in the process during the execution of the system. Each object is represented by means of a component (architectural element). For each use case, it is created a connector (architectural element) that links all components of the use case. It establishes a one-to-one relationship, between use cases and connectors.

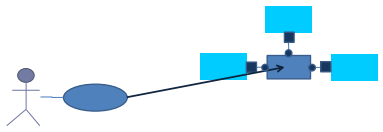


Fig. 1. Architectural model corresponding to use case

Given the existence of an «include» relationship between two use cases, it is fused connectors (see Figure 2). We consider this type of use case, as a subprocess of the process corresponding to the use case that includes.

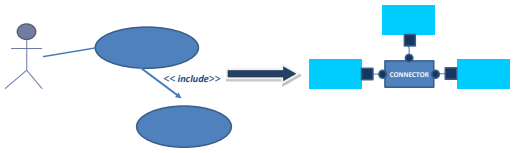


Fig. 2. Architectural model corresponding to "include" relation between two use cases

3) *Build the final architectural model*. Since the function of a component is unique throughout the system, provides that for each use case there will be a different view of each component. When it is created the final architecture, for each use case, the fusion of components (i.e., the union of the views) is done by adding the component ports that connect it with each connector through different roles.

The architectural elements of the final model are obtained using the following criteria:

• **connectors**: the number of connectors of the final architectural model is equal to the number of use cases

• **components**: we have one component per object or class that participates in the system, and to join each component's port we use attachments. The number of ports of a component will be equal to the number of (use cases) connectors where the component is used.

The architectural elements of the final model therefore have the following characteristics:

- there is one connector connecting all the architectural components for each use case;
- the number of ports of a component is the same as the number of use cases;
- the number of User Interface components is the number of actors of the use cases;
- the number of ports of the User Interface component is the number of use cases that can be accessed by an actor.

3.1 Case studies

In the following, we present the step-by-step modeling of a bank account system using as case study the automated teller machine (ATM), and an expert system using as case study the medical diagnosis.

3.1.1 The architecture of the ATM

The generic architecture modular model of an ATM is represented by four fundamental modules: i) User or Client Module, allowing user interaction with the system. ii) ATM Module, transforming data into useful information for decision making and implementing the inference process, iii) Source Account Module, containing the knowledge of the source account, iv) Target Account Module, containing the knowledge of the source account.

In the following, we present the step-by-step modeling of a bank account system using as case study the ATM.

1). *Specify the user functional requirements (at high abstraction level) modeled as a UML use case diagram*. In our example (see Figure 3) we have three use cases (check balance, withdrawal, transfer) and one actor (user or client), where the actor accesses three use cases.

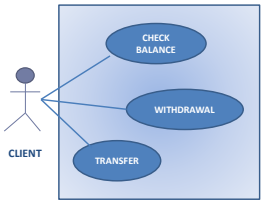


Fig. 3. Use cases diagram of ATM example

2) *Create an architectural model for each use case.* In our example, we have four components (User or Client, ATM, Source Account, and Target Account), and three use cases, therefore we build three architectural models with four components (see Figure 4).

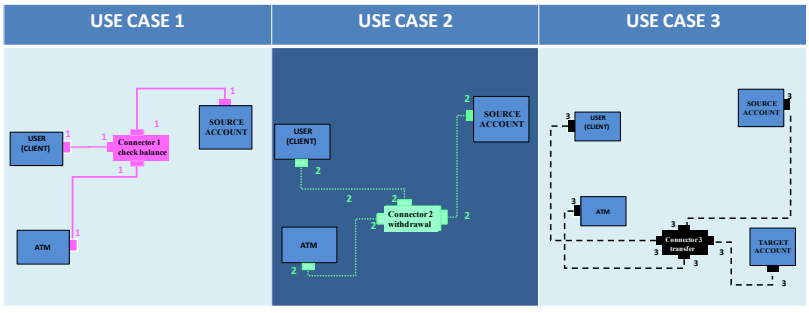


Fig. 4. Architectural models of ATM example

3*) Build the final architectural model*.-In our example, the final architectural model of the bank system is presented in Figure 5. It has seven architectural elements with the following characteristics:

- the User component (i.e., the client) has three ports because this component runs three use cases (check balance, withdrawal, transfer),
- the ATM component has three ports because this component runs three use cases (check balance, withdrawal, transfer),
- the Source Account component has three ports because this component runs three use cases (check balance, withdrawal, transfer),
- the Target Account component has one port because this connector coordinates one use case (transfer),
- the Check Balance connector has three ports because this connector coordinates three components (user, ATM, source account) through one use case (check balance),
- the Withdrawal connector has three ports because this connector coordinates three components (user, ATM, source account).through one use case (withdrawal),
- the Transfer connector has four ports because this connector coordinates four components (user, ATM, source account, target account).through one use case (transfer).

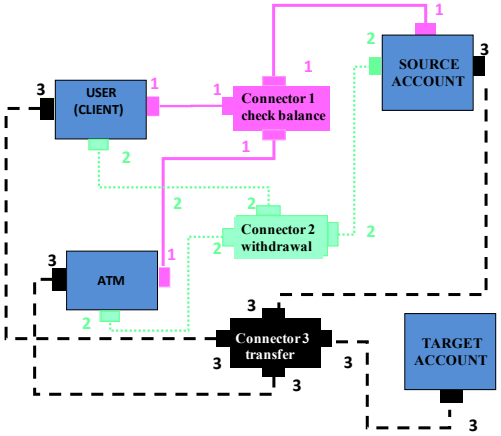


Fig. 5. Final architectural model of ATM example

3.1.2 The architecture of the Expert Systems

The generic architecture modular model of an Expert System is represented by three fundamental modules: i) Inference Process Module, transforming data into useful information for decision making and implementing the inference process. ii) Knowledge Base Module, containing the knowledge of the application domain. iii) User Interface Module, allowing user interaction with the system.

In our example, we have considered an expert system that performs a medical diagnosis, i.e., a medical diagnostic expert system, which has two users (doctor and laboratory member), and therefore two user interface components (Clinical User and Laboratory User respectively).

In the following, we present the step-by-step modeling of an expert system using as case study the medical diagnosis.

1) *Specify the functional requirements (at high abstraction level) by using UML use case diagrams*. In our example (see Figure 6) we have three use cases (get clinical diagnosis, get laboratory diagnosis, get diagnosis), and two actors (doctor, laboratory

member), where an actor (doctor) accesses two use cases (get clinical diagnosis, get diagnosis) and the other one (laboratory member) access a single use case (get laboratory diagnosis).

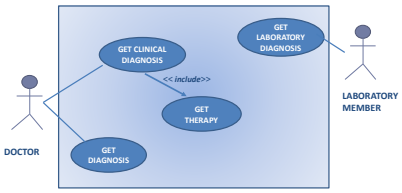


Fig. 6. The use case diagram of Expert System example

2) *Create an architectural model for each use case*. In our example, we have four components (Clinical User, Laboratory User, Knowledge Base, and Inference Motor), and three use cases, therefore we build three architectural models (see Figure 7).

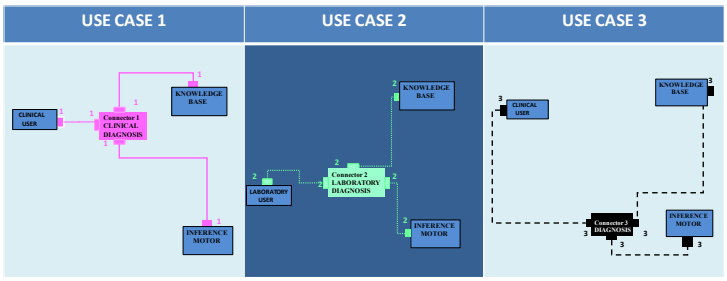


Fig. 7. Architectural models of Expert System example

In this example, given the existence of an «include» relationship between two use cases (get diagnosis, get therapy), it is fused connectors (see Figure 8).

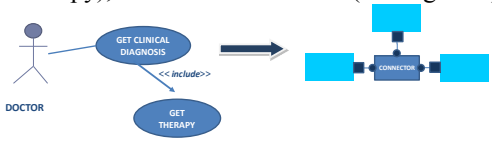


Fig. 8. Architectural model corresponding to "include" relation between two use cases in the Expert System example

3) *Build the final architectural model*.-In our example, the final architectural model of the medical diagnostic expert system is presented in Figure 9. It has seven architectural elements with the following characteristics:

- the Clinical User component (i.e., the doctor) has two ports because this component runs two use cases (get clinical diagnosis, get diagnosis),
- the Laboratory User component (i.e., the laboratory member) has one port because this component runs one use case (get laboratory diagnosis),
- the Knowledge Base component has three ports because this component runs three use cases (get clinical diagnosis, get laboratory diagnosis, get diagnosis),
- the Inference Motor component has three ports because this component runs three use cases (get clinical diagnosis, get laboratory diagnosis, get diagnosis),

- the Clinical Diagnosis connector has three ports because this connector coordinates three components (Clinical User, Knowledge Base, Inference Motor) through one use case (get Clinical Diagnosis),
- the Laboratory Diagnosis connector has three ports because this connector coordinates three components (Laboratory User, Knowledge Base, Inference Motor) through one use case (get Laboratory Diagnosis),
- the Diagnosis connector has three ports because this connector coordinates three components (Clinical User, Knowledge Base, Inference Motor) through one use case (get Diagnosis).

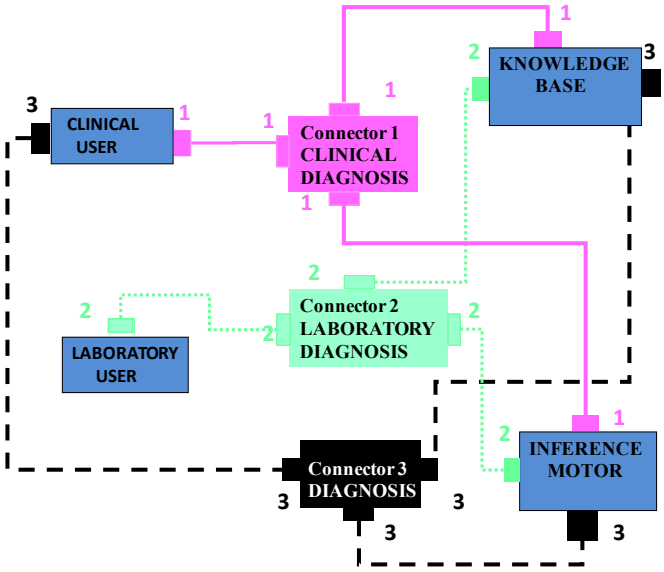


Fig. 9. Medical diagnostic Expert System architectural model

4 Conclusions and Future Work

This modeling technique shows that a software system as a component-connector architecture model varies both in the configuration and in the architectural elements. The variability in structure among the architectural elements of a software system includes: number of use cases, number of actors, and uses cases by actor, i.e., this is the end user requirements variability.

Our main contribution to research is to obtain skeletons of software architectures with component-connector view from a UML-Use Case Diagram of the system. Our technique is simple and based on design patterns [6].

This technique provides the design decisions of the authors of this paper. We use this technique in order to control and minimize the time to market and high costs of the software development process.

In the future, we want to apply our technique on several case studies and different domains. Finally, we intend to implement this modeling technique by using model transformations [3], in order to automate our proposal.

Acknowledgments. This work has been funded under the PIFI project (México government).

References

[1] Giarratano, J. and Riley, G.: *Expert Systems: Principles and Programming*. 4th Edition, Hardcover, ISBN: 0534384471, 856 pages, 2004.

[2] Kung-Kiu L.: *Component-Based Software Development: case studies*. Series on Component-Based Software Development, Vol. 1., World Scientific Publishing Company, ISBN: 978-981-238-828-5 (Hardcover), 312 pages, 2004.

[3] Mens, T., Czarnecki, K., and Van Gorp, P.:A Taxonomy of Model Transformations. Electronic Notes in Theoretical Computer Science 152 (2006) pp. 125–142. Available in: http://staffwww.dcs.shef.ac.uk/people/A.Simons/remodel/papers/MensVanGorpTaxonomy.pdf

[4] Model Driven Architecture: MDA. http://www.omg.org/mda

[5] Object Management Group: OMG: http://www.omg.org

[6] Pressman R.: Ingeniería del software: un enfoque práctico. McGraw-Hill, 7ª. Edición, ISBN: 978-607-15-0314-5, 2005.

[7] Razavian M. and Khosravi, R.: *Modeling Variability in the Component and Connector View of Architecture Using UML*. AICCSA '08 Proceedings of the 2008 IEEE/ACS International Conference on Computer Systems and Applications, IEEE Computer Society, Washington, DC, USA, pp. 801-809, 2008, doi:10.1109/AICCSA.2008.4493618

[8] Unified Modeling Language: UML. http://www.uml.org

Sistemas de Apoyo a la Toma de Decisiones basados en Modelos de Características de Líneas de Productos

(Ingeniería de Software)

María Eugenia Cabello[1], Francisco Preciado[2] , Salvador Macías[3]
[1,2,3]University of Colima, Ave. Universidad 333, 28040 Colima, México
[1]ecabello@ucol.mx, [2]frpreciado@ucol.mx, [3]mes@ucol.mx
[1,2,3] tel. 523123161075

Resumen: Se presenta la generación automática de un sistema de apoyo a la toma de decisiones que ayude al usuario a elegir un producto específico en una línea de productos, seleccionando las características de su preferencia. La variabilidad de dicha línea de productos, como información relevante para la toma de decisiones, es plasmada en un modelo de características, el cual es embebido en el propio sistema. Así, con una aplicación desarrollada "ad hoc", se podrá generar automáticamente una línea de productos software de sistemas de apoyo a la toma de decisiones, obteniendo un sistema diferente para cada línea de productos (i.e. diferentes dominios) al cambiar el modelo de características. Esto implica la reutilización de paquetes de software, con lo que se contribuye a mejorar costos de producción y tiempos de mercado para el desarrollo de software. Al usar el sistema de apoyo a la toma de decisiones se podrá reducir el tiempo invertido en la selección de un artículo/producto que satisfaga necesidades y preferencias del usuario (ejemplo: clientes/vendedores de una empresa), pero con la seguridad de contar con la existencia de un sólo artículo/producto que forma parte de la línea de productos. Para ilustrar esta propuesta se ha seleccionado el caso de los celulares.

Palabras clave: Sistemas de Apoyo a la Toma de Decisiones, Línea de Productos Software, Modelo de Características.

1 Introducción

En el mercado actual la elaboración de productos que comparten características similares ha llevado a considerar a éstos como una línea de productos, reduciendo tiempos y costos.

Las grandes compañías fabricantes de celulares lanzan al mercado cientos de productos de diversas características. Esto hace que la selección de un modelo, que satisfaga las necesidades y gustos del cliente se torne laborioso, extenuante e infructuoso en ocasiones.

Para evitar complicaciones, los vendedores de productos pueden apoyarse en software para seleccionar el producto que el cliente desea adquirir. El uso de los Sistemas de Apoyo a la Toma de Decisiones[17] [16], permiten solucionar este problema. Dado que estos sistemas interpretan los datos recibidos por el usuario/cliente[18] es posible generar una hipótesis sobre el producto más cercano a sus preferencias. Las ventajas son:

[17] Del inglés Decision Support Systems
[18] Características particulares de su elección.

permanencia, replicación, rapidez y fiabilidad de la información que se maneja y de la estrategia para solucionar un problema o realizar una toma de decisión.

Partiendo de lo anterior, en este artículo se propone el desarrollo automático de un sistema de apoyo para la toma de decisiones que seleccione un producto de una línea de productos (celulares en este caso), proporcionando las siguientes ventajas:

- Mostrar un producto afín al gusto del cliente.
- Reducir tiempos de búsqueda de los usuarios al buscar un producto.
- Asegurar que las características particulares del producto que el usuario desea integren un producto específico y único de la línea de productos.
- Facilitar la renovación de la línea de productos.

La estructura de este artículo es la siguiente: en el capítulo 2 se presentan brevemente los fundamentos teóricos de este trabajo. En el capítulo 3 se expone la metodología utilizada en el desarrollo de nuestro sistema de apoyo a la toma de decisiones. En el capítulo 4 se presentan un esbozo de la implementación de nuestra propuesta. En el capítulo 5 se comentan los aportes de otros autores relacionados con nuestra investigación. Finalmente, en el capítulo 6 se presentan las conclusiones y algunas ideas para el futuro.

2 Fundamentos

2.1 Sistemas de apoyo a la toma de decisiones

Los sistemas de apoyo a la toma de decisiones se encargan de tomar decisiones automáticamente, pero en un ámbito más común sirven de herramienta para facilitar al usuario en la toma de decisiones. Por ello, son muy utilizados cuando existe una gran cantidad de factores a tomar en cuenta para tomar una decisión o cuando existe una gran cantidad de resultados, obteniendo la mejor opción al llegar a una decisión. Estos sistemas permiten reproducir el conocimiento de un experto, así como la forma en que se razona la solución de un problema o la toma de una decisión.

Para conocer la arquitectura de dichos sistemas, realizamos un estudio de campo. Las fuentes en que se basa este estudio de campo contemplan la bibliografía consultada en libros de la temática [16], estudios realizados en la metodología y tipología de los sistemas inteligentes [12], además de nuestra propia experiencia en el desarrollo de dichos sistemas.

Con base en dicho estudio de campo, detectamos que la arquitectura genérica que captura la funcionalidad de dichos sistemas puede ser representada a partir de tres módulos básicos:

- Interfaz del Usuario: establece la interacción hombre-máquina, facilitando la comunicación entre el núcleo del sistema y el usuario, haciendo uso de imágenes y accesorios amigables con el usuario, como los botones, cuadros de texto, cursor, imágenes, listas, etc.
- Base de Conocimientos: contiene el conocimiento del dominio de aplicación, almacenando la información de hechos invariantes y reglas. Este módulo integra al módulo de Base de Hechos o Memoria de Trabajo con el módulo de Base de Conocimientos, que algunos autores separan.
- Motor de Inferencia: establece el proceso de inferencia y aplica una estrategia de razonamiento para tomar una decisión. Este módulo contiene una serie de normas y meta-reglas que guían al sistema para esclarecer si un escenario es posible, descartándolo en caso contrario hasta llegar al único que concuerde con los criterios establecidos.

2.2 Modelo de Características de una Línea de Productos

En una línea de productos se comparten y gestionan un conjunto común de características que satisfacen las necesidades específicas de un mercado seleccionado. La variabilidad de una línea de productos puede ser plasmada mediante un Modelo de Características [6]. Los modelos de características son modelos formales que utilizan características para especificar productos, mostrando semejanzas y variaciones de varios objetos de una línea de productos, expresados por aspectos visibles de un producto o características del dominio. Por ello, se describen todas las posibles variaciones en una línea de productos, con la finalidad de definir un producto mediante de la selección de características, en una configuración.

Batory [2] provee dos definiciones para el modelo de características: 1) como un conjunto de características organizadas jerárquicamente y 2) como un lenguaje para especificar productos en una línea de productos. Donde una característica es "una característica del producto que es usado para distinguir un producto de una familia de productos relacionados" [3]. Se distinguen tres características principales: i) las comunes: una característica (funcional o no funcional) puede ser común en todos los productos en la línea de producto; ii) las variables: una característica puede ser común en algunos productos, pero no en todos; y iii) las específicas del producto: una característica puede ser parte de solamente un producto; estas características son pocas veces requeridas por el mercado, pero son requeridas por el interés de clientes con necesidades específicas.

Las características también son llamadas puntos de variabilidad. Éstos se despliegan en variantes que determinan el producto seleccionado de la línea de productos (como celulares, equipo de cómputo y periféricos, línea blanca, automóviles, etc.).

3 La generación automática de sistemas de apoyo a la toma de decisiones

Básicamente nuestro trabajo consiste en crear una aplicación que permite desarrollar automáticamente sistemas de apoyo para la toma de decisiones, con el fin de que sirvan de apoyo a los usuarios (clientes o vendedores) en la selección de un producto específico y único de una línea de productos, con las características particulares que se desean, y con la confianza de que dicho producto existe.

Dicha aplicación consiste en dos herramientas: la primera es denominada MODELER, la cual permite al usuario el diseño y la edición de modelos de características. La segunda herramienta denominada GENERATOR, permite al usuario cargar el modelo de características creado con MODELER y, con base en dicho proceso, generar dos archivos: uno donde se almacena el código fuente del sistema de apoyo a la toma de decisiones, y otro donde se encuentra el archivo ejecutable del sistema.

Considerando que los modelos de características son la clave técnica de las líneas de productos para desarrollar *software* [2], embebimos nuestro Modelo de Características en la Base de Conocimientos. De esta forma, las variantes de las características se corresponden con los hechos, mientras que las relaciones de restricción y de dependencia se corresponden con las reglas.

Para desarrollar el sistema, utilizamos el modelo funcional con una arquitectura de tres módulos: la Base de Conocimientos (que embebe al Modelo de Características), el Motor de Inferencia (que aplica estrategias de razonamiento deductivo), y la Interfaz del

Usuario (que permite la comunicación del usuario con el sistema, de forma sencilla y amigable). Las restricciones del dominio son capturadas en el Modelo de Características.

Dado que seleccionamos el dominio de los teléfonos celulares comerciales, como nuestro caso de estudio, se realizó un análisis del dominio, con el fin de obtener los requisitos del usuario y modelar la variabilidad de dicha línea de productos (*cellphone*), mediante el modelo de características de la figura 1, el cual está formado por:

- 14 características o puntos de variabilidad (*color, camera, resolution, video, flash, screen, size, type, memory, hand writing, battery, system, headphones, radio*), donde dichos puntos de variabilidad cuentan con variantes (por ejemplo, *color* tiene cuatro variantes: *gold, red, black, white*) ,
- 3 relaciones de estructura (*xor, mandatory, optional*),
- 3 relaciones de restricción (*implies, bi-implies, excludes*),
- 1 relación de dependencia (*uses*),
- 1 característica de referencia (*internal*).

Todas estas relaciones aseguran que, entre las distintas variantes disponibles del dominio, se seleccionará un único producto existente, al incluir o excluir las características de acuerdo a las alternativas que son presentadas.

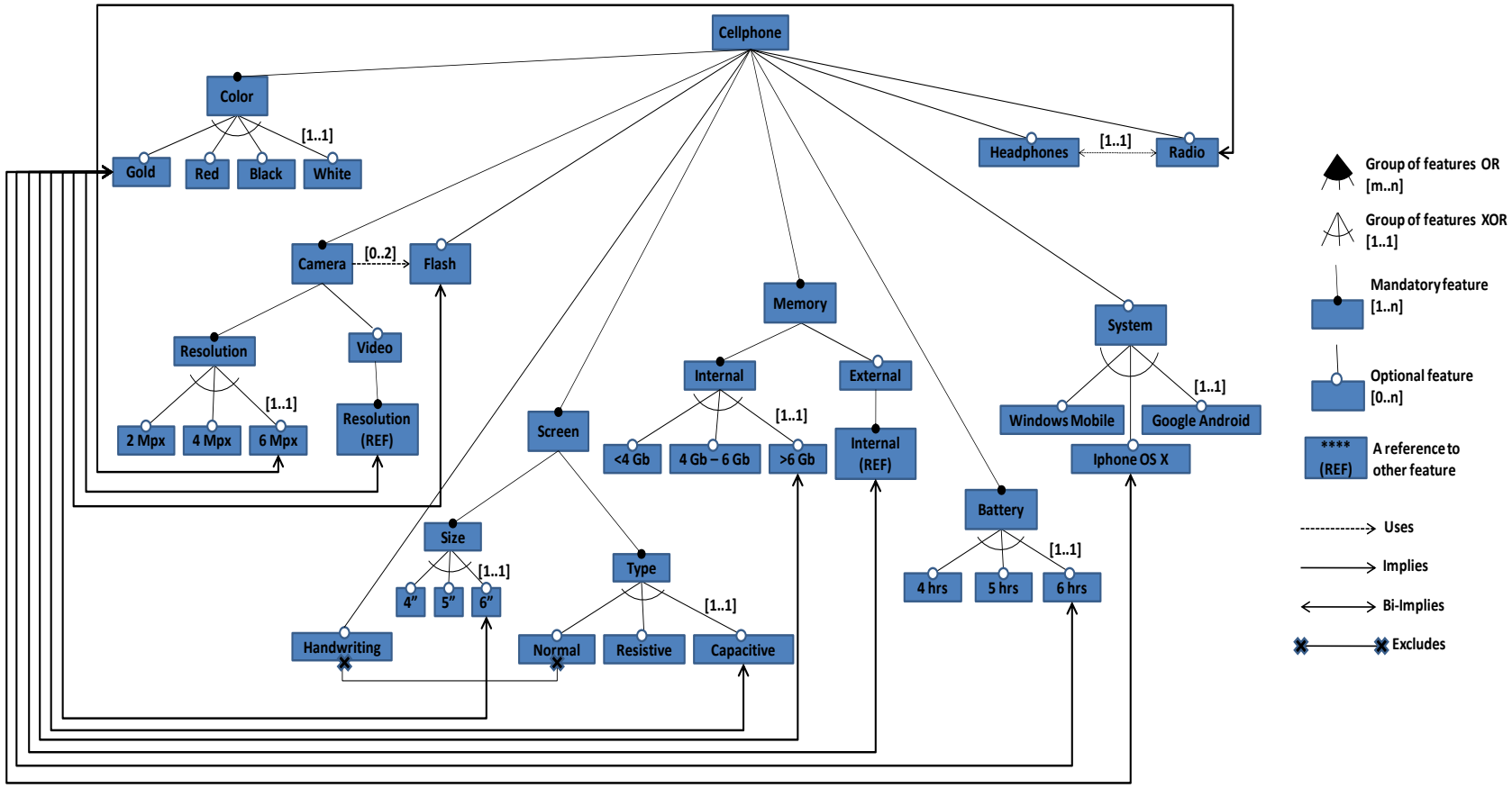


Fig. 1. Modelo de características de nuestra línea de productos de celulares

4 Implementación

Se desarrolló una aplicación utilizando la tecnología WPF (*Windows Presentation Fundation*) que permite el diseño de un modelo de características. Éste genera un archivo XML [18], especificando características de la línea de productos y sus restricciones. Dicho archivo cumple con dos tareas fundamentales: permitir la regeneración del modelo de características para su edición, y la generar automáticamente el sistema de apoyo para la toma de decisiones (implementado en C# [5]) para ser visualizado por medio de una GUI (*Windowsforms*), con la que trabajará el usuario para seleccionar su producto. Paralelo a la generación del sistema y su interfaz gráfica, se genera un archivo con el código fuente de la aplicación, el cual puede ser manipulado haciendo uso de herramientas como *Visual Studio 2010* [17].

El funcionamiento de la aplicación consiste en crear el modelo de características de la línea de productos (celulares, coches, línea blanca, equipo de cómputo, etc.), usando el MODELER, el cual cuenta con las siguientes funciones básicas: i) permite crear un modelo de características; ii) permite guardar un modelo de características diseñado con el MODELER; y iii) permite abrir un modelo de características creado anteriormente con el MODELER o cualquier modelo con estructura XML

El modelo de características se almacena en un archivo que contiene toda la información necesaria para que el GENERATOR realice su tarea.

La herramienta GENERATOR funciona identificando y separando las características contenidas en el modelo y las relaciones que existen entre las mismas. Dichas relaciones (que han sido expresadas en XML) son reemplazadas por código C#, que brinda la funcionalidad necesaria para ser usados por el usuario final. Los nombres de los *features* son asignados al atributo TEXT y CAPTION de objetos LABELS, y las relaciones entre *features* son reemplazadas por objetos como LISTBOX, COMBOBOX y código programado (IF, ELSE, ELSE IF, CASES, etc.). Una vez realizado dicho reemplazo, se genera el ejecutable del sistema de apoyo a la toma de decisiones (producto final de la línea de productos *software*) mostrado en la figura 4.

La aplicación fue desarrollada con *Microsoft Visual Studio 2010*, bajo la plataforma .Net (versión 4.0). Su arquitectura se muestra en la figura 2:

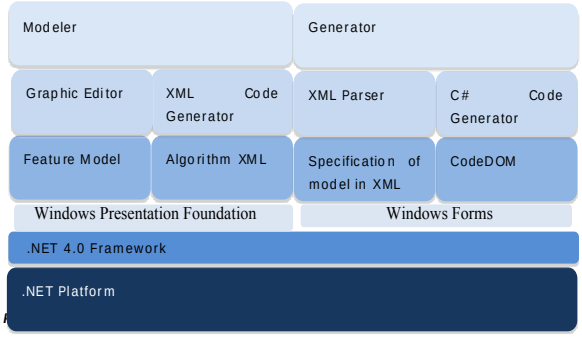


Fig. 2. Arquitectura de la aplicación.

4.1 Un ejemplo implementado

Con el fin de mostrar nuestra aplicación, hicimos uso del ejemplo del modelo de características de los celulares de la figura 1.

Primeramente diseñamos el modelo de características con la herramienta MODELER, como se muestra en la figura 3. Este modelo lo guardamos en un archivo XML, y lo cargamos en la herramienta GENERATOR, para obtener el ejecutable del sistema de apoyo a la toma de decisiones mostrado en la figura 4.

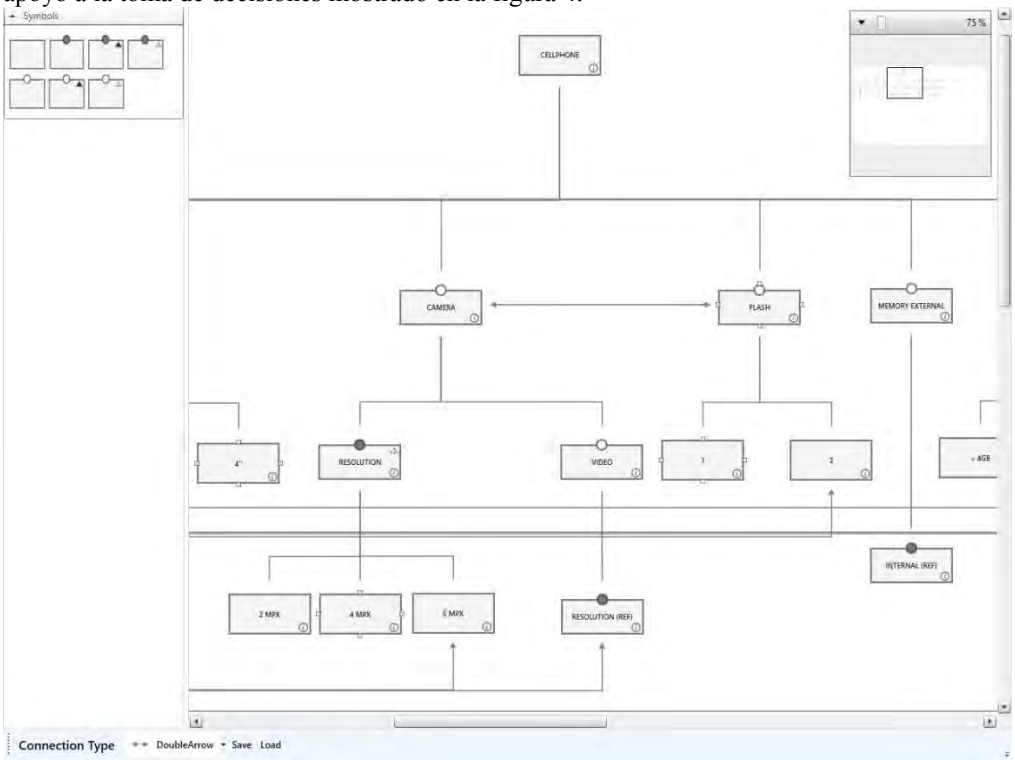


Fig. 3. Modelo de características del dominio de celulares creado en el MODELER.

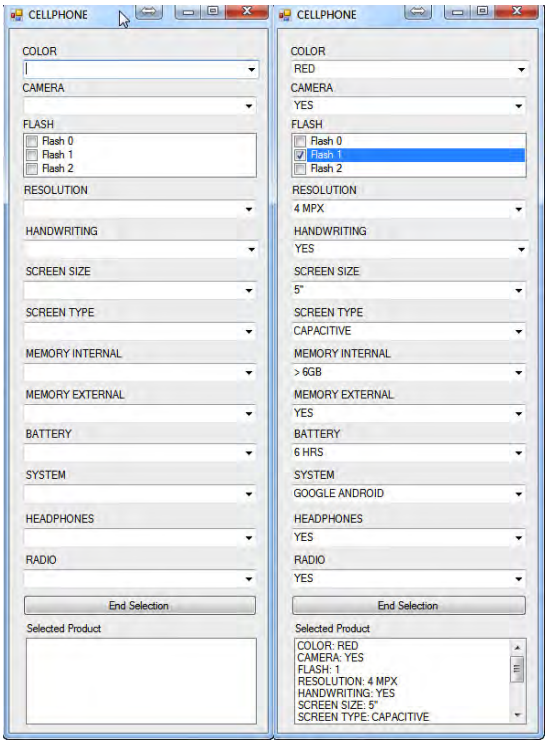


Fig. 4. Sistema de apoyo a la toma de decisiones resultante del modelo de características de celulares (izquierda), y el mismo sistema con un producto seleccionado (derecha).

4.2 Pruebas

Para evaluar la aplicación *software* (presentada ampliamente en [14]) se utilizó la metodología de Davis [8], que requiere realizar una encuesta llamada modelo de aceptación de nuevas tecnologías[19]. La encuesta fue administrada a usuarios poseedores de conocimiento básico sobre variabilidad en las líneas de productos y con conocimiento teórico y práctico en la creación de sistemas de apoyo a la toma de decisiones.

Se seleccionaron a 20 usuarios (ingenieros de la aplicación), entregándoles el modelo de características impreso (con nivel de complejidad baja) de una línea de productos de celulares (figura 1), y solicitándoles diseñar dicho modelo con la herramienta MODELER y generar el ejecutable con la herramienta GENERATOR.

De acuerdo a Nielsen [13], para evaluar un software es necesario tomar en cuenta tres criterios: precisión, tiempo y respuesta emocional. La evaluación de estos criterios de manera individual ayudó a determinar la usabilidad y facilidad de uso del sistema.

Con los resultados obtenidos durante las pruebas de precisión, se observa que nuestra herramienta no obtiene los valores esperados (al menos del 50%) en la generación de sistemas en el primer intento. Sin embargo, la aplicación de la prueba fue provechosa, dado que se observaron los errores más comunes por parte de los usuarios. Esto permitirá en futuras versiones realizar ajustes tomando en consideración dichas observaciones. En contraste, la segunda prueba de precisión excedió las expectativas al observarse que el 100% de los usuarios lograron generar un sistema.

[19] También conocido por sus siglas en inglés TAM (Technology Acceptance Model)

Respecto al tiempo que toma generar un sistema, nuestra herramienta retornó buenos resultados. Las pruebas mostraron que la herramienta permite a los ingenieros de la aplicación el generar sistemas mucho más rápido (31:20 minutos) que el método tradicional (4:19 horas). De igual manera, el editar y modificar los sistemas fue más eficiente haciendo uso de nuestra aplicación, ya que tomó tan solo 3 minutos en comparación a los 27 minutos que toma sin hacer uso de la herramienta.

En cuanto a la usabilidad y la facilidad de uso, nuestra herramienta fue evaluada con buenos resultados. La métrica de usabilidad muestra que el 56% encuentran el prototipo extremadamente probable, el 36% algo probable y el 8% poco probable. Por su parte la facilidad de uso muestra que el 61% dicen que es extremadamente probable, el 25% algo probable y el 14% poco probable. Lo anterior nos permite percibir la actitud de los usuarios hacia el prototipo. Se considera que el promedio de los valores arriba del 50% son buenos indicadores generalmente, y que muestran una actitud favorable hacia la herramienta en cuanto a la usabilidad y facilidad de uso.

5 Trabajos Relacionados

Existe una gran variedad de trabajos relacionados con el desarrollo de sistemas inteligentes, así como con el modelado de la variabilidad de una línea de productos, entre los que se encuentran varios relacionados con nuestro trabajo.

En un estudio realizado por Liao [12] menciona que el desarrollo de los sistemas de apoyo a la toma de decisiones está caracterizado por la separación del conocimiento y procesamiento en unidades independientes. En la arquitectura de nuestro modelo, los elementos son definidos tomando en cuenta este concepto, específicamente porque hay un componente que contiene la información del dominio y otro componente que ejecuta el proceso de inferencia.

Bachmann et al. [1] proponen separar la declaración de la variabilidad de los artefactos software. En nuestro trabajo, la separación de la variabilidad y la funcionalidad son contenidas en modelos conceptuales separados. El uso de un modelo de características embebido en el sistema permite que el usuario introduzca la información de las características, que contribuyen a obtener el producto seleccionado.

Algunos autores han propuesto una semántica formal para los modelos de características. Batory, en [2] expresa las características del dominio en un modelo de características, y en [3] propone una semántica formal para dichos modelos, basada en formulas proposicionales y gramática. Bontemps [4] explora la expresividad de las notaciones de un modelo de características. Estos trabajos han sido considerados en nuestro modelo de características.

Asimismo se han propuesto varias extensiones en BFM para incrementar la capacidad descriptiva y completar la formalización de los modelos de características. Czarnecki [7] incluye atributos, clonaciones y referencias de atributos como extensiones de las características básicas, y en [6] propone una notación basada en la cardinalidad. Gómez y Ramos [9] incluyen cardinalidad y limitaciones en los modelos de características. Nuestro modelo de características comparte las notaciones de Gómez y Czarnecki, pero incluye algunas diferencias en las relaciones estructurales y en la cardinalidad.

Respecto al diseño y edición de los modelos de características, existen algunos de trabajos relacionados al nuestro, como el *Feature Model DSL* [11] que permite la creación y validación de modelos de características, navegación entre diagramas y su exportación en archivos PNG; el *Feature Diagram Editor* [10] que hace uso de la sintaxis de Czarnecki y Eisenecker; y el *Feature IDE* [15] que soporta análisis de

dominio, implementación de dominios, análisis de requerimientos y generación de software.

Estos trabajos son *plug-ins*, por lo que no son aplicaciones independientes, sino extensiones, y requieren de otros programas para funcionar. *Feature IDE* es el más cercano a nosotros, debido a que dicho *plug-in* genera código; sin embargo, no es funcional y requiere mucho trabajo para mejorar su usabilidad. Los otros dos proyectos se enfocan en el diseño y la edición de modelos de características, pero no generan código o aplicaciones ejecutables. Nuestra propuesta únicamente hace uso del *framework* .Net, con la diferencia de que sí genera código y un archivo ejecutable.

Además, de acuerdo a nuestra consulta en la literatura, no existen trabajos que desarrollen sistemas de apoyo a la toma de decisiones con modelos de variabilidad embebidos, ni con la generación automática de estos sistemas con esta modalidad.

6 Conclusiones y Trabajo Futuro

Nuestra propuesta utiliza estándares internacionales para el desarrollo del software (ingeniería dirigida por modelos), y aplica técnicas de línea de productos software para el desarrollo de productos que comparten elementos entre sí. De tal forma que se ha logrado desarrollar una aplicación que genere automáticamente un sistema, a partir de un modelo de características de un dominio específico.

Para ilustrarla, hemos seleccionado una línea de productos de teléfonos celulares, como caso de estudio, sin embargo puede ser usado para otros dominios, debido a que la aplicación desarrollada permite cambiar el modelo de características del dominio de aplicación. Esto implica la generalidad de la herramienta para ser utilizada en cualquier línea de productos, *i.e.*, se podrán crear sistemas en diferentes dominios, minimizando tiempos y costos de producción en el desarrollo del software al desarrollar productos que comparten elementos entre sí (*i.e.*, al reutilizar "paquetes de software").

Un sistema generado con nuestra aplicación, dará soporte a los clientes que desean comprar un producto (un teléfono celular), siendo de gran utilidad a la compañía que utilice este software para vender su producto de forma más eficiente, al reducir tiempos y con la seguridad de que las características de la elección del cliente implicará que el producto existe en la línea de productos. Esto es posible dada la verificación que realiza el sistema, con base en las restricciones del dominio que son plasmadas en el modelo de características. Con ello, el sistema generado podrá ser considerado como una herramienta útil y confiable para seleccionar un producto específico de una línea de productos.

Las principales aportaciones de nuestro trabajo al desarrollo del software son: i) embeber el modelo de características dentro del sistema; ii) desarrollar una aplicación que genere automáticamente un sistema de apoyo a la toma de decisiones, con solo crear el modelo conceptual que plasma la variabilidad de un dominio específico; y iii) ofrecer la certeza de la existencia y unicidad del producto seleccionado por el usuario del sistema.

Cabe mencionar que aunque existe una gran variedad de trabajos relacionados con el desarrollo de sistemas inteligentes y con el modelado de la variabilidad de una línea de productos, no se encontraron, en la literatura consultada, trabajos que desarrollen sistemas de apoyo a la toma de decisiones con modelos de variabilidad embebidos, ni con la generación automática de dichos sistemas con esta modalidad.

Como trabajo futuro, deseamos agregar a la herramienta MODELER, un editor visual utilizando un Lenguaje Específico de Domino20 para crear el modelo de características, dando facilidad a los usuarios poco experimentados de interactuar con el sistema. Éste se validará con ayuda de algunos vendedores de celulares de la compañía seleccionada. Finalmente, es necesario para validar el sistema experto ejecutable por medio de pruebas de campo con vendedores de automóviles que puedan juzgar sobre la usabilidad y confiabilidad del sistema.

Referencias

[1]. Bachmann F., Goedicke M., Leite J., Nord R., Pohl K., Ramesh B., and Vilbig A.: A meta-model for representing variability in product family development. In: Proc. 5th International Workshop on Product Family Engineering (PFE'5), 66-8 (2003).

[2]. Batory D., Benavides D., and Ruiz-Cortés A.: Automated Analyses of Feature Models: Challenges Ahead. Communications of ACM on Software Product Lines (2006).

[3]. Batory D.: Feature models, grammars and propositional formulas. Technical Report TR-05-14, University of Texas, Austin, Texas, USA (2005).

[4]. Bontemps Y., Heymans P., Schobbens P.Y., and Trigaux J.C.: Semantics of FODA feature diagrams. In: Proc. SPLC´04 Workshop on Software Variability Management for Product Derivation–Towards Tool Support, 48-58 (2004).

[5]. Carol H., Matt T., Nancy R., Yariv R., Emily H., Leslie H. L., Laurie O. and Monique V.: C# Complete: A clear picture of everything you need to know for developing applications using C#. ISBN: 0-7821-4203-6. United States of America (2003).

[6]. Czarnecki K. and Kim C.: Cardinality-based feature modeling and constraints: a progress report. In: Proc. OOPSLA International Workshop on Software Factories, San Diego, California, USA (2005).

[7]. Czarnecki K., Kim C., and Kalleberg K.: Feature Models are Views on Ontologies. In: Proc. 10th International Software Product Line Conference, USA (2006).

[8]. Davis, F.: Perceived Usefulness, Perceived ease of use and user acceptance of information technology. USA: University of Michigan (1989).

[9]. Gómez A. and Ramos I.: Cardinality-based feature models and model-driven engineering: fitting them together. In Proceedings of Fourth International Workshop on Variability Modelling of Software-Intensive Systems (2010).

[10].KIT: Karlsruhe Institute of Technology, Feature Diagram Editor, 2012, https://sdqweb.ipd.kit.edu/wiki/Feature_Diagram_Editor.

[11].Lenz G. and Wienands C., Practical Software Factories in.NET. USA: Apress, 2006.

[12].Liao S.-H.: Knowledge Management Technologies and Applications- Literature Review from 1995-2002. In Expert Systems with Applications, Vol. 25, Issue 2, 155-164 (2003).

[13].Nielsen, J.: Usability Engineering . USA: Morgan Kaufmann (1993).

[14].Preciado, F.: Generación automática de Sistemas Expertos basada en Líneas de Productos Software. Tesis de Maestría en Tecnologías de la Información, Universidad de Colima (2013)

[15].Thüm T., Kästner C., Benduh F., Meinicke J., Saake G., and Leich T., FeatureIDE: An Extensible Framework for Feature-Oriented Software Development, Science of Computer Programming, 2012.

[16].Turban T, and Aronson J. E: Decision Support System and Intelligent System, Sixth Edition, ISBN: 0-13-089465-6, United States of America, (2001).

[17].Visual Studio. http://www.microsoft.com/spain/visualstudio.

[18].XML: http://www.csharpkey.com/csharp/xml/.

[20] Del inglés Domain Specific Language y siglas DSL

Análisis empírico de la influencia de los rasgos de la personalidad en los profesionales de la ingeniería de software.

[1]Perla Ivet Jarillo Nieto, [2]Carlos Enríquez Ramírez y [3]Roberto Sánchez Herrera
Universidad Politécnica de Tulancingo, Calle Ingenierías # 100. Col. Huapalcalco, Hidalgo, México C.P. 43629. México
perla.jarillo@upt.edu.mx[1]
carlos.enriquez@upt.edu.mx[2]
roberto.sanchez@upt.edu.mx[3]

Resumen. El interés en mejorar la calidad de los productos que se entrega al mercado a llevado a tomar preocupación en la identificación de las características de los profesionistas que se emplean en el desarrollo de software. En este trabajo se ha realizado un análisis de la literatura en torno al comportamiento del factor humano en las diversos componentes del desarrollo de software, que van desde la ocupación de roles, uso de herramientas, procesos o técnicas. Los hallazgos se describen con la finalidad de conocer la importancia que tiene el tema en la ingeniería de software de acuerdo a estudios dirigidos por expertos en el área.

Palabras claves: Rasgos de personalidad, factor humano, ingeniería de software, modelo de los cinco grandes.

1 Introducción

La importancia del factor humano en la ingeniería de software ha sido abordada por expertos investigadores en la materia bajo diferentes perspectivas durante la última década. En el presentre trabajo se ha realizado un análisis de publicaciones relacionadas al factor humano en el ámbito de la ingeniería de software con la finalidad de obtener información actual al respecto para tener información que aporte valor. La selección de los artículos se hizo a través de contenedores de información como Science Direct, IEEE, EBSCO y ACM DL Digital Library. Para la búsqueda de las publicaciones se utilizarón las palabras clave relacionadas con el tema, tales como "factor humano", "rasgos de la personalidad" y "modelo de los cinco grandes y desarrollo de software" tópicos con los cuales esta investigación se condujo.

2 Revisión de literatura

Los estudios consultados en los que se encuentra una relación entre la personalidad y la ingeniería de software que dan soporte a la investigación son mostradas en la tabla 1.

Referencia	Instrumento psicométrico	Variables de la Ingeniería de Software	Resumen
(1)	16PF (*Fifth Edition Personality Factor Test*)	Asignación de roles.	Evalúa los rasgos de personalidad de los individuos que forman parte de un equipo de desarrollo de Software.

Referencia	Instrumento psicométrico	Variables de la Ingeniería de Software	Resumen
(2)	No aplica	Rendimiento en los desarrolladores.	Verifica los motivos de un buen desempeño en desarrolladores y el impacto que esto tiene en el logro de los proyectos.
(3)	FFM	Asignación en de roles.	Capta los requisitos en la personalidad de los individuos en Ingeniería de software.
(4)	No aplica	Productividad de los desarrolladores.	Realiza un análisis de fuentes de información con el fin de extraer una lista de los principales factores que influyen en la productividad.
(5)	No aplica	Fases de la gestión de la Ingeniería de software.	Revisión de literatura para analizar los factores humanos las fases de la gestión de la Ingeniería de software.
(6)	FFM	Asignación de roles	Propone un modelo de Sistema de Inferencia Difuso Basado en una Red adaptativa (ANFIS), donde se localizan patrones del tipo de personalidad.
(7)	MBTI	Proceso de desarrollo de software.	Para cada etapa del desarrollo de software se describen las habilidades deseadas y esperadas.
(8)	Elaboración propia para identificar las dimensiones culturales.	Cultura y su relación con la tutoría.	Analiza la influencia de la cultura en las relaciones de tutoría dentro de la industria de la ingeniería de software.
(9)	No aplica.	Proceso de desarrollo	Revisa la literatura relacionada con las prácticas de enfoque de desarrollo ágil que se deben realizar en la producción de software de calidad.
(10)	No aplica	Resultados del proyecto.	Identifica y clasifica los factores que han demostrado tener un impacto en los resultados del proyecto.
(11)	Elaboración propia	Eficacia de un proyecto de software de código abierto.	Desarrolla un modelo teórico con el objetivo de explorar la eficacia de un proyecto de software de código abierto (OSS) (en términos de tamaño del equipo y trabajo colaborativo) se ve afectada por la integración de conocimientos.
(12)	*Ten Item Personality Inventory* (TIPI)	Apego a la tecnología de comunicación.	Define el tipo de personalidad de acuerdo al modelo de los cinco rasgos de la personalidad, basado en el uso de *smartphones*.
(13)	NEO-PI-R	Rendimiento del desarrollador.	Identifica las características de carácter humano que influyen en el rendimiento del individuo, para predecir la actuación humana en el desarrollo de software.
(14)	Construcción propia.	Motivación de los ingenieros de software.	Estudia los factores que influyen en la motivación de los ingenieros de software con el objetivo de orientar la definición de programas que los refuercen.
(9)	Elaboración propia.	Factores culturales y el desarrollo de software.	Examina el papel de los factores culturales en el desarrollo de software, tales como la comunicación, distribución de información sobre el proyecto, comportamientos y actitudes.
(15)	Elaboración propia.	Competencias técnicas descritas por *Software Engineering Body of Knowledge* (SWEBOK) y profesionales de software.	Prueba las brechas de competencias entre los profesionales de software para identificar la carencia de habilidades técnicas en la ingeniería de software relacionadas con el factor humano.
(16)	No aplica	Asignación de roles.	Revisión de literatura y comparación de FFM y MBTI para la asignación de roles en la ingeniería de software.
(17)	MBTI	Asignación de roles.	Explora los rasgos de personalidad de los profesionales del desarrollo de software que se relacionan con la adopción de roles en ingeniería de software.
(18)	No aplica	Retos en el desarrollo de software.	Identifica los retos relacionados con los factores humanos en GSD(*Global Software Development*).
(19)	No aplica	Construcción de proyectos.	Se centra en la comparación de los factores humanos que mencionan algunos autores reconocidos a lo largo de varias décadas.
(20)	NEO PI-3	Pruebas de software.	Determina empíricamente las relaciones entre los rasgos específicos de la personalidad y el desempeño en las pruebas de software

Tabla 1. Resumen de la literatura relacionada con la conexión entre personalidad y la ingeniería de software (producción propia).

Cada uno de los artículos analizados se encuentra en orden cronológico, se menciona la variable analizada y si se aplica o no un instrumento psicológico, a fin de que sea posible identificar fácilmente la aportación que ha hecho a la ingeniería de software.

3 Factores humanos mencionados en la literatura consultada

Después de realizar un análisis minucioso en los artículos seleccionados se identificaron 69 factores humanos que impactan de manera positiva o negativa en el desarrollo de software. El detalle de los factores humanos mencionados por cada uno de los autores es mostrado en la tabla 2.

Factor	Fuente																					% Menciones en la literatura
	((1)	(2)	(3)	(4)	(5)	(6)	((7)	(8)	(21)	(10)	(11)	(12)	(13)	(14)	(9)	(15)	(16)	(17)	(18)	((19)	(20)	
1. Capacidad de análisis																						11.59%
2. Toma de decisiones																						7.25%
3. Trabajo autónomo																						10.14%
4. Innovación y creatividad																						8.70%
5. Juicio																						5.80%
6. Tenacidad																						2.90%
7. Tolerancia al estrés																						5.80%
8. Auto-organización																						11.59%
9. Gestión de riesgos																						1.45%
10. Conocimiento del entorno de trabajo																						7.25%
11. Disciplina																						4.35%
12. Habilidad para entender el entorno de trabajo																						4.35%
13. Servicio al cliente																						4.35%
14. Habilidades de negociación																						4.35%
15. Empatía																						14.49%
16. Sociabilidad																						18.84%
17. Trabajo en equipo																						18.84%
18. Evaluación a compañeros de trabajo																						2.90%
19. Liderazgo																						5.80%
20. Planificación y organización																						8.70%
21. Apertura al cambio																						15.94%
22. Adaptabilidad																						2.90%

Factor	Fuente																					% Menciones en la literatura
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(21)	(10)	(11)	(12)	(13)	(14)	(9)	(15)	(16)	(17)	(18)	(19)	(20)	
23. Disposición																						1.45%
24. Proactividad																						2.90%
25. Solución de problemas																						13.04%
26. Curiosidad por probar nuevas tecnologías																						10.14%
27. Pensamiento crítico																						5.80%
28. Comunicación oral y escrita																						14.49%
29. Sensibilidad																						7.25%
30. Preocupación por la calidad																						2.90%
31. Administración del conocimiento																						7.25%
32. Confianza																						10.14%
33. Solución de malentendidos culturales y de idioma																						1.45%
34. Motivación																						5.80%
35. Destrezas de cálculo																						1.45%
36. Disponibilidad para recibir retroalimentación constante																						7.25%
37. Participación en todo el ciclo de vida de un proyecto																						1.45%
38. Ejercicio de prácticas de desarrollo de software																						2.90%
39. Empoderamiento																						1.45%
40. Hacer contribuciones/tareas significativas																						1.45%
41. Identificación con las tareas																						1.45%
42. Balance vida/trabajo																						1.45%
43. Trayectoria profesional																						4.35%
44. Cultura de la organización																						1.45%
45. Respeto del reglamento e instrucciones																						1.45%
46. Género																						2.90%
47. Orientación temporal largo plazo/corto plazo																						1.45%
48. Credibilidad																						2.90%
49. Respeto																						5.80%
50. Justicia																						1.45%

Factor	Fuente																					% Menciones en la literatura
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(21)	(10)	(11)	(12)	(13)	(14)	(9)	(15)	(16)	(17)	(18)	(19)	(20)	
51. Claridad en las metas																						1.4 5%
52. Capacidad del programador (habilidades del programador)																						2.9 0%
53. Experiencia de aplicaciones (experiencia con el dominio de la aplicación)																						2.9 0%
54. Experiencia en la plataforma																						2.9 0%
55. Experiencia en el lenguaje y herramientas de desarrollo																						4.3 5%
56. Experiencia como administrador de una aplicación																						2.9 0%
57. Competitividad																						2.9 0%
58. Habilidad para lidiar con la ambigüedad																						1.4 5%
59. Compromiso con las actividades diarias																						5.8 0%
60. Capacidad de respuesta rápida (receptivo)																						1.4 5%
61. Normas, valores, creencias y supuestos																						1,4 5%
62. Reciprocidad																						1,4 5%
63. Permanencia																						1,4 5%
64. Obediencia																						2,9 0%
65. Asertividad																						5,8 0%
66. Ansiedad																						5,8 0%
67. Depresión																						5,8 0%
68. Atención a los detalles																						4,3 5%
69. Hostilidad																						4,3 5%

Tabla 2 Factores humanos mencionados en los trabajos consultados.

El autor que menciona un mayor número de factores humanos es (16), con un total de (23), seguido por (3) con 21 y (1) con 20. Se identificó que los rasgos mencionados en la literatura consultada están relacionados con capacidades técnicas, procesos, capacidades y experiencias y/o cultura/calidad.

Capacidades técnicas

En diversos estudios es mencionada la importancia de que los ingenieros de software cumplan con habilidades técnicas que faciliten la realización de las tareas encomendadas (4). Los factores clasificados como capacidades técnicas son:

- Curiosidad por probar nuevas tecnologías.
- Conocimiento del entorno de trabajo.
- Habilidad para entender el entorno de trabajo.
- Ejercicio de prácticas de desarrollo de software.
- Capacidad del programador (habilidades del programador).
- Experiencia de aplicaciones (experiencia con el dominio de la aplicación).
- Experiencia en la plataforma.
- Experiencia en el lenguaje y herramientas de desarrollo.
- Experiencia como administrador de una aplicación.

Procesos

Existen algunos factores que están estrechamente relacionados con los procesos en la ingeniería de software, tales como la planificación y organización, la evaluación entre compañeros o la gestión de riesgos, que pueden impactar en aspectos organizacionales (1). De acuerdo al análisis de la literatura se proponen los siguientes factores en esta categoría:

- Planificación y organización.
- Gestión de riesgos.
- Evaluación a compañeros de trabajo.
- Participación en todo el ciclo de vida de un proyecto.

Capacidades y experiencias

Son factores que están relacionados con los individuos y denotan su habilidad para resolver problemas en el desarrollo de software basados en experiencias previas o en capacidades naturales de su personalidad (4). Esta categoría está compuesta por los factores:

- Capacidad de análisis.
- Toma de decisiones.
- Trabajo autónomo.
- Innovación y creatividad.
- Tenacidad.
- Tolerancia al estrés.
- Auto-organización.
- Disciplina.
- Servicio al cliente.
- Habilidades de negociación.
- Liderazgo.
- Adaptabilidad.
- Proactividad.
- Solución de problemas.
- Pensamiento crítico.
- Administración del conocimiento.
- Destrezas de cálculo.
- Hacer contribuciones/tareas significativas.
- Identificación con las tareas.
- Trayectoria profesional.
- Habilidad para lidiar con la ambigüedad.
- Capacidad de respuesta rápida (receptivo).
- Ansiedad.
- Depresión.
- Atención a los detalles.

Cultura/calidad

Incluye factores que involucran a grupos de personas y la forma en que se comportan entre ellos. Wagner menciona que hay factores de cultura coorporativa, aquellos que están en un nivel más en toda la compañía, y factores de cultura de equipo, los que denotan factores similares a nivel de equipo (4). Esta categoría está compuesta por:

- Juicio.
- Empatía.
- Sociabilidad.
- Trabajo en equipo.
- Apertura al cambio.
- Disposición.
- Comunicación oral y escrita.
- Sensibilidad.
- Preocupación por la calidad.
- Confianza.
- Solución de malentendidos culturales y de idioma.
- Motivación.
- Disponibilidad para recibir retroalimentación constante.
- Empoderamiento.
- Balance vida/trabajo.
- Cultura de la organización.
- Respeto del reglamento e instrucciones.
- Género.
- Orientación temporal largo plazo/corto plazo.
- Credibilidad.
- Respeto.
- Justicia.
- Claridad en las metas.
- Competitividad.
- Compromiso con las actividades diarias.
- Normas, valores, creencias y supuestos.
- Reciprocidad.
- Permanencia.
- Obediencia.
- Asertividad.
- Hostilidad.

La clasificación propuesta nos permite identificar claramente que los autores de la literatura analizada coinciden en la importancia de los factores centrando su atención en los cuatro grupos.

5 Conclusiones y trabajos futuros

El número de artículos coincidentes con el tema demuestra que es de interés para los investigadores y cada año surgen más estudios relacionados con los rasgos humanos y el desarrollo de software. Se puede destacar que el 100% de los autores consultados señala al menos un factor humano relacionado con la cultura y/o la calidad, que son aspectos que tienen que ver con el contexto social en el cual se desenvuelve un individuo y la formación que ha recibido, incluso, desde su núcleo familiar. En esta categoría se encuentra el trabajo en equipo y la sociabilidad, que son los factores más mencionados en los artículos. Los factores humanos mencionados con más frecuencia en la literatura revisada son el trabajo en equipo, y la sociabilidad, que fueron mencionados 13 veces en la literatura, lo cual representa un 18.84%, seguidos por la apertura al cambio con un 15.94%, la empatía y la comunicación oral o escrita, con un 14.49%. Dentro de los diéz factores más mencionados también se encuentra la solución de problemas, capacidad de análisis, auto-organización, trabajo autónomo, curiosidad por probar nuevas tecnologías y confianza. Como trabajos futuros se propone el planteamiento de un modelo que

permita identificar los factores humanos que pueden impactar en la adopción de procesos en el desarrollo de software.

6 Referencias

1. **Acuña, Silvia T., Juristo, Natalia y Moreno, Ana.***Emphasizing Human Capabilities in Software Development.* 2006, IEEE software, págs. 94-101.
2. **Baddoo, Nathan.***Software Developer Motivation in a High Maturity Company: a Case Study.* 2006, Software Process Improvement and Practice, págs. 219–228.
3. *An Improved Assessment of Personality Traits in Software Engineering.* **Sodiya, A.S.** 2007.
4. **Wagner, Stefan.** *A Systematic Review of Productivity Factors in Software Development.* Garching b, M¨unchen, Germany : s.n., 2008.
5. **Pirzadeh, Laleh.** *Human Factors in Software Development: A Systematic Literature Review.* 2010.
6. **Martínez, Luis G., y otros.** Big Five Patterns for Software Engineering Roles Using. *MICAI 2010.* 2010, págs. 428–439.
7. *Making Sense of Software Development and Personality Types.* **Capretz, Luiz Fernando y Ahmed, Faheem.** 2010, IEEE Computer Society, págs. 6-13.
8. **Casado-Lumbreras, Cristina.** *Culture dimensions in software development industry: The effects of mentoring.* 2011, Scientific Research and Essays Vol. 6(11), págs. 2403-2412.
9. **Mohammad, Al-Tarawneh** *Cultural factors: The Key Factors in software Development.***.** 2012, European Journal of Business and Management, págs. 113-123.
10. **Mcleod, Laurie.** *Factors that Affect Software Systems Development Project.* 2011, ACM Computing Surveys, págs. 24-56.
11. **Shih-Wei, Chou y Mong-Young, He.** *The factors that affect the performance of open source software development the perspective of social capital and expertise integration.* 2011, Info Systems J, págs. 195–219.
12. **Chittaranjan, Gokul, Blom, Jan y Gatica-Perez, Daniel.** *Who's Who with Big-Five: Analyzing and Classifying Personality Traits with Smartphones.* 2011.
13. **Georgieva, Konstantina.***Validation of the model for prediction of the human performance.* 2011.
14. **Fabio Q.B. da Silva, A. César C. Franc¸ a.***Towards understanding the underlying structure of motivational factors for software engineers to guide the definition of motivational programs.* 2012, The Journal of Systems and Software, págs. 216-226.
15. **Colomo-Palacios, Ricardo.** *Competence gaps in software personnel: A multi-organizational study.* 2012, Computers in Human Behavior.
16. **Rehman, Mobashar.** *Mapping Job Requirements of Software Engineers to Big Five Personality Traits.* 2012.
17. **Yilmaz, Murat Yilmaz y O'Connor, Rory V.** *Towards the Understanding and Classification of the Software Development Practitioners: Situational Context Cards Approach.* 2012.
18. **Misra, Sanjay.** *A discussion on the role of people in global software development.* 2013, Technical Gazette 20, págs. 525-531.

19. **Medinilla, Ángel.** *Agile Management: Leadership in an Agile Environment.* Mairena del Aljarafe : Springer, 2013. págs. 69-94.

20. **Kanij, Tanjila, Merkel, Robert y Grundy, John.** *An Empirical Study of the Effects of Personality on Software Testing.* 2013.

21. **Mohamed, Shafinah Farvin Packeer.** *Agile Software Development Practices that influence Software Quality: a review.* Bandung : s.n., 2011. Proceedings of the 3rd International Conference on Computing and Informatics. págs. 147-153.

Desarrollo de un Sistema de Información para apoyar el seguimiento académico de los alumnos de primaria

Italia Estrada[1], Mónica A. Carreño[2], Andrés Sandoval[3]

Universidad Autónoma de Baja California Sur, Carretera al Sur Km. 5.5,
La Paz B.C.S., 23080, México

{iestrada[1], mcarreno[2], sandoval[3]}@uabcs.mx

Resumen. Una de las actividades del docente de primaria es dar un seguimiento oportuno a cada uno de sus alumnos con la finalidad de estar siempre atento a la mejora en su proceso de aprendizaje y canalizarlo en caso de ser necesario a las instancias correspondientes para su atención. Este artículo reporta el desarrollo de un Sistema de Información para automatizar y apoyar el proceso de seguimiento académico oportuno; basándose en las técnicas, herramientas y métodos que proporciona la Ingeniería de Software para garantizar su calidad. Este Sistema de Información responde a la necesidad de automatizar el proceso concentrado en un único lugar la información necesaria para obtener de manera rápida y oportuna información que permita generar reportes y/o estadísticas tanto de individual como grupal y poder al analizarlas implementar estrategias en mejora del proceso de aprendizaje de los alumnos.

Palabras claves: Ingeniería de Software, Sistema de Información, Seguimiento Académico

1 Introducción

Día a día, la labor del docente de educación básica es de gran importancia, esto es debido a que son ellos los encargados de preparar mediante sus enseñanzas a los ciudadanos del mañana. El docente ciclo escolar tras ciclo escolar debe realizar una planeación exhaustiva de su grado asignado con la finalidad de obtener mejores resultados en el proceso enseñanza aprendizaje en sus alumnos. Es por ello, que él planea un conjunto de acciones que le sirven para dar un mejor seguimiento académico a sus alumnos con la finalidad de prevenir, remediar y/o potenciar el aprendizaje, para ello debe de crear estrategias que le permitan corregir problemáticas y potenciar los aprendizajes esperados.

Una de las tareas del docente es realizar un seguimiento académico oportuno de cada uno de sus alumnos y de su grupo; actualmente esté es llevado de forma manual, esto es, al iniciar el ciclo escolar realiza un registro de cada uno de sus alumnos del grupo asignado en papel y va registrando progresivamente sus evaluaciones con la finalidad de dar un seguimiento académico oportuno y así poder informar tanto al padre de familia como a la dirección sobre el aprovechamiento tanto individual como grupal.

Actualmente el docente no cuenta con un Sistema Informático específico [1,2] que le permita realizar un seguimiento académico y análisis de manera oportuna, concentrar información para ser utilizada para generar reportes tanto en papel como en pantalla y

visualizar gráficas; todo esto adecuado a las necesidades de información que se le solicitan por parte de la dirección que lo apoyen a tomar decisiones en mejoras de sus alumnos.

El no contar con un Sistema de Información específico para brindar un mejor seguimiento académico a sus alumnos, ha traído consigo los siguientes problemas: pérdida de tiempo en el análisis del historial de los alumnos, demora en la implementación de estrategias prevenir, remediar y/o potenciar el aprendizaje en sus alumnos, concentración de información en un único lugar respecto al seguimiento académico y evaluación, la obtención de datos numéricos confiables, repetibles y fácilmente verificables; ya que actualmente para realizar un análisis de la información contenida en papel y acumulada cada bimestre del ciclo escolar por cada alumno se lleven al menos 15 días; a su vez faltan reportes, estadísticas y gráficas que permitan al ser analizadas tomar decisiones en mejora del proceso de enseñanza aprendizaje en sus alumnos; es por ello que es necesario automatizar este proceso por medio de un Sistema de Información.

La finalidad del presente artículo es reportar como a través de la aplicación de diversas técnicas, herramientas y modelos que proporciona la Ingeniería del Software, se va a lograr el desarrollar un software de calidad que automatice el seguimiento académico de alumnos de primaria.

2 Marco Teórico – Estado del Arte

2.1 Sistemas de Información – Software Actuales

Actualmente los Sistemas de Información están cambiando la manera de operar las organizaciones. A través de su uso se logran mejoras: automatizan procesos, facilitan la manipulación de información para el proceso de toma de decisiones, facilitan el logro de ventajas competitivas a través de su implantación dentro de las empresas, entre otras. [3]

Según Jane & Kenneth C. Laudon un Sistema de Información se define como: "conjunto de componentes interrelacionados que permiten capturar, procesar, almacenar y distribuir la información para apoyar la toma de decisiones y el control en una institución"[4]. Estos sistemas básicamente realizan los cuatro pasos: entrada, almacenamiento, procesamiento y salida de información.

Hoy en día, no existen software específicos, que le permitan al docente de primaria apoyarse en un Sistema Informático para poder brindar un seguimiento oportuno a cada uno de sus alumnos con la finalidad de estar siempre atento a la mejora en su proceso de aprendizaje y canalizarlo en caso de ser necesario a las instancias correspondientes para su atención; y a su vez el poder entregar la información necesaria a la dirección del plantel para que este emplee estrategias en mejora de los alumnos de su escuela.

2.2 Ingeniería del Software

Hoy en día, para desarrollar un Sistema de Información (software) concretamente se necesita "definir quién está haciendo qué, cuándo hacerlo y cómo alcanzar un cierto objetivo" [5], para ello la Ingeniería de Software, ofrece métodos y técnicas para

desarrollar y mantener software de calidad [1]. Una definición desarrollada por la IEEE es: “La ingeniería de software es la aplicación de un enfoque sistemático, disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento del software” [6].

Con independencia del área de aplicación, tamaño o complejidad del proyecto, cualquier sistema se encontrará al menos en una de las siguientes fases genéricas: definición, desarrollo y mantenimiento [1] Esto es:
• *Definición*.- Se centra sobre el qué; esta fase intenta identificar: que información va a ser procesada, que función y rendimiento desea, que compartimiento, que criterios de validación, entre otros.
• *Desarrollo*.- Se centra en el cómo. Esta fase define cómo son las estructuras de datos, cómo se traduce el diseño en una codificación, cómo son las interfaces, entre otros.
• *Mantenimiento*.- Se centra en el cambio que va asociado a la corrección de errores, a las adaptaciones requeridas a medida que evoluciona el entorno del software, entre otros.

La Ingeniería del Software permite desarrollar software de calidad, para ello actualmente existen dos enfoques: estructurado y orientado a objetos [7]. La Ingeniería de Software orientada a objetos permite crear software modelando el mundo de forma tal que ayuda a entenderlo, controlarlo y gobernarlo de una mejor manera. [1].

3 Desarrollo del Sistema

Para garantizar la calidad del Sistema de Información se normó bajo los estándares, herramientas y técnicas proporcionadas por la Ingeniería del Software, específicamente siguiendo cada una de las fases del ciclo de vida del prototipo evolutivo: recolección y refinamiento de requisitos, diseño, construcción del prototipo, evaluación del prototipo, refinamiento del prototipo, producto de ingeniería [2] y para la descripción de la arquitectura se utilizó el lenguaje unificado de modelado (UML) [9].

El primer paso que se llevó a cabo para el desarrollo del Sistema de Información para apoyar el seguimiento académico de alumnos de primaria fueron reuniones entre la docente de primaria y el grupo de desarrollo para determinar la idea general que permitiera concentrar en un único lugar toda la información necesaria para realizar un seguimiento académico oportuno para prevenir, remediar y/o potenciar el aprendizaje en mejora de los alumnos de primaria, esto se ilustra en la figura 1, los puntos principales fueron:

- Identificación de la información necesaria, tanto de los alumnos como de las evaluaciones del proceso enseñanza aprendizaje para formar parte de la base de datos del sistema.
- Concentración de la información mediante el manejo de una base de datos.
- Interfaz de usuario, que facilite al docente el uso del sistema para realizar las tareas; por ejemplo: captura de alumnos del ciclo escolar, registro de las evaluaciones bimestrales, generación de reportes.

Fig. 1. Idea general del Sistema de apoyo para el seguimiento académico de alumnos de primaria

Teniendo la idea general del Sistema a desarrollar, se inició con la primera fase del ciclo de vida para el desarrollo del sistema, esto es definir claramente los requerimientos, estos son:

- Administrar de manera eficiente cada una de las evaluaciones de los alumnos.
- Concentración en un único lugar de la información necesaria para poder realizar un seguimiento académico oportuno del alumno esto mediante el manejo de una base de datos.
- Automatizar el proceso de seguimiento académico para el docente de primaria.
- Manejar información de alumnos (nombre completo, fecha de nacimiento, sexo, curp, dirección), evaluaciones (proyectos, aspectos, bimestres), ciclos escolares, grado, grupos.
- Registrar evaluaciones bimestrales.
- Llevar registros que le permitan verificar en cualquier momento el progreso del alumno.
- Llevar un historial del alumno en el ciclo escolar para realizar un puntual seguimiento académico para su mejora en el proceso de aprendizaje.
- Generación de reportes individualizados, grupales, por bimestre, por ciclo escolar, entre otros.
- Interfaz de usuario, que facilite al docente el uso del sistema para realizar las tareas; por ejemplo: captura de alumnos del ciclo escolar, registro de las evaluaciones bimestrales, generación de reportes.

Definidos los requerimientos del Sistema, se prosiguió al análisis del sistema, para ello se analizó cada uno de ellos para determinar ¿Qué es lo que se requiere?, por lo cual se apoyó con la realización de un Diagrama de Casos de Uso (ver figura 2), el cual muestra un conjunto de casos de usos, su actor principal y relaciones. El actor principal que interactúa con el Sistema es:

- ***Docente,*** es la persona responsable de la administración del sistema, esto es: registrar a los alumnos, registrar proyectos y sus aspectos a evaluar, administrar bimestres, ciclos escolares, grados y grupos, consultar avances individuales y grupales, generar reportes solicitados tanto para el padre de familia como para la dirección del plantel, a su vez es el encargo de mantener los catálogos actualizados de alumnos, evaluaciones, bimestres, ciclos escolares, seguimiento académicos.

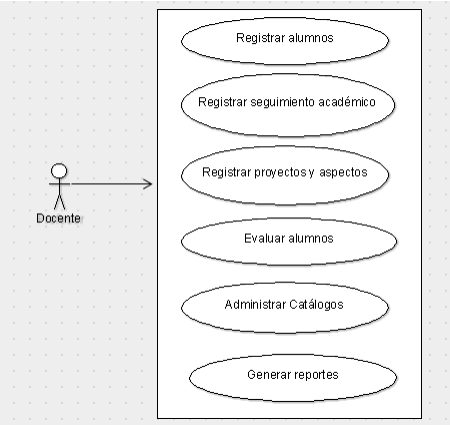


Fig. 2. Diagrama de casos de uso del Sistema de seguimiento académico

Los casos de uso identificados son:

- *Registrar alumnos,* permite registrar a los alumnos de un ciclo escolar en un grado y con un grupo, a su vez brinda la posibilidad si es que le toca el mismo grupo pasarlos al grado siguiente sin necesidad de volver a capturar.
- *Registrar seguimiento académico,* permite registrar como se le ha dado seguimiento a cada uno de los alumnos del grupo en cada bimestre del ciclo escolar.
- *Registrar proyecto y aspectos,* permite registrar cada uno de los proyectos y aspectos según el grado y bimestre a dar seguimiento académico durante el ciclo escolar.
- *Catálogos*, permite administrar todo lo relacionado con alumnos, proyectos, aspectos, evaluaciones, bimestres, ciclos escolares, entre otros.
- *Generar reportes,* permite generar diversos reportes del sistema.

La siguiente fase fue la de diseño, en ella se diseñaron los diagramas de secuencia del comportamiento correspondiente, y las interfaces de usuario, las cuales tomaron en cuenta a las personas que utilizarían el sistema, las tareas a realizar, así como el entorno donde se manejaría; a cada interface se le midió la complejidad, además se aseguró que fueran: consistentes, fáciles de aprender, flexibles, robustas, esto de acuerdo a los principios de la usabilidad.
El sistema se desarrolló utilizando el lenguaje de programación Java y el manejador de base de datos MySQL, respaldados por sus características de funcionamiento, popularidad y exención de inversión financiera.

Las pruebas correspondientes se realizaron con la finalidad de encontrar errores antes de que entrara en operación el Sistema; para la realización de las pruebas de caja blanca se diseñaron un conjunto de casos de pruebas para asegurar que todas las sentencias y condiciones se ejecutarán al menos una vez; y las pruebas de caja negra permitieron verificar el comportamiento adecuado de los módulos del Sistema, en respuesta a las acciones o entradas originadas por el usuario. Terminada la fase de prueba se procedió a dar la capacitación correspondiente para su correcto funcionamiento. Estas fases se realizaron de forma iterativa e incremental; es así como el desarrollo de éste Sistema de apoyo para el seguimiento académico de alumnos de primaria siguió la metodología de la Ingeniería del Software para asegurar su calidad.

4 Breve recorrido por el sistema

A continuación se realiza un breve recorrido a través de algunas de las interfaces del Sistema de Información: Pantalla Principal, en ella se puede apreciar las opciones del Sistema de Información para apoyar el seguimiento académico de alumnos de primaria, estos son: Administración de Ciclos Escolares, Administrar Grupos, Administrar Materias, Administrar Aspectos, Administrar Proyectos, Administrar Alumnos, Administrar Bimestres (para la administración de cada catálogo se tienen las opciones de registrar, actualizar, eliminar y consultar), Generar Reportes y Evaluar Alumnos (Ver figura 3).

En la figura 4 correspondiente a la asignación de alumnos a un grupo y a un grado escolar y a un ciclo escolar; en esta opción es importante mencionar que también se puede seleccionar un grupo ya registrado anteriormente y simplemente asignarlo a otro grado y/o grupo. Para realizar una evaluación de un alumno (ver figura 5), es necesario seleccionar un grupo-grado y ciclo, el bimestre a evaluar y el proyecto de la asignatura correspondiente; posteriormente se selecciona al alumno a evaluar y se evalúa cada aspecto del proyecto. Una vez evaluado el alumno, este va pasando a una lista de alumnos evaluados para tener un control en la misma interfaz. (ver figura 6)

Para agregar aspectos al proyecto previamente definidos, únicamente se selecciona el aspecto y se va agrega como se necesite, (ver figura 7) de igual manera en bimestre se asignar proyectos a desarrollar por materia y por grupo (ver figura 8).

Por otro lado, la figura 9, ilustra dos interfaces, la primera correspondiente a las opciones de reportes que se pueden generar con el Sistema, y la segunda el reporte del formato de seguimiento individual de cada uno de los alumnos, el cual es presentado cada bimestre al padre de familia, incluso a la dirección del plantel si es lo solicitara; estos reportes también tienen la posibilidad de generarse de acuerdo a una serie de criterios establecidos por el docente, por ejemplo: niños con promedio menor a seis, niños sin evaluar, comparativos entre niños y niñas, por mencionar algunos.

A lo largo de esta sección se presentaron algunas interfaces de las tareas principales del Sistema de Información, observando cómo estas interfaces a través de su simplicidad y facilidad de uso permiten que la realización de las tareas para dar un buen seguimiento académico a sus alumnos sea lo más sencillo y práctico.

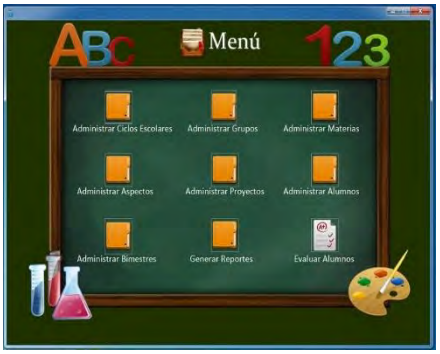


Fig. 3. Pantalla Principal

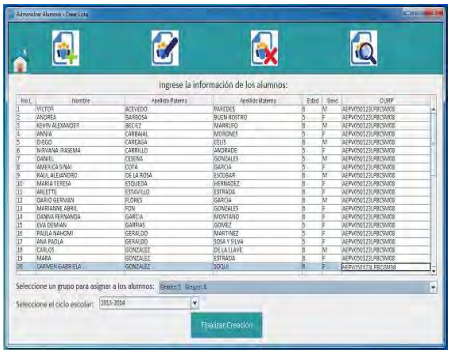


Fig. 4. Asignación de alumnos a un grupo de un grado en un ciclo escolar

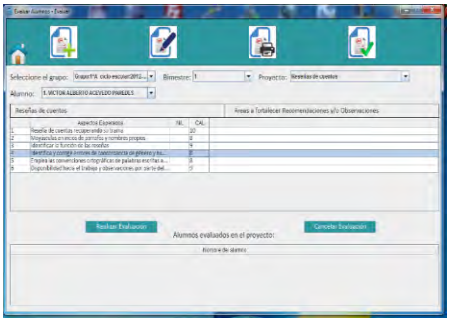


Fig.5. Evaluación de un alumno

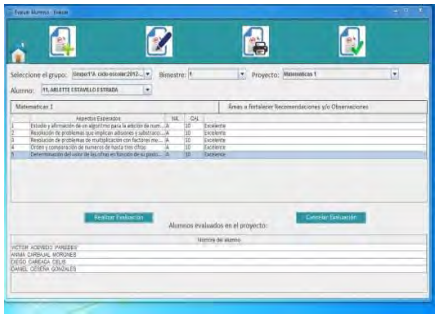


Fig.6. Evaluación de alumnos

Fig. 7. Interfaz para la agregación de aspectos al proyecto

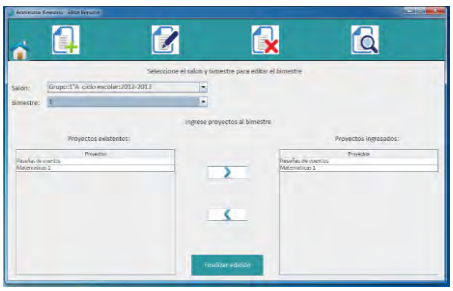
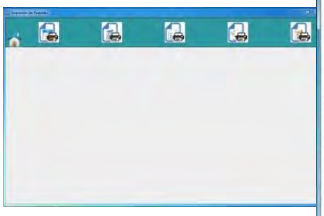


Fig. 8. Interfaz para la agregación de proyectos al bimestre

Fig. 9. Formato de seguimiento académico



5 Resultados y conclusiones

El principal resultado es que el docente cuente con un Sistema de Información que le sirva de apoyo para brindar de manera eficiente y oportuna un seguimiento académico oportuno, ya sea de manera individual o grupal. A su vez, el Sistema de Información concentra en un único lugar la información, lo cual agiliza la obtención de reportes y

estadísticas de una manera rápida, fácil y oportuna que le han permito tomar decisiones e implementar estrategias en mejora del aprendizaje y rendimiento escolar de sus alumnos.

Por otro lado, los padres de familia recibirán una mayor optimización en su atención, con más eficiencia y rapidez; por parte del docente esto debido a que en cualquier momento se le podrá proporcionar información académica de su hijo. Del lado de la dirección, se podrán tomar decisiones rutinarias en menor tiempo y con mayor confianza sobre la información entregada por parte de los docentes del plantel.

Bibliografía

1.Pressman R.S. Ingeniería del Software, un enfoque práctico. McGraw Hill. 6 Ed. (2002)
2.Kendall y Kendall. Análisis y diseño de Sistemas. Prentice Hall. Sexta edición (2005)
3. Cohen D., Asín E. Sistemas de Información para los Negocios, Mc Graw Hill. 2Ed. México.(2000)
4. Laudon K Laudon J.Management Information Systems: managing the digital firm. Pearson, Prentice Hall. Lebanon, Indiana USA (2007).
5. I. Jacobson, "Applying UML in The Unified Process" disponible en http://www.rational.com/uml/UMLconf.zip
6. IEEE. IEEE Standards Collection: Software Engineering 610.12-1990. IEEE 1993
7. Craig Larman. UML y Patrones. Introducción al análisis y diseño orientado a objetos. Ed. Prentice Hall. 1ra. Edición. México. (1999)
8. J.A. Senn. Análisis y diseño de sistemas de información. McGraw Hill. Segunda Edición. México. (2003)
9. C. Larman, UML y Patrones. Introducción al análisis y diseño orientado a objetos. Ed. Prentice Hall, Primera Ed., México, 2002.

Desarrollo de una aplicación para la administración de proyectos de investigación de la División Académica de Ingeniería y Arquitectura.

Área de Conocimiento: Sistemas de Información.

Manuel Villanueva Reyna, Martha Patricia Silva Payró, Marcela Cristal Fernández Alcudia. [1]
[1] División Académica de Informática y Sistemas, Universidad Juárez Autónoma de Tabasco.

Carr. Cunduacán-Jalpa km. 1, Cunduacán, Tabasco, México
manuel.villanueva@ujat.mx, martha.silva@ujat.mx,
marce_cris_14@hotmail.com

Resumen. Esta investigación tuvo como principal objetivo desarrollar un Sistema de Información que permitió automatizar el proceso de registro, revisión y seguimiento de proyectos de investigación de la DAIA. Para esto se desarrolló una página web mediante la cual se interactúa con el sistema, permitiendo realizar las tres acciones siguientes: a) Los profesores investigadores podrán registrar, modificar o actualizar los proyectos de investigación que estén proponiendo realizar, con ó sin financiamiento, asimismo podrán registrar los informes parciales ó finales. b) Los integrantes del comité de investigación divisional podrán revisar y evaluar el contenido de los proyectos de investigación para que los profesores reciban las observaciones oportunamente y puedan realizar las correcciones pertinentes en tiempo y forma. c) El Coordinador responsable de dar seguimiento a la presentación y aprobación de proyectos de investigación podrá asignar los revisores, revisar cada uno de ellos y validar los aprobados por el comité de investigación.

Palabras clave: Aplicación de Software, Sistema de Información, Administración de Proyectos, Proyectos de Investigación.

1. Introducción

1.1 Antecedentes

La Investigación en la División Académica de Ingeniería y Arquitectura de la Universidad Juárez Autónoma de Tabasco, inicia en forma sistemática en 1979 con la creación de la Maestría en Hidráulica, aunque, en 1978 se realizaron varias investigaciones en Ingeniería Hidráulica, destacando el proyecto: "Cause de Alivio Samaria Golfo de México" en el que participaron varias dependencias federales y estatales en área hidráulica, en coordinación con la UJAT. En 1985, con la transformación del Instituto de Ingeniería como División Académica de Ingeniería y Tecnología (DAIT), las actividades de investigación empezaron a ser coordinadas por el Centro de

Investigación. Se definen inicialmente cuatro áreas de investigación las cuales fueron: Energía, Materiales, Sistema e Impacto Ambiental. En el año de 1994 se construyen las instalaciones del nuevo Centro de Investigación. El 5 de julio de 1995, el H. Consejo Universitario aprueba el cambio de nominación de la División Académica de Ingeniería y Tecnología (DAIT) por la de División Académica de Ingeniería y Arquitectura (DAIA) (Universidad Juárez Autónoma de Tabasco, 1996).

Actualmente cuenta con cinco Programas Educativos donde los Profesores Investigadores registran proyectos de investigación internos, se reportan avances o termino de proyectos, también se reportan proyectos con financiamiento externo, como es el caso de PROMEP (Programa para el Mejoramiento del Profesorado).

La Coordinación de Investigación y Posgrado en conjunto con el Comité de Investigación se encargan de dar seguimiento para ambos tipos de proyectos, cabe mencionar que la convocatoria de proyectos PROMEP es una vez por año y las de proyectos sin financiamiento es dos veces por año.

Los proyectos de investigación son parte de una actividad complementaria del Profesor Investigador, ya que estos fungen como responsables, los cuales tienen que combinar sus actividades de la docencia con la participación en actividades de investigación y así llevar a cabo la vinculación de proyectos de investigación con el sector productivo público y privado

1.2 Planteamiento del problema

La Coordinación de Investigación y Posgrado, carece de un Sistema de Información que les permita controlar y automatizar cada uno de los procesos que implica el seguimiento de los proyectos de investigación y así reducir el tiempo de búsqueda y administrar de manera eficiente cada uno de los proyectos registrados que se estén llevando a cabo en la División en un determinado periodo.

Actualmente el seguimiento de los proyectos de investigación se realiza de forma manual y esto hace que el proceso del mismo se haga laborioso para la Coordinación de Investigación y Posgrado, ya que cuando se requiere información, por la gran cantidad de proyectos registrados se pierde tiempo en la búsqueda del mismo, porque toda la documentación de los proyectos se encuentra almacenada en carpetas, generando un concentrado con los datos relevantes en una hoja de cálculo de Excel, además, se tiene acceso fácilmente a las carpetas y, por lo tanto, aunque no ha sucedido se tiene que evitar pérdida de la misma.

El objetivo principal de este proyecto fue el de desarrollar un Sistema de información que permitiera automatizar los diversos proyectos de investigación de la División Académica de Ingeniería y Arquitectura, integrando en una base de datos toda la información.

Entre los objetivos específicos se encuentran el de analizar y diseñar cada uno de los procesos que permitan controlar y administrar los proyectos de investigación de la DAIA, el de administrar la base de datos para la gestión y automatización de procesos de cada uno de los proyectos de investigación y el de diseñar una página Web dinámica para que los profesores investigadores y los miembros del comité de Investigación pudieran alimentar el sistema y poder dar seguimiento a los proyectos de Investigación.

2. Estado del Arte

Un Sistema de Información es un conjunto de componentes que interactúan entre sí, orientado a la recolección, almacenamiento, procesamiento y recuperación de información. El origen de los Sistemas de Información se puede rastrear tan atrás como los censos (en donde se recopila, almacena, procesa y recupera información que posteriormente se usa para la toma de decisiones) que realizaban los babilonios y egipcios 4000 años antes de Cristo. Actualmente, se piensa en Sistemas de Información con sustento en las Tecnologías de la Información y las Comunicaciones.

En cuanto a trabajos similares o proyectos que nos permitieron contar con antecedentes y tener puntos de comparación en la elaboración de nuestro proyecto mencionamos algunos de los trabajos consultados.

En [2], los autores elaboraron una tesis llamada "Desarrollo de un sistema de información para la automatización de los proyectos de investigación (Caso CIPDAIS)" realizada en la DAIS, con la finalidad de darle seguimiento a las diversas actividades: registro de proyectos institucionales, seguimiento de cuerpos académicos y alumnos de posgrado.

En [3], el autor realizó una tesis llamada "Análisis y estudio comparativo de los programas de computación para la administración de proyecto" con el objetivo de dar a conocer los tipos de administración de proyecto para elegir un buen sistema que cubra sus necesidades. Fue realizada en la Universidad Autónoma del Estado de Hidalgo.

En [4], el autor realizó una tesis llamada "Administración de proyectos orientada a la implementación de sistemas de información" con el objetivo de plantear el mecanismo a seguir para que cualquier infraestructura proporcione la información necesaria, veraz e inmediata para que el gerente tome las decisiones oportunas y adecuadas. Realizada en la Universidad Francisco Marroquín.

En [5], el autor realizó una tesis llamada "Implementación de una herramienta de gestión de proyectos en el área de sistemas e informática de una empresa de telecomunicaciones" con el objetivo de llevar con éxito la implementación de la herramienta de gestión de proyectos: Microsoft Project Server versión 2007. Realizada en la Pontificia Universidad Católica del Perú.

3. Metodología

Para el análisis y recolección de datos en esta investigación se utilizó el enfoque cuantitativo. El enfoque cuantitativo se basa en las técnicas de recolección y el análisis de datos, se sirve de números y métodos estadísticos, partiendo así de casos concretos para llegar a una descripción general o comprobar hipótesis establecidas previamente, es decir busca la mayor claridad entre los elementos que conforman el problema, buscando que tenga definición, los limita y con exactitud saber dónde inicia el problema. [1].

En la metodología de la investigación se utilizó el enfoque cualitativo ya que este nos permite elaborar observaciones no estructuradas, entrevistas abiertas, revisión de documentos, etc., tanto al Comité de Investigación como a la Jefatura de Investigación.

En [1], los autores definen el enfoque cualitativo como "la recolección de datos sin medición numérica para descubrir o afinar preguntas de investigación en el proceso de investigación". Para la indagación se utilizaron las fuentes de datos como entrevistas semiestructuradas y observaciones. La observación para estar atenta a los detalles, sucesos, e interacciones durante la entrevista. Y la entrevista semiestructurada para interactuar personalmente con el (la) entrevistado (a). se utilizaron ambas fuentes de datos para recolectar todo tipo de información, útiles para la construcción del sistema de

información. El universo de estudio fueron los profesores-integrantes del Comité de Investigación y de la Jefatura de Investigación.

En cuanto a la metodología del sistema se eligió trabajar con la Programación Orientada a Objetos, la metodología de Programación Extrema y el Modelado Incremental, ya que estos tres elementos trabajan conjuntamente y son fundamentales para el análisis y diseño de un SI. Como utilizamos una metodología iterativa lo más conveniente fue utilizar un modelo iterativo para el ciclo de vida, y en este caso fue el Modelo Incremental.

De acuerdo con la metodología utilizada se siguieron los siguientes pasos: 1) una fase de inicio que abarcó la comunicación con el cliente y las actividades de planeación, a través de los tres tipos de usuarios, en el cual se realizó el refinamiento de casos de uso como un modelo primario; 2) una fase elaboración que abarcó la comunicación con el cliente y las actividades de modelado con un enfoque en la creación de modelos de análisis y diseño, con énfasis en las definiciones de clase y representaciones arquitectónicas; 3) una fase de construcción en la que se realizaron actividades de refinamiento que permitió obtener un modelo de diseño en componentes de software implementados; 4) una fase de transición en la que el software desarrollado se presentó a los usuarios y se realizaron pruebas de funcionalidad, de confiabilidad y de facilidad de uso para asegurar su aceptación; y 5) una fase de producción en la cual se realizó la implementación, como se muestra en la fig. 1.

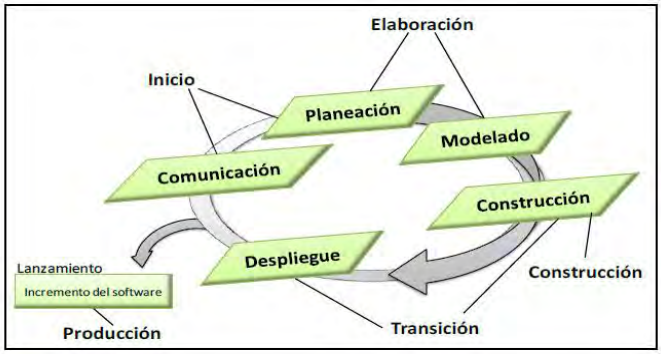


Fig. 1.- Proceso Unificado

En [6], el autor describe como se muestra en la Fig. 2, el modelo incremental aplica secuencias lineales de manera escalonada conforme avanza el tiempo en el calendario. Cada secuencia lineal produce "Incrementos" del software. A menudo, al utilizar un modelo incremental el primer incremento es el producto esencial. Es decir, se incorporan los requisitos básicos. El producto esencial queda en manos del cliente (o se somete a una evaluación detallada).

Fig. 2.- Modelo Incremental

Como resultado de la evaluación se desarrolla un plan para el incremento siguiente. El plan afronta la modificación del producto esencial con el fin de satisfacer de mejor manera las necesidades del cliente y la entrega de características y funcionalidades adicionales. Este proceso se repite después de la entrega de cada incremento mientras no se haya elaborado el producto completo.

Este modelo para la elaboración del ciclo de vida del sistema, es uno de los modelos recientes que pertenecen al desarrollo ágil, el cual nos dio la facilidad de entregar el sistema como su nombre lo dice en "incrementos" y esto ayudó para que el sistema en cada incremento mejorara.

La Programación Extrema (PE) es un proceso del Desarrollo Ágil, este último combina una filosofía y un conjunto de directrices de desarrollo. La filosofía busca la satisfacción del cliente y la entrega temprana del software incremental. Por otra parte, las directrices de desarrollo resaltan la entrega sobre el análisis y el diseño, y la comunicación activa y continua entre los desarrolladores y los clientes.

La PE utiliza un enfoque orientado a objetos como su paradigma de desarrollo preferido. La PE abarca un conjunto de reglas y prácticas que ocurren en el contexto de cuatro actividades del marco de trabajo: planeación, diseño, codificación y pruebas, como se aprecia en la Fig. 2 en la que se ilustra el proceso de la PE y se observan algunas de las ideas y tareas clave asociadas con cada actividad del marco de trabajo.

4. Resultados

La realización de este proyecto permitió llegar a diversos resultados, los cuales se mencionan a continuación, asimismo se muestran algunas pantallas de las principales interfaces desarrolladas. Asimismo para cada uno de los usuarios involucrados en la operación del sistema; profesores investigadores de la División Académica, Integrantes del Comité de Investigación así como para el Coordinador responsable de dar seguimiento a cada uno de los proyectos de investigación se presentan los resultados más importantes.

Para el Profesor.

Una vez que el Profesor realizó un Registro de proyecto, en el apartado Proyectos Propuestos se observa que el proyecto queda almacenado correctamente en la base de datos. Mostrando el título del proyecto y la fecha de registro. Ver Fig. 3.

Fig.3 Proyectos propuestos

Para el Comité de Investigación.

Cuando a un integrante del Comité se le asigna un proyecto para evaluarlo, en el apartado Evaluación de Protocolo se le mostrará la siguiente pantalla, indicándole que tiene un nuevo proyecto para evaluar. Ver Fig.4.

Fig.4 Evaluación de protocolo

Para el Coordinador Responsable.

Si el Coordinador desea ver los Proyectos en Evaluación se muestra la siguiente pantalla. Contiene los datos más relevantes del proyecto, el comité y el dictamen que se obtuvo, al seleccionar "Actualizar estado del proyecto" al profesor se le informa sobre la evaluación de su proyecto. Ver **Fig**. 5.

Fig.5 Actualizar estado del proyecto

5. Conclusiones y Trabajos futuros.

Con la realización del proyecto de administración de proyectos de investigación de la División Académica de Ingeniería y Arquitectura se alcanzaron diversos objetivos.

Se logró concluir satisfactoriamente un Sistema de Información que permite automatizar los diversos proyectos de investigación de la división en un área que maneja información importante para el control y seguimiento de los proyectos de Investigación. Esto permitió integrar en una Base de Datos la información correspondiente al manejo y control de los diversos proyectos de investigación.

Asimismo, se analizaron y se diseñaron cada uno de los procesos que permite controlar y administrar los proyectos de investigación.

Se implementó la administración de una base de datos para la gestión, automatización y seguimiento en el desarrollo de los proyectos de investigación.

Se diseñó una página web dinámica para que los profesores investigadores, el Comité de Investigación y el coordinador responsable contaran con una herramienta que les permitiera monitorear los proyectos de investigación y actualizar la base de datos correspondiente.

Con la ayuda del servidor Appserv se logró desarrollar el Sistema, ya que se utilizó el lenguaje de programación PHP y el manejador de bases de datos MySQL, los cuales facilitaron la realización de la programación.

Con la metodología del Proceso Unificado se analizó y diseñó el Sistema conforme a sus cinco fases realizadas y con el Lenguaje de Modelado Unificado (UML) se diseñó cada uno de los procesos del Sistema.

Asimismo, El resultado de las pruebas de usabilidad realizadas utilizando la norma ISO 9126, los profesores determinaron que el funcionamiento general del sistema es bueno.

Adicionalmente y dada la posibilidad de continuar con este proyecto consideramos posible realizar en un futuro las siguientes actividades con la intención de generalizar la aplicación del proyecto en las diversas divisiones académica de la UJAT, así como la posibilidad de implementarse en otras instituciones del mismo ramo.

Alojar el sistema en el servidor de la UJAT brindando la posibilidad de acceso

desde cualquier División Académica.

Agregar a la base de datos registro de Profesores Investigadores de otras Divisiones Académicas.

Integrar más actividades con el fin de automatizar las áreas de la Coordinación de Investigación y Posgrado, ya que esto permitiría ampliar la mejora de los procesos mediante el uso de la tecnología.

Finalmente, este sistema de información podrá implantarse en diversas instituciones que requieren de la Administración y Seguimiento de proyectos de Investigación y/o Vinculación.

6. Referencias

[1] R. Hernández, C. Fernández & P. Baptista. Metodología de la Investigación. McGraw-Hill, México. 2013.

[2] A. Osorio y J. Noriega. Desarrollo de un sistema de información para la automatización de los proyectos de investigación: Caso CIP-DAIS, Universidad Juárez Autónoma de Tabasco, Cunduacán Tabasco, México. 2008.

[3] F. Montiel, Análisis y estudio comparativo de los programas de computación para la administración de proyectos, Universidad Autónoma del Estado de Hidalgo. Pachuca Hidalgo, México. 2006.

[4] E. Godínez, Administración de proyectos orientada a la implementación de sistemas de información, Universidad Francisco Marroquín. Guatemala, Guatemala. 2003.

[5] J. Toledo, Implementación de una herramienta de gestión de proyectos en el área de sistemas e informática de una empresa de telecomunicaciones", Pontificia Universidad Católica del Perú. 2012.

[6] R. S. Pressman, (2005). Ingeniería del software. Un enfoque práctico. (6ta ed). Mc Graw Hill.

[7] L. J. Aguilar, Programación orientada a objetos (2da ed.). Mc Graw Hill. 2005.

[8] K. E. Kendall & J. E. Kendall. Análisis y Diseño de Sistemas (8va ed.). Prentice Hall, 2014.

[9] M. Fowler & K. Scott. UML gota a gota. Addison Wesley.2009.

[10] J. F. Gil, Creación de sitios web con PHP5. Mc Graw Hill, 2007.

[11] C. Pérez, Dreamweaver 8. Desarrollo de páginas web dinámicas con PHP y MySQL. Alfaomega. 2008.

[12] E. Oz, Administración de los sistemas de información (5ta ed.). Thomson. 2008.

[13] G. Booch. Análisis y diseño orientado a objetos con aplicaciones ed. Pearson.

Control de trazabilidad de requerimientos en ambientes ágiles de desarrollo de software

José Manuel Hernández Reyes [1], Carlos Enríquez Ramírez [2]
Universidad Politécnica de Tulancingo, Calle Ingenierías #100, Col. Huapalcalco, Tulancingo, Hidalgo, C.P. 43629. México.
{jose.hernandez[1], carlos.enriquez[2]}@upt.edu.mx

Resumen. En el ámbito de los sistemas de software, los requerimientos son la parte primordial del desarrollo por lo que se debe cuidar el manejo del mismo. Esta investigación genera un proceso de gestión de requerimientos a través de la trazabilidad, apoyado en un sistema *TraceAM* para ambientes ágiles de desarrollo de software, que permite un monitoreo y la entrega en tiempo y forma según la especificación del usuario. Las actividades que se realizan son: modelar un proceso de trazabilidad de requerimientos en ambientes ágiles de desarrollo de software; generar un sistema de información que permita dar el seguimiento y monitorear los requerimientos en un proyecto de acuerdo al proceso generado; probar la efectividad de la herramienta en escenarios reales.

Palabras clave: Ingeniería de Requerimientos (IR), Trazabilidad, *Stakeholders*, Metodologías Ágiles.

1 Introducción

Los sistemas de software siempre han tenido la característica de contener muchos recursos desde su creación incluyendo una gran cantidad de productos como: documentos de requerimientos y diseño, código fuente, casos de prueba. Estos son desarrollados y actualizados durante largos períodos de tiempo por diferentes personas. El establecimiento y mantenimiento de dichas conexiones entre los productos de software es reconocido como un problema difícil de administrar. Es por ello que la presente investigación implementa un proceso de trazabilidad de requerimientos enfocadas a organizaciones que hacen uso de metodologías ágiles para el desarrollo de software, y una solución adoptada por esta investigación para extraer y analizar los datos es el uso de técnicas de Recuperación de Información (RI). La RI permite generar un índice (id) que identifica a cada elemento y permite su relación con otros. Una clara ventaja de usar esta técnica es que no se basa en un vocabulario o la gramática predefinida, esto permite que pueda aplicarse sin necesidad de grandes cantidades de pre-procesamiento.

Es importante mencionar que en el ámbito de los sistema de software, los requerimientos son la parte primordial del desarrollo por lo tanto se debe cuidar el manejo de los mismos [1], es por ello que el enfoque de este trabajo se realiza en la gestión de los requisitos del software a través de la trazabilidad ya que está es vista como una medida de la calidad de los sistemas.

2 Estado del Arte

La trazabilidad permite la recuperación eficiente de los productos relacionados, lo cual es útil en una variedad de tareas de ingeniería de software, tales como el mantenimiento del software, la comprensión del sistema y depuración del mismo [2]. Mientras tanto, los ambientes de desarrollo de software son cada vez más distribuidos, descentralizado y depende de terceros, motivando la necesidad de técnicas trazabilidad eficaces. Los *stakeholders* (involucrados) tienen interés en el tema, además requieren de soporte personalizado, como el nivel de especificación de la captura del trazado y los tipos de productos a rastrear. Los beneficios más importantes proporcionados por la trazabilidad pueden ser realizados durante la fase de Verificación y Validación (V&V) [3]. La implementación correcta de la trazabilidad puede utilizarse para demostrar que un sistema de software cumple con sus requisitos y que se han aplicado correctamente. Por ejemplo si un requisito puede ser rastreado hasta el código, se realiza la validación de su implementación, adhiriéndole un caso de prueba y así se demuestra que ha sido verificado. Esta actividad apoya la gestión del proyecto, asegurando que el software se ajuste a las especificaciones de los clientes [4].

2.1 La Trazabilidad en las Metodologías Ágiles

Las Metodologías Ágiles (MA) o "ligeras" constituyen un nuevo enfoque en el desarrollo de software, aceptado por los desarrolladores a diferencia de la adopción de las metodologías convencionales, debido a la simplicidad de sus reglas y prácticas, además de su orientación a equipos de desarrollo pequeños, la flexibilidad ante los cambios y su ideología de colaboración [5]. Representando un conjunto de procesos de desarrollo donde los requisitos y la solución entregada evolucionan gradualmente, a través de una serie de iteraciones cortas. Estos proyectos se caracterizan por un énfasis sobre las interacciones humanas y colaboraciones, aligerando el desarrollo de procesos, entregas frecuentes y documentación mínima. El punto de partida de esta metodología es el Manifiesto Ágil, documento que resume la filosofía donde se valora lo siguiente:

- *Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas.*
- *Desarrollar software que funcione más que conseguir una buena documentación.*
- *La colaboración con el cliente más que la negociación de un contrato.*

Los modelos de trazabilidad son incompatibles con las prácticas de desarrollo ágil, porque la IR hace un marcado énfasis en la documentación. Por lo tanto las metodologías ágiles buscan un punto medio entre ningún proceso y demasiados procesos, proporcionando sólo lo suficiente para que el esfuerzo sea redituable. Actualmente se han propuesto varios enfoques para apoyar la creación de trazabilidad, estos métodos se pueden clasificar como manuales y semi-automáticos.

En el artículo sobre "*Lean Traceability*" Brad Appleton describe varias técnicas adicionales para el trazado ligero. Una de estas técnicas es utilizar las herramientas de gestión de configuración existentes con el fin de capturar los enlaces de trazabilidad [1]. En febrero del 2009 *Marcus Jacobsson* de la Facultad de Ingeniería, del *Department of Computer Science Sweden* define una serie de prácticas que se deben considerar para la implementación de la trazabilidad en desarrollo de software ágil. Mientras que algunas de las prácticas son baratas o gratuitas existen otras que son caras y probablemente no

tiene retorno positivo de la inversión. Estas prácticas se podrían utilizar para agregar la trazabilidad de los métodos ágiles [6]. A continuación se discuten las más importantes:

- *Stakeholders a Requerimientos* – consiste en realizar un seguimiento de quien contribuyo en cada requerimiento (o historia de usuario). Con esto es posible identificar la fuente de cada requerimiento y llevar a cabo un ciclo de retroalimentación para notificar a los *stakeholders* el progreso de desarrollo.
- *Requerimientos (historias de usuario) a Versiones* – permite rastrear exactamente qué requisitos se han aplicado en una revisión específica. Esto se puede lograr con el uso de una herramienta para el control del código fuente como *Perforce*, *SVN*, o *Clear Case*[21] uniendo los identificadores de cada historia de usuario con los cambios confirmados.
- *Relación entre Requerimientos* – permite rastrear las dependencias entre las historias de usuario. Aunque esto podría potencialmente crear una sobrecarga significativa, es relativamente fácil insertar dependencias identificadas en las historias de usuario.

La adición de la trazabilidad en el camino equivocado resulta en un costo adicional en el proyecto, mientras que hacerlo de la manera correcta resultará en un beneficio para todos los involucrados en el proyecto. Entonces hacer bien el rastreo tendrá como resultado un producto superior, así como menos tiempo trabajando en el proyecto (a la hora de pasar de la recolección de requisitos hasta que el producto es finalizado). La información almacenada durante el desarrollo podría ahorrar el tiempo para muchos miembros del equipo, evaluadores, directivos y equipos de mantenimiento.

3 Metodología

El sistema propuesto se define como *TraceAM* (*Agile Methodology*), es una aplicación web que permite el control de la trazabilidad de los requerimientos en ambientes ágiles de desarrollo de software, basado en el sistema de planificación *Kanban*[22]. El método *Kanban* en *TraceAM* permite la visualización de los requerimientos mediante una lista, cambiante de estados según el avance del desarrollo.

Las características con las que cuenta son:

- Monitoreo del grado de avance y seguimiento de los requerimientos en la implementación.
- Lectura clara de los requerimientos con sus observaciones y retroalimentación.
- Demostrar que un sistema cumple con sus requerimientos correctamente.
- Revisión, validación y verificación de requerimientos.
- Información histórica para el mantenimiento del producto de software debido al rastreo de requerimientos desde su origen hasta su desarrollo.

3.1 Estrategia de *TreaceAM*

[21] *Sistemas de control de versiones para equipos de desarrollo de software*

[22] *Kanban es un sistema de planificación just in time desarrollado por Toyota para gestionar el flujo de trabajo.*

La estrategia para la implementación de la trazabilidad es mediante la técnica de *Esquemas o Índices de Referencias Indexadas*, que consiste en colocar un identificador *id* al proyecto en desarrollo, iteración, requerimiento y a los productos generados de dicho requerimiento como se muestra en la Fig. 1, a fin de generar una cadena de trazado que permita unir y seguir a cada requerimiento lo largo del desarrollo del sistema de software solicitado.

Fig. 1. Generación de indexación de referencias

Este *id* permitirá identificar al elemento y relacionarlo con otros a través de su indexación, permitiendo la generación de un trazado entre estos. Un proyecto estará constituido por un número de iteraciones y este a su vez por un conjunto de requerimientos pertenecientes a dicha iteración, posteriormente un requerimiento estará constituido por los productos generados en el desarrollo (diseño, codificación de módulos y casos de prueba), como se representa en la Tabla 1.

Esquema de trazado de elementos	
1. PR01	⇨ Proyecto
1.1. IT01	⇨Iteración
1.1.1. RQ001	⇨Requerimiento
1.1.1.1. Ref_Diseño	⇨Productos Generados
1.1.1.2. Ref_Codificación	
1.1.1.3. Ref_Casos de Prueba	
1.1.2. RQ002**,** RQ003, …***n***	
1.2. IT02**,** T03, …***n***	
2. PR02**,** PR03, …***n***	

Tabla 11. Esquema de trazado de elementos.

La obtención de los índices de referencias de cada elemento hace uso del siguiente esquema para la recuperación de los enlaces de trazabilidad la cual consta de cuatro pasos como se muestra en la Fig. 2.

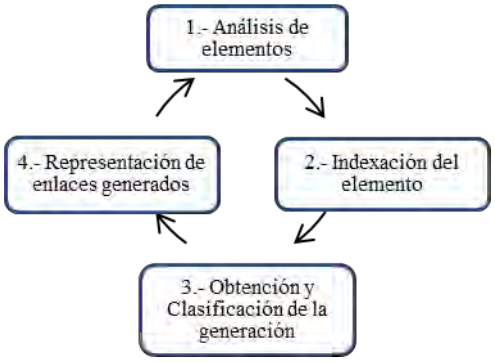


Fig. 2. Proceso para la recuperación de enlaces de Trazabilidad

En el primer paso, los productos de software son extraídos a un nivel de especificación (por ejemplo, proyecto, iteración, requerimiento), y representados a través de un identificador. En el segundo paso, la trazabilidad utiliza un método que permite identificar y relacionar con los diversos productos de software. El tercer paso consiste en examinar todos los elementos relacionados a un determinado nivel de presentación para determinar su propósito. El cuarto paso consiste en representar los vínculos generados para su seguimiento y monitoreo.

3.2 Proceso *TraceAM*

Al iniciar el proceso el usuario debe registrar el proyecto que se va a gestionar, posteriormente se determina la iteración y se documenta los requisitos del cliente. Una vez finalizada este proceso cada requerimiento genera una cadena de trazado que controla el recorrido del requisito durante su proceso de desarrollo como se visualiza en la Fig. 3 y representado en el sistema en la Fig. 4. La *Cadena de Trazabilidad*: representa un conjunto de enlaces que permite identificar las relaciones existentes de cada uno de los elementos trazados a un nivel de presentación. Los posibles estados de la cadena de trazado pueden ser:

- *Detenido*, aplica cuando el requerimiento es cancelado y esta relación da fin a su desarrollo.
- *Remplazado por*, aplica cuando el requerimiento cambia totalmente y al utilizar la relación está apunta a una nueva cadena de trazado con un nuevo requerimiento que sustituye al anterior.
- *Extiende a*, aplica cuando se tiene agregación de funcionalidad a un requerimiento existente por lo que genera una nueva cadena de trazado con un nuevo requisito que da continuidad al anterior que relaciona.

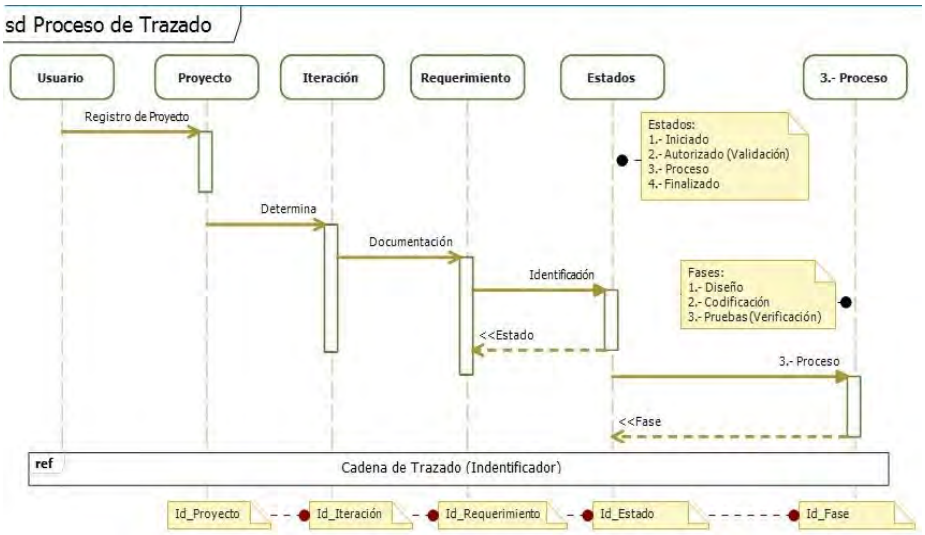


Fig. 3. Proceso de Trazabilidad de Requerimientos

Fig. 4. Sistema de *TraceAM*

4 Resultados

En este trabajo investigación se realizó tanto el proceso como el producto, para ello fue seleccionada una empresa desarrolladora de software dónde se presentó el uso y propósito *TraceAM*, posteriormente los participantes implementaron la herramienta en el

seguimiento de los requerimientos de un proyecto real. Para poder evaluar *TraceAM*, se tomaron en cuenta algunas de las características de calidad del estándar ISO/IEC 25010: *System and software product Quality Requirements and Evaluation* (*SQuaRE*) [7]. Definiendo la calidad como "el grado en que un producto de software satisface las necesidades expresas e implícitas cuando se usa bajo condiciones específicas". Estas necesidades están representadas por el modelo de calidad que categoriza las características y subdivide en dos vistas: calidad en uso y del producto. La primera, identifica las características de calidad esperadas por un usuario del sistema y la segunda, las que debe cuidar el constructor del sistema.

La evaluación fue aplicada en *Hidalgo Software Services S.A. de C.V.* en el desarrollo del proyecto de software: **Transporte** (Diagnóstico de Energéticos en Empresas de Autopartes de Carga) V3.0 que consiste en la "Administración de sub-empresas (flotas) y diagnósticos de la infraestructura interna, basado en metodologías de diagnóstico energético desarrolladas por la dirección de transporte CONAE[23]".

Para la obtención de resultados se aplicó el proyecto de 3 iteraciones, trabajando en la última iteración en un tiempo de 5 semanas con un número total de 24 requerimientos. En la herramienta se hace uso de las relaciones aplicadas a los requisitos teniendo los siguientes valores: (1 Detenido, 0 Remplazado por, 5 Extiende a). En la interpretación de resultados se tiene que el cliente realizo 5 monitoreos (una vez por semana) para visualizar el grado de avance y el gestor del proyecto 15 (tres veces por semana al inicio, mediados y fin) para el seguimiento y trazado, adicionalmente se realizaron 24 revisiones y validaciones. El porcentaje obtenido de la evaluación son representados en las sub-características del modelo *SQuaRE,* para cada uno de los índices de calidad se muestra en la siguiente Tabla 2.

Tipo de vista	Característica	Sub-Característica	Obtenido
Calidad de Producto	Funcionalidad apropiada	Funcionalidad correcta	88%
		Funcionalidad completa	92%
		Funcionalidad adecuada	84%
	Usabilidad	Reconocimiento adecuado	80%
		Fácil de aprender	92%
		Operable	76%
		Protección contra errores de usuario	80%
		Interfaz estética de usuario	92%
		Accesibilidad	76%
	Eficiencia de desempeño	Tiempo de respuesta	84%
		Utilización de recurso	88%
Calidad en Uso	Satisfacción	Cumplimiento de propósito	92%
		Confianza	84%
	Seguridad	Integridad	92%
		Autenticidad	88%

[23] *Comisión nacional para el uso eficiente de la energía.*

Tabla 2. Características de Calidad (*ISO/IEC 25010*)

5 Conclusiones y trabajos futuros

Los principios de esta guía han sido para erradicar las técnicas tradicionales de trazabilidad y reemplazarlos con técnicas que son significativamente más rentables. Mientras que muchos desarrolladores ágiles tienen una tendencia a no usar el concepto de trazabilidad, sin embargo el creciente tamaño y complejidad de los proyectos ágiles nos obliga a encontrar formas de lograr los beneficios de la misma, sin el costo y el esfuerzo de los enfoques tradicionales. Los resultados obtenidos en este trabajo de investigación tanto en la definición del proceso y producto generado proporcionan un marco de referencia favorable en el control de la trazabilidad de requerimientos, brindando un soporte a la organización presentada en el que podemos afirmar que se obtiene una mejor gestión del proyecto de software en desarrollo.

Para trabajo futuro de esta investigación podrían concentrarse en generar un historial de información que permita una posterior extracción del conocimiento y ayude al sistema a evolucionar de manera inteligente. Implementar el desarrollo en otras empresas con un mayor número de iteraciones y obtener más resultados para generar un mayor análisis estadístico que aporte mayor información al proceso y producto.

Referencias

[1] Appleton, B., Cowham, R., & Berczuk, S. "*Lean Traceability: A smattering of strategies and solutions*. CM Crossroads (Configuration Management) págs. 16-57". September, 2007.

[2] Automating traceability for generated software artifacts. Richardson, J y Green, J. 2004. Proceedings of the International Conference on Automated Software Engineering.

[3] Kannenberg A., Saiedian H. Why Software Requirements Traceability Remains a Challenge. The University of Kansas : QAI Quality Assurance Institute, 2010.

[4] B.B. Agarwal, S.P. Tayal, M. Gupta. *Software Engineering & Testing.* s.l. : Jones & Bartlett Publishers, 2009.

[5] Vergara M, Gallo E. European Software Institute. [En línea] 2009. [Citado el: 23 de Noviembre de 2012.] http://www.esi.es/Berrikuntza.

[6] Jacobsson, M. (January de 2009). Implementing traceability in agile software development. Master's Thesis. Lund Institute of Technology.

[7] ISO/IEC 25010. (s.f.). *System and software product Quality Requirements and Evaluation* (SQUARE): "Modelo actualizado de las características de calidad". Recuperado el 20 de 10 de 2013, de Software Guru: http://sg.com.mx/content/view/990.

Luxury I: Lenguaje para control de motores a pasos.

ISC Carlos Arturo Espinoza Galicia [1], Ricardo Francisco Guillen Mallete [2] y Erick Hernandez Nájera [3].
[1, 2, 3] Instituto Tecnológico Superior de Huichapan, El Saucillo S/N, Huichapan, Hgo., CP: 42400, México; cespinoza@iteshu.edu.mx[1], rfguillen@iteshu.edu.mx[2], ehernandez@iteshu.edu.mx[3]

Resumen. El presente trabajo muestra el diseño e implementación de un lenguaje de alto nivel para controlar un modelo 3D que contienen motores a pasos; se diseñó e implemento un compilador y una interfaz gráfica de usuario que permite compilar y ejecutar los programas hechos, dando como resultado una animación que muestra el resultado del mismo.

Palabras Clave: Compilador, ITESHU, Visual C# 4.0, XAML.

1. Introducción

Una de las partes fundamentales de cualquier ingeniero en sistemas computacionales es la creación de software de propósito general y software a la medida, pero un área poco explorada y que sin embargo se imparte en el plan de estudios es el software científico como compiladores e intérpretes, uno de los proyectos de investigación con los que cuenta el Instituto Tecnológico Superior de Huichapan (ITESHU) es el de "Programación de un Brazo Robótico Didáctico", el cual contempla controlar un prototipo básico de un brazo robótico el cual está conformado por 4 motores a pasos que se interconectan al puerto USB de la PC, dicho proyecto será usado como material didáctico en materias de la carrera de Ingeniería en Sistemas Computacionales dentro del ITESHU; este trabajo menciona la forma de cómo se diseñó e implemento un lenguaje de alto nivel básico que se usa en dicho proyecto, así como su interacción en un modelo virtual que en conjunto permite controlar los movimientos del brazo en sus cuatro articulaciones: base, codo, muñeca y garra que permiten rotar, girar, flexionar y sujetar.

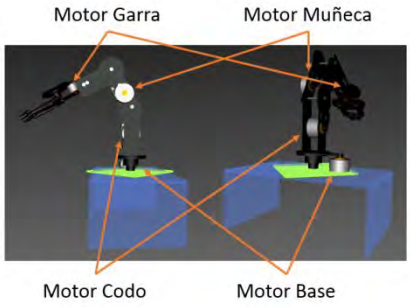


Fig. 1: Modelo 3D del Brazo Robótico con la ubicación y nombre de los motores a pasos.

Un lenguaje, según la Teoría de Lenguajes y Autómatas, es un conjunto de cadenas de algún alfabeto definido [1]. Para un lenguaje de programación, como C por ejemplo, o

cualquier otro lenguaje de programación, los programas validos son un conjunto de cadenas de un alfabeto del lenguaje. Este alfabeto es un subconjunto de caracteres ASCII. Dicho alfabeto puede diferir ligeramente en los distintos lenguajes, pero generalmente se incluyen letras mayúsculas y minúsculas, los dígitos, los signos de puntuación y los símbolos matemáticos [2].

Un compilador es un software que lee un programa escrito en un cierto lenguaje, llamado comúnmente código fuente y lo traduce a un programa equivalente en otro lenguaje, como parte importante del proceso de traducción, el compilador informa al usuario de la presencia de errores en el programa o código fuente [5]. Un compilador normalmente tiene las siguientes fases de análisis léxico, análisis sintáctico, análisis Semántico, generador de código intermedio, optimizador de código y generador de código; sin embargo para este proyecto solo se trabajaran las tres primeras fases, ya que no se generará código máquina alguno, en su lugar, se creara una lista de instrucciones que interpretara el modelo virtual para mover dichos motores.

Anteriormente en el "XXVI Congreso Nacional y XII Congreso Internacional de Informática y Computación 2013" se publicó el artículo "Prototipo Didáctico de Brazo Robot desde el Contexto del estudiante de Ingeniería en Sistemas Computacionales" donde se muestra como fue diseñado un prototipo físico y virtual del brazo robótico (Figura 1), la parte de electrónica y una interfaz de usuario básica para el control del mismo; en el VI Congreso Nacional de Mecatronica, Tecnologías de la información, Energías Renovables e Innovación Agrícola 2014, se publicó el artículo "Programación de un Brazo Robótico" donde se muestra como se construyó una interfaz gráfica de usuario para controlar dicho brazo robótico mediante dispositivos de hardware como Kinect®, Wii Mote® y Joystick®.

2. Estado del Arte

En el mercado existen muchos kits (paquetes) como el "Kit de brazo mecánico para armar, con control remoto alámbrico" [3] el cual contiene los componentes electrónicos con instructivos para armar, sin embargo este carece de una interconexión a la PC; otro ejemplo es el "Diseño de un Brazo Mecánico" [4] donde se menciona la estructura base de un brazo robótico, ese mismo documento hace referencia a una interfaz de puerto paralelo proporcionando un esquema de circuito y un programa ejecutable para controlarlo, pero no indica la lógica de programación; también existe una publicación de la Editorial REDUSERS que se titula "Robótica Avanzada" del Dr. Nicolás Landa Cosio [6], el cual es un compendio de proyectos desarrollados de forma didáctica, mencionando la lista de materiales, la forma de ensamblarlos y la puesta en marcha, una de sus desventajas es el costo de los materiales, además de que se enfoca más al desarrollo del proyecto físico; proyectos como estos se encuentran muchos dentro de la red, careciendo de una secuencia de pasos a seguir.

En lenguajes para Hardware, existen de varios tipos y para varios propósitos, por ejemplo, el lenguaje usado para ***Arduino®*** que si bien como tal es un lenguaje de programación que permite gran uso de funciones directas de hardware y es muy similar al lenguaje C, se requieren conocimientos de electrónica para armar los distintos dispositivos, otro lenguaje como el ***VHDL*** (Combinación de **VHSIC** que es un acrónico de Very High Speed Integrated Circuit y ***HDL*** es a su vez el acrónimo de Hardware Description Language) usado en los ***FPGA®***, los cuales son sumamente potentes y representan una gran apoyo para los diseñadores de hardware, sin embargo es un

lenguaje de uso específico de diseño de hardware, también se puede mencionar el lenguaje Ensamblador, con el cual directamente se pueden programar microcontroladores u otros dispositivos de hardware.

3- Descripción de la técnica usada

Luxury es el nombre que se le da al lenguaje de alto nivel, consta de 3 módulos principales: *El compilador*, el cual informa de errores de construcción del programa a ejecutar, *El generador de instrucciones*, el cual trasforma el programa ya compilado a una cadena de información la cual es interpretada por el ultimo módulo *El modelo virtual 3D*, para realizar los movimientos en los motores solicitados.

El objetivo es brindar al usuario una serie de instrucciones mucho más comprensibles (de alto nivel) y que permitan un desarrollo de proyectos más rápido, para cumplir este objetivo se diseñó una serie de instrucciones que permitiesen mover los motores a cierto ángulo indicando el sentido de giro, de esta forma se definió la sintaxis de como estarían construidas dichas instrucciones: <*motor*> <*[sentido] ángulo*> , el Código 1, es una descripción básica del lenguaje diseñado:

```
Tipos de Motor
==============
Base
Codo
Muñeca
Garra

Rangos de movimientos
=====================
Números enteros entre 0 y 360

Movimiento de cada motor
========================
<Tipo_Motor>(<ángulo>);
Ejemplo:
Base(5);      ->mueve   el   motor
base al ángulo 5°
Base(+);      ->mueve   el   motor
base un grado mas
Codo(-30);  ->mueve  el  codo  al
ángulo        -30°=330°
Muñeca(-); ->mueve la muñeca un
ángulo menos
Garra(15); ->Abre la garra 15°

Comentarios
===========
Ejemplos:
// Esto es un comentario
Base(5);  //así también puede ser
un comentario
//Codo(15);    esta    línea    es
totalmente ignorada

El terminador de línea es el ";"
```

```
Ejemplo de código fuente:
=========================
Inicial
{
//obligatorio   que   defina   la
posición inicial de los motores
Base(90);
Codo(ángulo);
Muñeca(-ángulo);
Garra(10);
Intervalo(100);  //obligatorio  en
ms
}

Secuencia
{
//Comentarios
Garra(15); //Comentarios
Muñeca(-15);
Codo(5);
Base(-10);
Codo(5);
Muñeca(10);
Garra(10);
Garra(5);
```

Código 1: Definición básica del lenguaje de alto nivel a utilizar

Como se puede observar en el Código 1, existe una instrucción la cual se marca como Intervalo, esta instrucción define cuanto tiempo (en milisegundos) pasa entre grado y grado de cada movimiento en el motor.

Una vez definido la estructura del lenguaje y las instrucciones a emplear se diseñó el compilador; como se mencionó anteriormente para este proyecto se omitirán las partes de optimizador de código y generación de código objeto, ya el resultante es una lista de instrucciones que indican que motor se moverá a que grado.

Para la creación del compilador se usó ***Visual C#®*** y para el modelo 3D se usó el lenguaje de marcado ***XAML®*** ambos de ***Microsoft®***, ya que son los lenguajes en los que está construido el resto del proyecto.

El diagrama de la Figura 2, marca el funcionamiento del análisis Léxico, el cual indica que caracteres son aceptados por el compilador [2]:

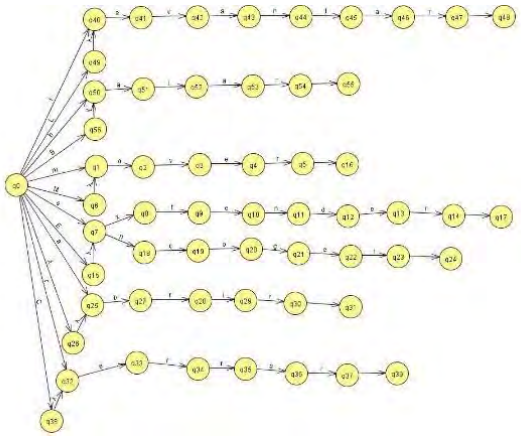


Fig. 2: Diagrama del analizador léxico para el compilador.

El diagrama de la Figura 3, muestra el analizador sintáctico y la Figura 4 muestra el analizador semántico usado respectivamente.

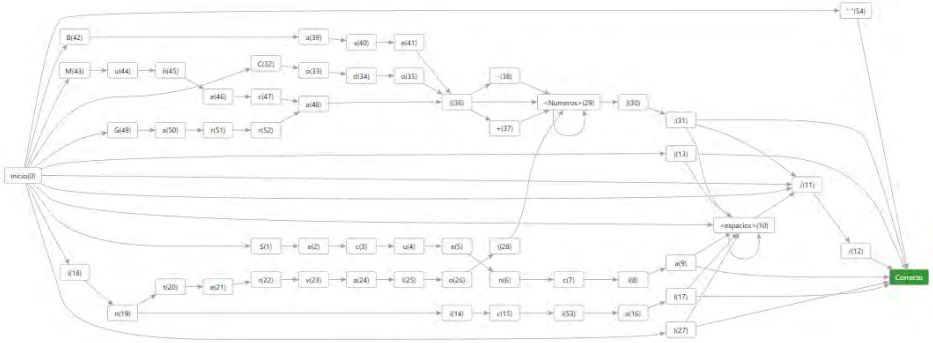


Fig. 3: Diagrama de estados del analizador sintáctico

Una vez definido los diagramas anteriores, se procedió a desarrollar diagramas UML que ayuden al momento de la creación del código fuente. Como primer paso, se definió una clase para informar de los errores encontrados en el proceso de compilado (Figura 5), posteriormente se definió una clase que realizaría el proceso de compilación (Figura 6), una vez que el script no contenga errores, este es trasferido a la clase que maneja el tratamiento de este para su conversión a la lista de movimientos (Figura 7) donde simplemente manda llamar el método del motor adecuado informando el ángulo al que se debe mover (Figura 8).

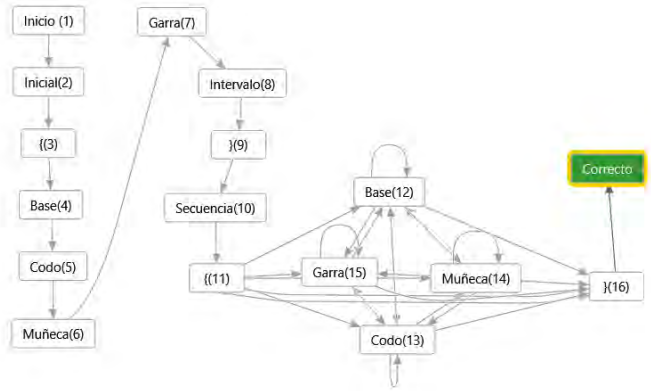


Fig. 4: Diagrama del analizador semántico

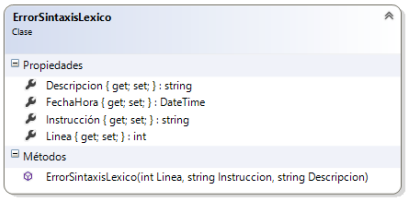


Fig. 2: Diagrama UML de la clase que define un error encontrado en el proceso de compilación

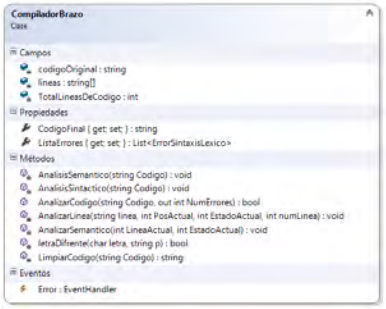


Figura 3: Diagrama UML de la clase encargada del proceso de compilación

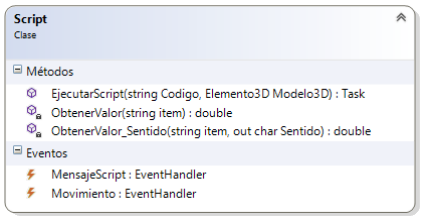


Fig. 4: Diagrama UML de la clase encargada de la conversión de código fuente a lista de movimientos y ejecución en el modelo 3D

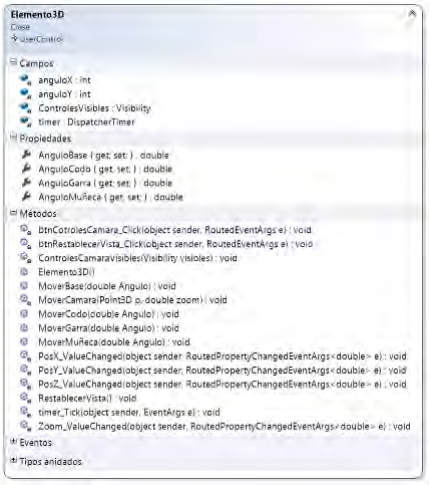


Fig. 5: Diagrama UML de la clase que define el funcionamiento del modelo 3D

4. Resultados Experimentales

Como resultado se obtiene una interfaz de usuario que tiene un editor de texto simple para este lenguaje de programación, donde el usuario final puede escribir, compilar y probar sus programas, los cuales se verán reflejados en el modelo 3D, dicha interfaz, proporciona una plantilla base la cual permite iniciar con el proceso de escritura del programa (Figura 9), al compilar, si el programa tiene errores, el usuario será informado de cuantos errores tiene y en la parte de notificaciones o mensajes se mostrará el error encontrado y la línea donde se encuentra localizado (Figura 10), después de corregir dichos errores, se recompila nuevamente donde se informa que el programa está escrito correctamente (Figura 11), posteriormente se ejecuta la secuencia automatizada donde al terminar se informa de la finalización de la misma (Figura 12).

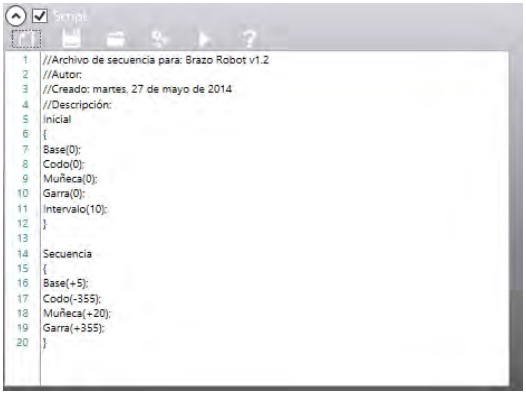


Fig. 9: Plantilla proporcionada por la interfaz de usuario.

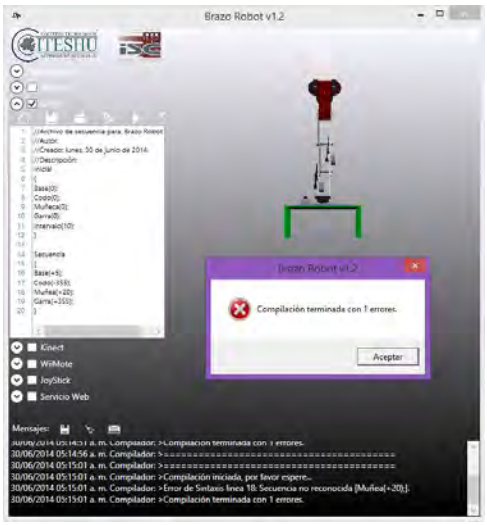


Fig. 10: Proceso de compilación terminado donde se informa los errores encontrados.

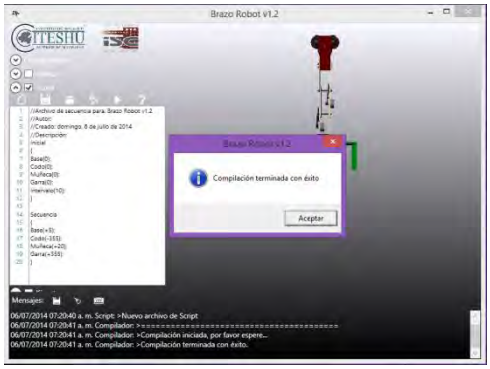


Fig. 11: Proceso de compilación terminada sin errores encontrados.

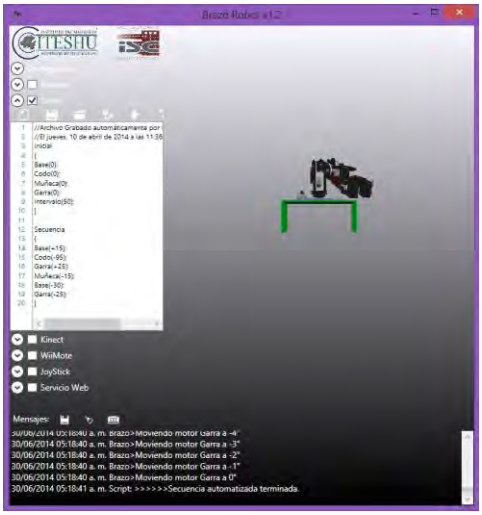


Fig. 12: Ejecución terminada del programa.

5. Conclusiones y Trabajos Futuros de Investigación.

El trabajo realizado ya se usa como caso de estudio en las materias de la carrera de Ingeniería en Sistemas Computacionales del ITESHU, más específicamente en la materias de sistemas programables y lenguajes y autómatas I y II. Tanto la generación que esta por egresar, como la que actualmente está en 7mo semestre y los docentes involucrados, están trabajando en la mejora del mismo, en un futuro se tiene previsto integrar la parte de ciclos, variables y decisiones; si bien esta es la primera versión del lenguaje, se tiene un conjunto de instrucciones más entendibles para un usuario, otra parte importante es que ya se tiene un gran avance con la parte de interconexión con hardware, de modo que los movimientos también se realicen físicamente, posteriormente se tiene pensado retroalimentación de parte de sensores para hacerlo mucho más eficiente.

El lenguaje presentado es para un modelo específico, pero se puede generalizar a otros modelos o aplicaciones, de la misma forma se pudieran añadir otro tipo de actuadores o motores.

Referencias

[1] J. E. Hopcroft, R. Motwani y J. D. Ullman, Introducción a la teoría de autómatas, lenguajes y computación., Madrid: Person Educación S.A., 2002.

[2] A. V. Aho, R. Sethi y J. D. Ullman, Compiladores. Principios, técnicas y herramientas, Massachusetts. E.U.A.: AddisonWesley Publishing Company, 1986.

[3] Steren Shop, «Kits Avanzados,» Steren, [En línea]. Available: http://www.sterenshop.com.mx/catalogo/prod.asp?f=9&sf=118&c=1055&p=102090. [Último acceso: 10 Junio 2013].

[4] Todo Robot, «Controladora para cuatro Motores Paso a Paso,» 10 octubre 200. [En línea]. Available: http://www.todorobot.com.ar/proyectos/4stepper/4stepper.htm. [Último acceso: 27 marzo 2013].

[5] K. C. Louden, Compilers Constructions, Principles and Practtices, PWS Publishing, 1997.

[6] USER SHOP, «Robótica Avanzada,» [En línea]. Available: http://usershop.redusers.com/ficha.asp?marca=libros&numero=lpcu179. [Último acceso: 25 Junio 2013].

CURRICULUM VITAE

DATOS PERSONALES

Nombre: Carlos Arturo Espinoza Galicia
Domicilio: Manuel Robledo No. 15 Int. 5 Barrió San Mateo, Huichapan, Hgo. CP: 42400
Fecha de Nacimiento: 26 de Noviembre de 1982
Teléfono: (045) 7737364381
Credencial de Elector: 121371762778
CURP: EIGC821126HHGSLR07
No. de Cartilla Militar: C-5935047
E-mail: espinoza_vfp@hotmail.com cespinoza@iteshu.edu.mx
Sitio Web: http://carlosespinoza.azurewebsites.net/

FORMACIÓN ACADÉMICA

- Maestría en Ingeniería y Desarrollo de Software, Colegio de Posgrado de Desarrollo de Software, iniciando en Marzo de 2014 (actualmente estudiando)
- Maestría en Educación Superior Área Administración y Gestión Educativa Obteniendo Mención Honorifica de aprovechamiento (Promedio de 9.68); Titulo en Tramite, Universidad La Salle, Pachuca Hgo. 2008-2010.
- Ingeniero en Sistemas Computacionales. Instituto Tecnológico de Pachuca 2000-2005 (Titulado, 9 de Marzo de 2006)

- Técnico en Computación. Centro de Bachillerato Tecnológico Industrial y de Servicios No. 59 1996-2000 (Titulado, 28 de Octubre de 2004)
- Secundaria Técnica No. 2, Cd. Sahagun, Hgo. 1994-1996
- Primaria Estado de Hidalgo. Cd. Sahagun, Hgo. 1988-1994

FORMACION COMPLEMENTARIA

- Certificación Oracle Certified Professional, Java SE 6 Programer, Oracle, Septiembre 2013
- Curso-Taller: Auditor Interno de los Sistemas de Gestión de Calidad ISO 9001:2008, ISO 14001:2004, MEG:2003, ITESHU, julio 2013
- Taller: Gestion del currículum, didáctica y evaluación de competencias desde el enfoque socio-formativo, ITESHU, junio 2013
- Diplomado "Windows Phone 8 Application Development" Microsoft Mayo 2013
- Diplomado "Introducción a Windows Communication Fundation 4.0" TI-Capacitación Febrero 2013
- Curso "Instrumentos Virtuales con LabView" ITESHU Diciembre 2012
- Diplomado "Windows 8 Application Developmet" Micrisoft Diciembre 2012
- Diplomado "Cloud Computing Development Solutions with Windows Azure", Microsoft, Abril 2012
- Diplomado "Introducción al lenguaje de Programación Visual C# 4 con Visual Studio 2010" TI-Capacitación, Marzo 2012
- Curso-Taller: Competencias Profesionales en la instrumentación didactica, Unidad de Desarrollo Empresarial, S.C., Enero 2012
- Curso-Taller: Tutoría, Unidad de Desarrollo Empresarial, S.C., Enero 2012
- Diplomado "Fundamentos de Desarrollo con HTML 5 e Internet Explorer 9" Microsoft, Diciembre 2011
- Curso Auditor líder Interno en Sistemas de Gestión de la Calidad, COMPITE 29 de Septiembre de 2010
- Curso Auditorías Internas en Sistemas de Calidad, COMPITE 10 de Marzo de 2010
- Curso Interpretación de la Norma ISO 9001:2008, COMPITE 11 de Febrero de 2010
- Curso Acciones Correctivas y Preventivas, Medición, Análisis y Mejora de Procesos, 14 de Noviembre de 2009
- Curso Introducción a los Sistemas de Administración de la Calidad, 13 de Junio 2009
- Seminario Tecnológico Windows Vista, Microsoft 7 de agosto de 2008.
- Seminario Tecnológico SQL Server 2008, Microsoft 7 de Agosto de 2008.
- Curso de Inglés, Comprensión de Textos por la Universidad La Salle Pachuca, Agosto 2008
- Curso de Seguridad en Redes, Symantec de México. 2004
- Curso de Mercadotecnia, Colegio de México. 2000
- Curso de Ingles Básico Técnico enfocado a la computación, Instituto Tecnológico de Pachuca, 2003
- Curso Bases de Datos, Grupo Intercom, Curso en Línea, 2004
- Curso Bases de SQL Server, Grupo Intercom, Curso en Línea, 2004
- Curso Programación en C, Grupo Intercom, Curso en Línea, 2004
- Curso Virus y Ataques Externos, Grupo Intercom, Curso en Línea, 2004
- Curso Redes Inalámbricas, Grupo Intercom, Curso en Línea, 2004
- Curso Como vender mas, psicología aplicada a las ventas, Grupo Intercom, Curso en Línea, 2004

EXPERENCIA LABORAL

1. Docente Investigador en la División de Ingeniería en Sistemas Computacionales del Instituto Tecnológico Superior de Huichapan, actualmente laborando, Jefe Inmediato Lic. Marcos Mendoza, participando en las siguientes actividades:
    a. Semestre Agosto – Diciembre 2011

i. Materias Impartidas
    1. Matemáticas Discretas
    2. Fundamentos de investigación
    3. Sistemas Operativos
    4. Lenguaje Ensamblador
ii. Asesorías de las materias:
    1. Lenguaje Ensamblador
    2. Taller de Investigación
iii. Proyectos de Investigación:
    1. Integración de tecnologías EPC, RFID en estándares de programación para el control y seguimiento de activos en el ITESHU
    2. Sistema tipo BPM/ERP para PyMES

b. Semestre Enero – Julio 2012
i. Materias Impartidas
    1. Programación Orientada a Objetos
    2. Tópicos Avanzados de Programación
    3. Inteligencia Artificial
    4. Interfaces
ii. Tutorías
    1. 2do Semestre de Sistemas
iii. Asesoría a Materias para exámenes Globales/Especiales
    1. Fundamentos de Programación
    2. Sistemas Operativos
    3. Taller de Base de Datos
iv. Proyectos de Investigación
    1. Sistema tipo BPM/ERP para PyMES
v. Concursos
    1. 1ra Ronda Internacional de Imagine Cup 2012
vi. Asesoría a Residencias Profesionales
    1. Reingeniería del Carrito de Compras, Grupo Adorote.
vii. Sinodal de Titulación
    1. Maribel Paz Sánchez
    2. Diana Zúñiga Rivera
viii. Coordinador del Student Club Ixiptlayot

c. Semestre Agosto – Diciembre 2012
i. Materias Impartidas
    1. La Tecnología y su Entorno
    2. Graficación
    3. Fundamentos de Programación
    4. Fundamentos de Ingeniería de Software
ii. Tutorías
    1. 1er Semestre de Sistemas
iii. Asesoría a Materias para exámenes Globales/Especiales
    1. Graficación
    2. Fundamentos de Base de Datos
    3. Fundamentos de Programación
iv. Proyectos de Investigación
    1. Sistema Tipo BPM/ERP para PyMES
v. Producción Académica (artículos publicados en congresos)
    1. Conjuntando CloudComputing y Tecnologias EPC y RFID, Congreso Nacional y Congreso Internacional de informtica y Computación, Octubre 2012, ISBN: 978-607-707-563-9
    2. MarketMatic, un Ejemplo de uso de Cloud Computing y Tecnologías EPC y RFID, 5° Congreso Nacional de Mecatronica y Tecnologías Inteligentes, Septiembre 2012, ISBN: 978-607-95556-1-0
vi. Talleres impartidos en el Congreso Nacional de Mecatronica y Tecnologías Inteligentes
    1. Servicios Web
    2. Programación Paralela con C#
vii. Sinodal de Titulación
    1. Julio César Zamudio Montalvo
    2. José Luis Gómez Hernández

viii. Conferencias Impartidas
    1. TIC's para el apoyo educativo
ix. Participación en Concursos:
    1. Etapa Local de Expociencias y Emprendedores 2012, proyecto "MarketMatic"
    2. Etapa Estatal de Expociencias y Emprendedores 2012, proyecto "MarketMatic"
x. Coordinador del Student Club Ixiptlayot
xi. Participación como Jurado en la etapa Estatal de Expociencias y Emprendedores Pachuca 2012

d. Semestre Enero – Julio 2013
    i. Materias Impartidas
        1. Programación Orientada a Objetos
        2. Inteligencia Artificial
        3. Productividad Laboral
        4. Graficación
        5. Lenguajes de Interfaz
    ii. Tutorías
        1. 2do Semestre Sistemas
    iii. Asesoría a Materias para exámenes globales/especiales
        1. Graficación
        2. Sistemas de Base de Datos
        3. Taller de Base de Datos
        4. TIC's
    iv. Sinodal de Titulación
        1. Edgar Guerrero Zamudio
    v. Proyectos de Investigación
        1. Sistema Tipo BPM/ERP para PyMES
    vi. Secretario de Academia de la División de Ingeniería en Sistemas Computacionales
    vii. Coordinador del Student Club Ixiptlayot

e. Semestre Agosto – Diciembre 2013
    i. Materias Impartidas
        1. Fundamentos de Programación
        2. Taller de Base de Datos
        3. Simulación
        4. Programación de Dispositivos Móviles
        5. Sistemas Programables
    ii. Asesoría a Materias para exámenes globales
        1. Taller de Base de Datos
    iii. Producción Académica (artículos publicados en congresos)
        1. Ofuscador de Código para Macros de Microsoft Excel, Segundo Congreso de Administración e Ingeniera Industrial, CADII 2013, Septiembre 2013, ISBN: 978-607-95556-2-7
        2. Prototipo Didáctico de Brazo Robot desde el Contexto del estudiante de Ingeniería en Sistemas Computacionales, Congreso Nacional y Congreso Internacional de informática y Computación, Octubre 2013, ISBN: en tramite
    iv. Talleres Impartidos en el Segundo Congreso de Administración e Ingeniería Industrial 2013:
        1. Windows 8.1
    v. Conferencias
        1. El campo profesional del ingeniero, SNCyT en ITESHU, octubre 2013
    vi. Proyectos de Investigación
        1. Programación de un Brazo Robótico Didáctico
    vii. Asesor de Residencias Profesionales
        1. Módulo de Indicadores para el Sistema Integral de Gestión Administrativa (SIGA)
        2. Desarrollo e Implementación del Módulo de Control de Inventarios para San Sebastián Tenochtitlán S.P.R. de R.L
    viii. Participación en Concursos:

1. 1er lugar en la etapa Local de Expociencias y Emprendedores 2013, proyecto “Turismo para personas con discapacidad”
2. 1er lugar en la etapa Estatal de Expociencias y Emprendedores 2013, proyecto “Turismo para Personas con Discapacidad”
3. Participación en la etapa Nacional de Expociencias y Emprendedores 2013, proyecto “Turismo para personas con Discapacidad”

ix. Participación en Proyectos con el Sector Productivo:
1. Planta Biotecnológica para cultivar micro algas y procesamiento para producir alimento balanceado para productos acuícolas

x. Presidente de Academia de la División de Ingeniería en Sistemas Computacionales
xi. Coordinador del Student Club Ixiptlayot

f. Semestre Enero – Julio 2014
i. Materias impartidas:
1. Circuitos Eléctricos y Aplicaciones Digitales
2. Graficación
3. Lenguaje de Interfaz
4. Tecnologías de Computo en la Nube

ii. Asesoría a materias para exámenes globales
1. Simulación
2. Tecnologías de Computo en la Nube
3. Lenguaje de Interfaz
4. Principios eléctricos y aplicaciones digitales

iii. Talleres impartidos dentro del ITESHU
1. Java Programer SE6

iv. Proyectos de Investigación
1. Programación de un Brazo Robótico Didáctico

v. Asesor de residencias profesionales
1. Página dinámica de servicio social

vi. Presidente de Academia de la División de Ingeniería en Sistemas Computacionales
vii. Coordinador del Student Club Ixiptlayot

2. Docente a nivel medio superior y superior en la Universidad del Nuevo México, Campus Huichapan. Jefe Inmediato MDP Rosalìa Ariadna Lòpez Zapiàin .
3. Asesor Externo del Área de Sistemas, Traslados Universales S.A. Jefe inmediato Ing. Andres Montaño Sanchez (Actualmente laborando)
4. Coordinador del Área de Sistemas, Traslados Universales S.A. de C.V. Dando soporte técnico y de desarrollo de software a las bases de Cd. Sahagun Hidalgo y Monterrey Nuevo León, Encargado del Desarrollo de Software y Base de datos; desarrollando sistema de Seguimiento mediante GPS y J2ME; Mantenimiento de la Página web de la empresa; utilizando herramientas como: ASP.Net, VB.Net, C#, Visual Studio 2010, SQL Server, Windows Server 2000 y 2008, J2ME, Web Services; Jefe Inmediato Ing. Andres Montaño Sanchez (Agosto 2008-Julio 2011)
5. Coordinador del Área de Capacitación de Traslados Universales, coordinando y preparando cursos para la capacitación del personal en las áreas Operativa y Administrativa, Jefe Inmediato Ing. Andres Montaño Sanchez (Agosto 2008-Julio 2011)
6. Integrante del Comité de Calidad ISO 9001:2008 de Traslados Universales, con capacitación como Auditor Líder Interno y habiendo participado dentro de 4 auditorías externas, una de ellas de recertificación por parte de BSI y 4 auditorías internas participando como auditor, Jefe Inmediato Ing. Andres Montaño Sanchez (Agosto 2008-Julio 2011)
7. Profesor Instituto Tomas Garrigue Masaryk, Impartiendo materias como Informática y Física(Agosto 2009-Julio 2011), Jefe Inmediato M.A. Maricarmen Rivas Hernandez.
8. Asesoría en TI además de Reparación y Mantenimiento de equipo de Cómputo desde 2006 (desarrollando en tiempos libres).
9. Encargado del área de desarrollo de software, Grupo Financiero FRABESA SOFOM; Encargado de desarrollar software para la administración de la SOFOM utilizando tecnologías .Net de Microsoft, Jefe Inmediato: Maricruz López Mera, Septiembre 2009-Diciembre 2010.
10. Programador, Gestor México, BBVA Bancomer, Proyecto SegProy, Jefe Inmediato: Martin Oliver Morales, Agosto 2009

11. Encargado del laboratorio de computo en la Preparatoria Fray Toribio De Benavente, a cargo de 24 estaciones de trabajo y un servidor, desarrollo de sistema de administración en Visual Studio 2005, ASP.Net y VB.Net Jefe inmediato: Ing. Ludmila Holkova Oborna, directora del Plantel.
12. Asesor para trabajo de prácticas profesionales de jóvenes de diversas instituciones educativas y asesor de proyectos de Ciencia y tecnología de la preparatoria, destacando Software de Ingles y transmisor de datos por medio de la red eléctrica. Jefe inmediato: Ing. Ludmila Holkova Oborna, Directora del Plantel.
13. Titular del Área de Informática y Docente en la Preparatoria Fray Toribio de Benavente, con 6 Materias por semestre, entre las que destacan: Lógica computacional y programación, Base de Datos I y II, Redes, entre otras. Jefe inmediato: Ing. Ludmila Holkova Oborna, directora del Plantel.
14. Profesor en Grupo Educativo DATCOM (CD. Sahagún, Hgo.) Teniendo a mí cargo 8 grupos, impartiendo materias como Base de Datos, Programación, Redes, Contabilidad Digital y Paquetería Diversa.
15. Prácticas Profesionales en Refinería "Miguel Hidalgo" de PEMEX en el Área de Procesos de Información, Desarrollo de Sistema de Seguimiento de Compras para la Intranet de la Empresa. Jefe Inmediato. ISC José de Jesús Ramírez. Director del Departamento. 2005
16. Servicio Social en la Universidad Autónoma del Estado de Hidalgo en el centro de computo de Matemáticas Aplicadas, teniendo a cargo 30 estaciones de trabajo y un servidor, mantenimiento de la red y equipo de computo. Jefe inmediato: ISC Robles., Administrador del Centro de Cómputo. 2004.
17. Servicio Social en la Secretaria de Turismo del Estado de Hidalgo, en el departamento de Informática, dentro del programa Servicios Informativos por Internet. Jefe Inmediato: Lic. Lida Olivares. 2004.
18. Practicas Profesionales en la empresa: MILAND (Maquilas Industriales Landaverde), Desarrollo de sistema informático para el control de proveedores. Jefe Inmediato: Lic. En Finanzas Arturo Landaverde Miranda. 1999
19. Practicas Profesionales en Despacho contable privado, Desarrollo de Sistema de Contabilidad. Jefe Inmediato: C.P. Emilio López Arrollo. 1999.
20. Trabajo desarrollado en la Empresa "Espigal Multicosturas", Software de Nomina y Recursos Humanos. Jefe Inmediato Rosa Adriana Galicia Delgadillo. 1998.

IDIOMAS

- Curso de Ingles, Comprensión de Textos por la Universidad La Salle Pachuca 2007
- Nivel Básico de Ingles hablado y escrito enfocado a la computación. Estudiado mientras cursaba estudios de la Ingeniería en Sistemas Computacionales; Noviembre 2003.

INFORMATICA

- Dominio del entorno Windows, Windows NT, Windows 2003 y 2008 Server y de los siguientes programas de Microsoft: Word, Excel, PowerPoint, Publisher, Access, Project, FrontPage, Visio, One Note, Outlook, Outlook Express, Internet Explorer.
- Dominio de Internet y Correo Electrónico.
- Conocimiento y uso de manejadores de Base de Datos como Microsoft SQL Server 2000 y 2008 Express Edition, Access y MySql.
- Dominio de programación en los siguientes lenguajes: Visual Basic 6.0, C, C++, Visual FoxPro, Cobol, Java, PHP, HTML, ASP, ASP.NET, Visual Basic.NET, C#.NET y plataforma Network.NET desde 1.1 hasta 4.0 además de conocimiento de desarrollo para dispositivos móviles con Visual Studio 2012 y Windows Phone 7.5, 8.0 con Visual Studio 2012, XAML y WPF.
- Conocimiento de Programas como: Dreamweaver y Flash de Macromedia, AceHtml y AceFtp de Visicomedia, COI, NOI, SAE de ASPEL, Autocad, PhotoShop, Corel Draw.
- Dominio y conocimiento de Redes informáticas, Implantación. Seguridad, Monitoreo y Mantenimiento.
- Conocimiento del Entorno Linux y Solaris
- Dominio de Instalación y mantenimiento de PC 's en todas sus partes, Hardware y Software.

OTROS

- Deportes: Tae Kwon Do (Cinta Roja), Squash, Natación.
- Pasatiempos: Música, Leer, Bailar, etc.

DATOS PERSONALES

Nombre: Ricardo Francisco Guillén Mallette
Domicilio: Nochebuena #56 Col. Comevi Banthi, San Juan del Río, Qro CP 76808
Fecha de Nacimiento: 03 de Abril de 1965, Teléfono (427) 274-0574, Celular:(427) 128-8113
No. Curriculum Vitae Único (Conacyt): 526399 Usuario: X_rguillen4283
E-mail: rfguillenm@gmail.com / rfguillen@iteshu.edu.mx

FORMACIÓN ACADÉMICA

- **Ingeniero en Sistemas Computacionales. Instituto Tecnológico de San Luis Potosí 1983-1988 (Titulado, 1991)**
- Maestría en Informática Administrativa; Facultad de Contaduría Pública, Universidad Autónoma de Nuevo León, 1989-1991
- Licenciatura en Homeopatía, Escuela Superior de Homeopatía, Guadalajara Jalisco, Mayo 2010 a Mayo 2014
- Técnico en Electrónica, Instituto Tecnológico de Saltillo, 1980-1982

PERFILES

Perfil Técnico:

- Más de 10 años de experiencia en servicios de Consultoría de Procesos de Mantenimiento Industrial apoyado en el ERP JDEdwards para la industria cementera (1993-2011).
- Más de 5 años de experiencia en servicios de Consultoría de Procesos de Inventarios y Compras apoyados en el ERP JDEdwards para la Industria cementera.
- Más de 10 años de Experiencia en configuración, implementación, Soporte, Análisis y Desarrollo de programas, Migración de datos en el ERP JDEdwards World Software con plataforma en AS400.
- Más de 5 años de experiencia trabajando con equipos multiculturales en proyectos internacionales de Implementación y Soporte JDEdwards efectuados vía remota en países como: Estados Unidos de América, Colombia, Venezuela, España, Egipto, Emiratos Árabes y Filipinas. En forma local en países como Venezuela, Panamá, República Dominicana y a lo largo del territorio nacional en diferentes plantas cementeras de México.
- Facilidad para estar aprendiendo en corto tiempo, autodidacta, pro-activo, trabajo por objetivos, trabajo en equipo, liderazgo de proyectos.

Perfil Docente:

- Docente del Instituto Tecnológico Superior de Huichapan (ITESHU), **Agosto 2012 a la fecha** (Materias impartidas: Matemáticas discretas, Arquitectura de Computadoras, Planificación y Modelado, Programación Avanzada I y II, Sistemas Operativos, Sistemas de Bases de Datos Distribuidas, Lenguajes y Autómatas I y II, Ingeniería de Sistemas, Software de Aplicación Ejecutiva, Desarrollo sustentable, Inteligencia artificial, Programación lógica y funcional, Taller de investigación, Estructura de datos, Fundamentos de investigación, Taller de ética, Ingeniería de sistemas.

- Participación en el proyecto de Investigación: "Sistema tipo BPM/ERP para PYME's", proyecto "Programación de un brazo robótico didáctico", proyecto externo con Conacyt, Universidad Autónoma de Querétaro y empresa Caru Ardica S.A de C.V. llamado "Desarrollo Tecnológico para el control de precisión de helióstatos basado en sistemas embebidos para aplicación termosolar".
- Presentación de un artículo en Congreso CADI 2013, un artículo aceptado en Congreso Conamti 2014
- Más de 10 años de experiencia en la impartición de cursos técnicos correspondientes a algunos módulos del Sistema ERP JDEdwards, (1993-2011).
- Impartición puntual de clases en diferentes instituciones: ITSLP, UAC (Licenciatura y Maestría), UANL (Licenciatura) e ITESHU (2012-a la fecha).
- Más de 3 años impartiendo cursos cada 4 meses de medicinas alternativas Reiki, Homeopatía, Fitoterapia, PNL y Técnicas de aprendizaje. (2009-2011).
- Más de 3 años impartiendo cursos de capacitación en CAD (Diseño asistido por computadora).
- Durante el estudio de la carrera ISC realizando actividades de Promotor de Ajedrez en ITSLP.

Perfil salud:

- Más de 6 años sanando personas conocidas en lo físico, mental, emocional utilizando medicina alternativa: Bioenergía, Homeopatía, Fitoterapia, PNL e Hipnosis terapéutica (**2008-A la fecha**).
- Master-practitioner en Reiki, grado 3, Certificación por Miguel Balderas Bolaños.
- Master-practitioner en PNL con Gabriel Guerrero, Design Human Engineering con Richard Bandler
- Diplomado de Especialización en Hipnosis Clínica Ericksoniana en el Instituto Milton H. Erickson de Monterrey (IMHEM) en 2007
- Licenciado en Homeopatía realizando el servicio social en el ITESHU.

Language:

Intermediate level of English (70%)

_____

Ing. Ricardo Francisco Guillén Mallette

I.S.C. ERICK HERNÁNDEZ NAJERA

Independencia #4, San Pablo Atlazalpan, Chalco, Estado de México, C.P. 56620
| Tel: 17-358049 | CEL 044-55-15076079 | isc_erickhn@hotmail.com |

FORMACIÓN ACADÉMICA

Tecnológico de Estudios Superiores de Chalco
Ingeniero en Sistemas Computacionales
Esp**. Desarrollo de Sistemas Web**

Periodo: **2004 - 2009**

Cedula Profesional: **6475630**

EXPERIENCIA DOCENTE

"Instituto Tecnológico Superior de Huichapan"
Materias Impartidas: **Programación Web, Ingeniería de Software, Tópicos Avanzados de Programación, Arquitectura de Computadoras, Matemáticas Discretas, Programación Orientada a Objetos, Fundamentos de Programación – Nivel Superior**
13/01/2014 – Fecha actual
Dirección: Domicilio conocido S/N, El Saucillo, Huichapan, Hidalgo. C.P. 42411 Teléfonos: (761) 72 4 80 79

"Instituto Tecnológico de Tláhuac"
Materias Impartidas: **Graficación, Programación Web – Nivel Superior**

20/08/2012 – 13/12/2013
Dirección: Av. Estanislao Ramírez # 301 Col. Ampliación Selene, Delegación Tláhuac, México D.F., C.P. 13420 Tels. 2594-4096, 5866-1640

"Tecnológico de Estudios Superiores de Chalco"
Materias Impartidas: **Desarrollo de proyectos de software, Programación Web, Graficación, Taller de bases de datos, Fundamentos de Bases de Datos, Principios Eléctricos y Aplicaciones Digitales, Arquitectura de Computadoras, Sistemas Operativos, Taller de Sistemas Operativos, Sistemas Electrónicos para Informática, Probabilidad y Estadística, Programación Orientada a Objetos, Matemáticas Discretas, Fundamentos de Programación – Nivel Superior**

01/10/2010 – 9/01/2014
Dirección: Carretera Federal México Cuautla s/n, La Candelaria Tlapala, Chalco, Edo de México Teléfonos: 01 (55) 59 82 10 89, 59 82 08 48

Instituto "María del Carmen Plascencia"
Materias Impartidas: **Matemáticas I, II, III, IV y V, Física I, II y III, Algoritmos Computacionales, Lenguaje de Programación I y II, Administración de redes – Nivel Bachillerato**

07/02/2008 – 31/07/2012
Dirección: Av. Cuauhtémoc #102, Ixtapaluca, Edo de México C.P. 56560 Teléfono: 01 (55) 15-636030

Bachillerato Tecnológico y Centro de Estudios Tecnológicos y Universitarios "Enrique Fernández Garmendia"
Materias Impartidas: **Introducción a la Informática, Computación I y II, Paquetería Computacional, Software de Diseño Grafico, Informática Aplicada I y II, Laboratorio de Informática IV y V, Algoritmos Computacionales, Lenguaje de Programación I y II, Administración de redes – Nivel Bachillerato**

07/08/2006 – 31/01/2008
Dirección: 5 de Febrero #13, Amecameca, Edo de México C.P. 56900 Teléfono: (01) 597-97 82819

EXPERIENCIA RELACIONADA

Bachillerato Tecnológico y Centro de Estudios Tecnológicos y Universitarios "Enrique Fernández Garmendia"
Área Administrativa
29/01/2007 – 31/01/2008

- Informes y atención a alumnos, padres de familia y Docentes.
- Encargado del área de control de pagos.
- Encargado del área de servicios escolares.
- Administrador del sistema de control escolar (**Academium**).

DESARROLLO DE PROYECTOS DE SOFTWARE

Sitio Web y Sistema de Registro en línea COMANTI 2014.

Congreso Nacional de Mecatrónica, Tecnologías de la Información, Energías Renovables e Innovación Agrícola – Instituto Tecnológico Superior de Huichapan **2014**

Sistema Registro de Citas en Línea (SIRCL).

Coordinación de Educación Básica, Escuela Normal de Chalco **2013**

Sistema Estimulo al Desempeño Docente (SEDD).

Dirección de Institutos Tecnológicos Descentralizados **2011 – Fecha actual**

Sistema para el Control de Pagos (SICOP).

Colegio Centro Pedagógico Danfer **2010**

Sistema para Programa Municipal de Dotación de Útiles Escolares (SPDUE)
H. Ayuntamiento de Netzahualcóyotl, Edo de México. **2010**

Sistema de Control de Evaluación en Línea (SICEL)
Escuela "María del Carmen Plascencia" **2008 - 2009**

Sistema de Control Escolar
Tecnológico de Estudios Superiores de Chalco **2007 - 2008**

IDIOMAS

Ingles Nivel Básico, Intermedio y Avanzado
Tecnológico de Estudios Superiores de Chalco & CELEX **2007 - 2009**

CURSOS DE ACTUALIZACION

Curso "Programación Orientada a Objetos"
Desarrollo Web y Escuela Tecnologías de la Información S.L. **2013**

Diplomado "Master de PHP "
Aula Formativa Soluciones Online S.L. **2013**

Curso "JQuery"
Desarrollo Web y Escuela Tecnologías de la Información S.L. **2013**

Curso "¿Cómo controlar la conducta del alumno en el aula?"
Tecnológico de Estudios Superiores de Chalco **2013**

Certificación “Microsoft Office Specialist”
CERTIPORT (Instituto Tecnológico Fidel Velázquez) **2011**

Taller “Estrategias de enseñanza - aprendizaje”
Instituto Mexica, Chicoloapan de Juárez, Estado de México **2011**

Taller “Herramientas Avanzadas de Dreamweaver para Paginas Dinámicas con PHP y MySQL”
Aula Formativa Soluciones Online S.L. **2009**

Curso “Actualización Windows Vista y Office 2007”
CCPM – Chalco **2008**

Taller “Mantenimiento Preventivo y Correctivo para Computadoras”
Tecnológico de Estudios Superiores de Chalco **2008**

Seminario “SQL Server 2008”
Microsoft TeschNet – Universidad La Salle **2008**

DATOS PERSONALES

CURP: **HENE851126HMCRJR02**, RFC: **HENE851126K43**, Licencia de manejo: **N04840174** Tipo: **A**, Cartilla Servicio Militar: **0272501**

GESTIÓN DE PRUEBAS DE SOFTWARE: "Análisis para la su implementación en el ITS de Puerto Vallarta."

Sergio Alan Martinez Romero [1] y Miguel Ángel Gallardo Lemus [2]

[1] UNIVERSIDAD AUTONOMA DE GUADALAJARA e Instituto Tecnológico de Puerto Vallarta- Corea del Sur 600, Colonia El Mangal, Puerto Vallarta, Jalisco, 48338. México
sergio.martinez@tecvallarta.edu.mx
[2] UNIVERSIDAD AUTONOMA DE GUADALAJARA e Instituto Tecnológico de Puerto Vallarta- Corea del Sur 600, Colonia El Mangal, Puerto Vallarta, Jalisco, 48338. México
Miguel.gallardo@tecvallarta.edu.mx

Resumen La gestión de pruebas debe de ser orientada a la verificación del cumplimiento de los requisitos de las aplicaciones. Los servicios procurados son tanto de definición de planes de pruebas, definición de metodologías de pruebas personalizadas, adecuación de herramientas de pruebas, destinada a la externalización completa de la gestión de las pruebas sobre una aplicación en concreto o sobre el global de software producido.

En el IT de PV se realizó los análisis para la gestión de pruebas de software basados en las investigaciones empíricas y técnicas considerando en el proceso de control de calidad seleccionando el estándar ISO/IEC 29110. Se analizan los documentos ISO/IEC 29110 además de otros para seleccionar criterios y a posteriori definir los parámetros y requerimientos para su implementación.

El propósito del proyecto es adaptar y modificar requerimientos del estándar

ISO/IEC 29110, fusionando los roles, documentos y actividades con otros modelos.

Palabras Clave: Ingeniería de Software -Calidad – gestión de pruebas- organización y planeación- ISO/IEC 29110.

1 Introducción

Una de las partes importantes del desarrollo de software y de las que requiere más atención, es la del área de pruebas de software, con el incremento de mejora del desarrollo de software, aunado al grado de complejidad, es una parte coyuntural para localizar tanto sea posible errores en el programa, para cumplir con la garantía de alta calidad en los productos de software, el fortalecer la administración y organización es específicamente importante para esta área.

En la gestión de pruebas de Software la problemática a resolver, es cómo garantizar que los métodos de pruebas de software, en la vida de los proyectos sean aplicados con éxito en el interior, y originar los resultados deseados.

Considerando la perspectiva del ciclo de vida del software, las acciones referidas al proceso de pruebas han constituido, la parte integral del desarrollo del mismo y en consecuencia, también su gestión [9], las acciones referidas al proceso de pruebas; es que el objeto que se somete a prueba, la operatividad debe ser considerablemente enérgica, sin embargo, debe de estar respaldada en: especificaciones, diseño de instructivos y documentación.

Puesto que un error en el diseño de las mismas generara un conflicto más, lo cual pondría en riesgo la garantía de la calidad del software.

2 Estado del Arte

El desarrollar y proponer estándares a la gestión de pruebas de software es necesario considerarla una disciplina, sustentando que en la actualidad las mismas se han incrementado su grado de complejidad, debido a gran cantidad de metodologías de desarrollo, lenguajes de programación, sistemas operativos, hardware, etc., sumado en la misma proporción que las técnicas y los paradigmas del desarrollo de software. Una demostración es el estudio que se presenta [3]; Originariamente solo en el ámbito académico se desarrollaba software en Instituciones educativas y algunas gubernamentales como el DoD (Departamento de Defensa de USA) y la NATO(Organización del Tratado del Atlántico Norte), las aplicaciones de índole comercial eran desarrolladas internamente, después pasaron a ser desarrolladas por empresa especializada en la fabricación o la venta de software, diseñado para la masa o nichos de mercados, conocidas como ISVs (proveedor de software independiente). Normalmente, esto se aplica para la aplicación específica o software incorporado, de otros productores de software [12].

En lo referente a modelos se desarrollaron del tradicional en cascada [14], el modelo en espiral [4], y el modelo V[4], posteriormente los modelos agiles e iterativos[5][12], incluyendo las ventajas en sus preceptos; En cuanto a las estrategias de desarrollo se trasladó de un desarrollo centralizado a el de software distribuido generando las fases de desarrollo (Gestion de Requisitos, Desarrollo y Pruebas), realizadas por diferentes equipos [14]; la generación de estándares para la implementación de proyectos y procesos de desarrollo de software como: SPEM (Modelo de Proceso o Meta Modelado), teorías unificadoras de Programación (UTP) ISO, CMM posteriormente CMMI[17].

Los modelos de desarrollo han evolucionado desde los años 80 en que aparecieron los modelos orientados a objetos, en los 90 los modelos basados en componentes y en la actualidad han aparecido los modelos orientados a servicios. Aunque SOA (arquitectura orientada a servicios) no es un concepto nuevo, fue descrita por primera vez por Gartner en 1996, se ha visto aumentada su aparición en la actualidad, en gran medida por el aumento de uso de servicios web[4]; y la cual introduce cambios en las actividades del ciclo de vida en el desarrollo de software, principalmente en la fase de pruebas cuyo impacto es crítico para garantizar la calidad del software.

Una práctica que a partir de finales de los años noventa ha tomado gran fuerza en la comunidad de Ingeniería del Software (industria e investigadores) es la Mejora de Procesos Software (conocida por las siglas inglesas SPI, Software Process Improvement) en pequeñas organizaciones software.[12]

3 Gestión de Pruebas

Una razón del impulso a esta área es que el IT de Puerto Vallarta, desea implementar el departamento de desarrollo de software, en equipos pequeños de desarrollo, por lo que al considerar en que las características especiales de las pequeñas organizaciones software hacen que los programas de mejora de procesos deban aplicarse de un modo particular y visiblemente diferente a como se hace en las grandes organizaciones y que esto no es tan sencillo como el hecho de considerar dichos programas de mejora versiones a escala de las grandes compañías. Además, las propuestas de mejora del SEI e ISO (como CMMI, IDEAL, SCAMPI, ISO 12207, ISO 15504) han sido creadas y están estructuradas para ser utilizados por organizaciones grandes y difícilmente pueden ser aplicadas a organizaciones pequeñas. Esto se debe a que un proyecto de mejora supone gran inversión en dinero, tiempo y recursos, así como a la alta complejidad de las recomendaciones y al hecho de que el retorno de la inversión se produce a largo plazo. [1]

Es por esto que las pruebas de software deben apoyarse en estándares que revisan los aspectos fundamentales que debe considerar todo proceso de pruebas. Debido a esta complejidad actualmente se cuentan con una gran cantidad de software diseñado exclusivamente para la etapa de pruebas pero que no abarca todo el ciclo de vida [16].

En la gestión de pruebas de Software la problemática a resolver, es cómo garantizar que los métodos de pruebas de software, en la vida de los proyectos sean aplicados con éxito en el interior, y originar los resultados deseados.

Ante estos problemas el estándar ISO/IEC 29119 para pruebas de software es un referente internacional en el ámbito de las pruebas software y permite eliminar las inconsistencias existentes entre las normas actuales, así como cubrir aquellas áreas del proceso de pruebas de software que simplemente no habían sido tratadas hasta ahora en el resto de normas publicadas

La gestión de pruebas de software involucra diferentes aspectos como, la organización del equipo de pruebas, gestión de proyectos, gestión de seguimiento a defectos de software (errores). La presente investigación propone la implementación de un modelo de gestión de pruebas de software que sistematice el estándar ISO/IEC 29119 [16], que cubrirá el ciclo de vida completo, a través del análisis, diseño, implementación y mantenimiento de las pruebas software.

Por lo que se consideraran como inicio a cuatro elementos principales: la organización, la estructura del ambiente de desarrollo de pruebas de software, el equipo de desarrollo y las Pruebas de software Figura1.

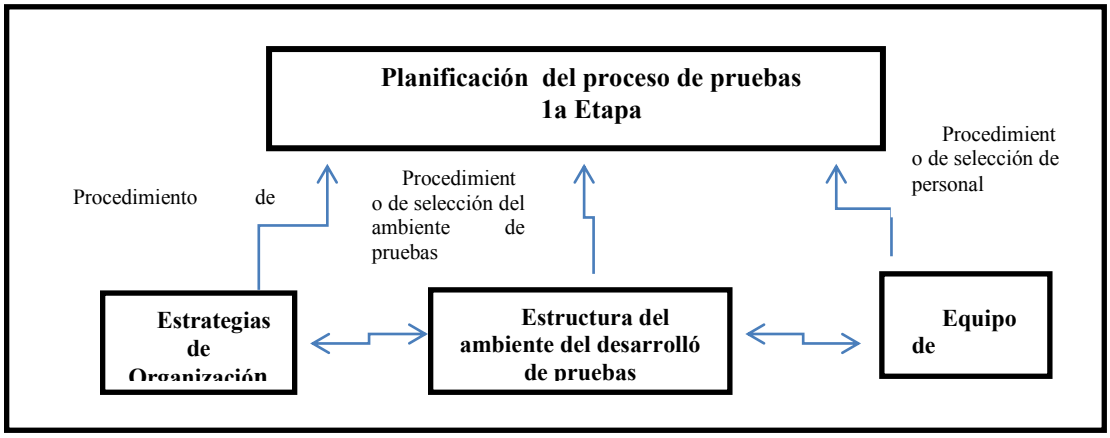


Figura 1

3.1 la Organización del equipo de desarrollo de Pruebas

Frecuentemente en algunos proyectos omiten la importancia de las pruebas, al emplear por lo general, programadores que actúan temporalmente como verificadores, de ahí es importante que el equipo de pruebas, este bajo la responsabilidad y realización de un Líder de Pruebas de Software (TSL).

Ya que define dentro de sus funciones, se encuentra la determinar el proceso de pruebas, así como también establecer los parámetros y criterios a utilizar en las mismas.

Así mismo la gestión, la Planificación, seguimiento, asignación de recursos, cronograma y métricas del proceso de pruebas, son responsabilidades que deben de ser consideradas en la organización y asignadas al Líder de Pruebas de Software; dentro de las cuales se hallan: Planificación y Diseño de las Pruebas; Definición, diseño y desarrollo de la estrategia de pruebas; Diseñar los Scripts de prueba; Preparación de datos de pruebas; Desarrollo de casos de prueba; Automatización de pruebas; Ejecución de pruebas; Verificación de la documentación del resultado de las pruebas; Registro, análisis y seguimiento de métricas: Reporte e informe de avance de actividades;

Por lo que el equipo debe de estar constituido primero por un buen líder, con experiencia en los procesos de desarrollo y en la comprensión e interpretación de los defectos o errores más comunes, además de afinidad, carisma para conseguir los objetivos. en el área de pruebas de software, la gestión para organizar el equipo debe de ser contemplada de la forma más simple posible en su configuración.

3.2 Estructura del ambiente de desarrollo de pruebas

Así mismo, es importante destacar que para la implementación de gestión de pruebas, se debe de considerar aspectos relevantes como el ambiente de desarrollo, esto con el propósito de optimizar el proceso y obtener un producto que cumpla con estándar calidad.

El desarrollo de software hoy en día se ha caracterizado por combinación de equipos de proyectos y mantenimientos trabajando de forma simultánea, con cronogramas cada vez más exigentes y desarrollando sistemas que interactúan con otras aplicaciones y plataformas. En un escenario como este, la gestión de los ambientes (entornos) de pruebas integrales (System Integration Tests, ó SIT), adquiere gran importancia para asegurar que los procesos en producción c u m p l a n con los niveles de calidad.

En la gestión de pruebas es parte fundamental considerar los ambientes de pruebas integrales de sistema (SIT).

El definir las características del ambiente de pruebas SIT, aunado a las restricciones que deben aplicarse sobre el ambiente, la homologación con producción, implementar procedimientos a para una óptima gestión de los ambientes y prácticas que deberán asumir los equipos de pruebas en los diferentes proyectos.

La buena práctica dentro de la administración de ambientes de pruebas SIT, se debe considerar lo siguiente: Características que convienen tener los ambientes; Restricciones a aplicar a los mismos; No deberán ser utilizados para actividades de desarrollo o producción; Homologación con el ambiente de producción; Procedimientos para implementar la gestión de ambiente de pruebas integrales; Procedimiento de uso del ambiente por el equipo de pruebas.

Lo anterior debe ser sustento para determinar la elección del estándar o modelo de pruebas para garantizar calidad del software.

3.3 Equipo de Desarrollo de Pruebas

La labor de determinar que ambientes se necesitan y asegurar que estén operativos cuando se necesiten es lo que debe realizar, un Líder de proyecto de Pruebas ó Tester de operaciones, lo cual implica considerar el equipo de desarrollo.

Al considerar el equipo de desarrollo de pruebas, tanto interno como externo a la organización, se debe suministrar con antelación sus requerimientos de ambiente a la administración de desarrollo de Software.

Definir el equipo de desarrollo de pruebas, aunado a los requerimientos de ambiente, deberán ser documentados en cada proyecto, usando los formatos y plantillas preestablecidos, deben ser usados estrictamente sólo para desarrollo y pruebas.

El Tester debe realizar su trabajo exclusivamente en el ambiente de desarrollo de pruebas, nunca en otros directamente, debera ser distinto a los de desarrollo y producción.

El equipo debe convenir en sólo contener datos de pruebas, por lo que debe disponerse de procedimiento para eliminar datos sensibles, cuando se transfieren del ambiente de producción.

El equipo debe de considerar múltiples ambientes de desarrollo, pero necesariamente asegurar que periódicamente sea homologado entre ellos y con el ambiente de pruebas.

El Tester debe de contar con y un proceso claramente definido y auditable para la asignación de ambientes, uso compartido, accesos múltiples, acuerdos de nivel de servicio, actualizaciones, Upgrades, instalaciones, .asistencia a los múltiples equipos usuarios del ambiente, emitir reportes de uso y carga que asistan a la planeación**.**

3.4 Plan de Pruebas

La prueba es una actividad fundamental en muchos procesos de desarrollo, incluyendo el del software, ya que estas permiten detectar la presencia de errores que pudieran generar las entradas o salidas de datos y comportamientos inapropiados durante su ejecución y es una de las vías más importantes para determinar el estado de la calidad de un producto de software.

Está dirigido a componentes del sistema en su totalidad, con el objetivo de medir el grado en que se cumple con los requerimientos.

En el plan se usan casos de prueba, especificados de forma estructurada mediante sus objetivos, métodos y técnicas usadas las cuales se describen en un plan de prueba.

Es indispensable contar con un Plan de Pruebas de Software para especificar cuidadosamente las funciones a probar, como serán hechas esas pruebas, quienes serán los responsables y el cronograma para su ejecución.

El Plan de Pruebas de Software puede aplicarse a todo el Proyecto de Desarrollo de Software, o delimitar a una iteración o conjunto de casos, asimismo, podrá definir jerarquías de casos de prueba a valorar.

El plan debe de considerar, Historial de Versiones, Ficha del Proyecto, Aprobaciones, Alcance de las Pruebas, Criterios de Aceptación o Rechazo, Entregables, Recursos, Planificación y Organización.

3.4 Descripción del Proceso de Gestion de Pruebas

En el análisis para la implementación de la gestión de pruebas en el IT de PV se consideró como base la normatividad ISO ya que ha desarrollado “La Metodología ISO para evaluar y comunicar los beneficios económicos de los estándares”; consideramos como objetivos clave los siguientes:

• Proporcionar un conjunto de métodos que midan el impacto de los estándares en la creación de valor organizacional.

• Proporcionar la toma de decisiones con criterios claros y prácticos para evaluar el uso de estándares.

El enfoque utilizado para cualquier empresa comprende cuatro pasos:

• Entender la cadena de valor de la empresa

• Analizar los motivadores de valor

• Identificar los impactos de los estándares

• Evaluar y consolidar los resultados

Realizando un análisis de la ISO / IEC / IEEE 29119 "Pruebas de software" es un conjunto internacionalmente acordado de normas para que podrían ser utilizadas dentro de cualquier ciclo de vida de desarrollo de software o de una organización.

El perfil más simple de la serie ISO/IEC 29110 (el perfil inicial), fue el utilizado como base para desarrollar los procesos de administración e proyectos de pequeña escala. La guía ISO/IEC 29110 de ingeniería y gestión describe un proceso de administración de proyectos y un proceso de implementación.

El propósito del proceso de administración de proyectos será establecer y llevar a cabo de manera sistemática las tareas a fin de cumplir con los objetivos de costo, tiempo y calidad esperados.

Durante la actividad de la Gestion de pruebas, se desarrollara el plan del mismo. Entonces, las tareas de evaluación y control son utilizadas para comparar el progreso

del proyecto contra el plan del proyecto (son manejadas para evaluar cómo avanza el proyecto con relación al plan).

Se actuara en caso necesario, para corregir desviaciones del plan de proyecto o para incorporar cambios al plan.

La actividad de cierre de proyecto concentra los entregables producidos durante el proceso de implementación, como el software o el manual de usuario, y envía para obtener la aceptación por escrito del cliente para formalizar el cierre del proyecto.

Se establece un repositorio físico y digital para guardar los productos de trabajo y para controlar sus versiones durante el proyecto.

4. Gestión de Pruebas en el IT de PV

Los proyectos en este departamento se clasificaran, en categorías de acuerdo a duración, tamaño, número de disciplinas relacionadas y honorarios de ingeniería.

Se subdividirán los proyectos en tres categorías: pequeña, mediana y gran escala.

El enfoque empleado está basado en el meta- problema desarrollado por Potter y Sakry[12], fue utilizado para establecer las prioridades del Departamento de desarrollo de software y asegurar que las metas establecidas en el área de pruebas este enfocada e involucre los siguientes pasos:

1) Identificación de la meta de negocio

2) Agrupación de la meta y el problema

3) Priorización del problema

4) Desarrollo de un plan de acción

La responsabilidad de la Gestion de Pruebas en el IT de PV, es la desarrollar e implementar procesos de Pruebas en proyectos de pequeña y mediana escala de software, se consideró de importancia sustentarlo en el estándar internacional ISO/IEC/IEEE 29119, estándar internacional para pruebas de software , proporciona las directrices cubriendo todos los aspectos del ciclo de vida como :

- Composición consistente, definiciones, procesos, procedimientos y técnicas para las pruebas de software.

- Soluciona la dispersión existente actualmente
- Cubre huecos no cubiertos por estándares existentes
- Adoptado por los comités de normalización nacionales, IEEE y BSI

La nueva serie ISO/IEC/IEE 29110 permite a la empresa desarrollar pruebas de software que ofreciendo, un enfoque estructurado, seleccionando que las acciones requeridas sean las más esenciales, a fin de simplificar la gestión de pruebas del proyecto.

Finalmente, estimando las prioridades de los objetivos y el costo por cada uno es necesario establecer las fases de implementación.

Se seleccionaron los siguientes criterios:

- El estándar adecuado para proyectos de pequeña escala;
- Identificar rápidamente las formas y formatos disponibles para la administración de Pruebas en los proyectos de software;
- Fácil evaluación de la implementación de los procesos.

Se previene que para mejorar las prácticas se deben de desarrollar las siguientes verificaciones en:

- Proceso de administración de proyectos pequeños
- Proceso básico de administración de Pruebas de software
- Elaboración de propuestas de servicio
- Planeación detallada de Pruebas

Al verificarse las soluciones en el contexto de un proyecto real, servirá como patrón de evaluación, consistente pero sobretodo alcanzable y exhaustivo en las soluciones propuestas.

Es necesario considerar la implementación de herramientas para realización y administración de pruebas, las cuales serán evaluadas y autorizadas por parte de los líderes de proyecto.

Cuando se realicen los ajustes finales a los procesos de administración de proyectos, selección de herramientas de prueba de software, sera necesario implementar una estrategia que cubra los siguientes aspectos:

- Comunicación para informar a los líderes de Pruebas, para solucionar y mitigar los impactos de acciones que se generen por situaciones problemáticas durante los cambios que se realicen.
- Capacitación a los líderes de proyecto.
- Difusión de los cambios y correcciones aceptadas, utilizando medios adecuados de información.

5 Conclusiones y trabajos Futuros.

Las pruebas de software permiten pasar de forma confiable del cómodo ambiente planteado por la ingeniería de software, es decir del controlado ambiente de análisis, diseño y construcción, al exigente mundo real en el cual los entornos de producción someten los productos a todo tipo de esfuerzos identificar y cuantificar factores de calidad como: capacidad de uso, capacidad de prueba, capacidad de mantenimiento, capacidad de ser medible, capacidad de ser confiable y a desarrollar prácticas de ingeniería que contribuyen a la obtención de productos de alta calidad.

En 2012, IEE muestra el estándar ISO/IEC 29119, cuyo objetivo es ser un estándar definitivo sobre pruebas software, que recoja y estandarice el vocabulario, los procesos, técnicas de documentación, etc., del ciclo de vida de las pruebas.

Con esta normativa se pretende generar el modelo de gestión de pruebas de software en el IT de PV en una segunda etapa en donde se considerara; Definiciones y Vocabulario, Proceso de Pruebas, Documentación de Pruebas, Técnicas de Pruebas.

El aseguramiento de la calidad del producto, es un aspecto que sera tomado con mucha relevancia.

Otro punto que debe desarrollarse es el de las herramientas CASE para apoyar los procesos de administración de proyectos y pruebas, que serían bastante útiles, colaborando a integrar rápidamente el conocimiento requerido para ejecutar los procesos de pruebas.

La norma ISO/IEC/IEEE 29119 permite el desarrollo de documentación de los procesos de gestión de pruebas, para proyectos de pequeña escala. La Figura 2 ilustra la relación de los procesos definidos en el estándar.

Adicionalmente, se contempla desarrollar un plan de administración de riesgos para la prevención, específicamente para reducir la probabilidad y minimizar el impacto de ciertos eventos en el proceso del proyecto.

El mejoramiento del proceso de pruebas no solo es posible a través de la retroalimentación, la automatización de pruebas sugiere también la forma no solo de acelerar el diseño y ejecución de casos de prueba, sino también de obtener un mayor grado de cobertura en las pruebas, una posible implementación estaría orientada a un sistema experto parametrizable para la aprobación de productos y procesos mediante la automatización del proceso de prueba.

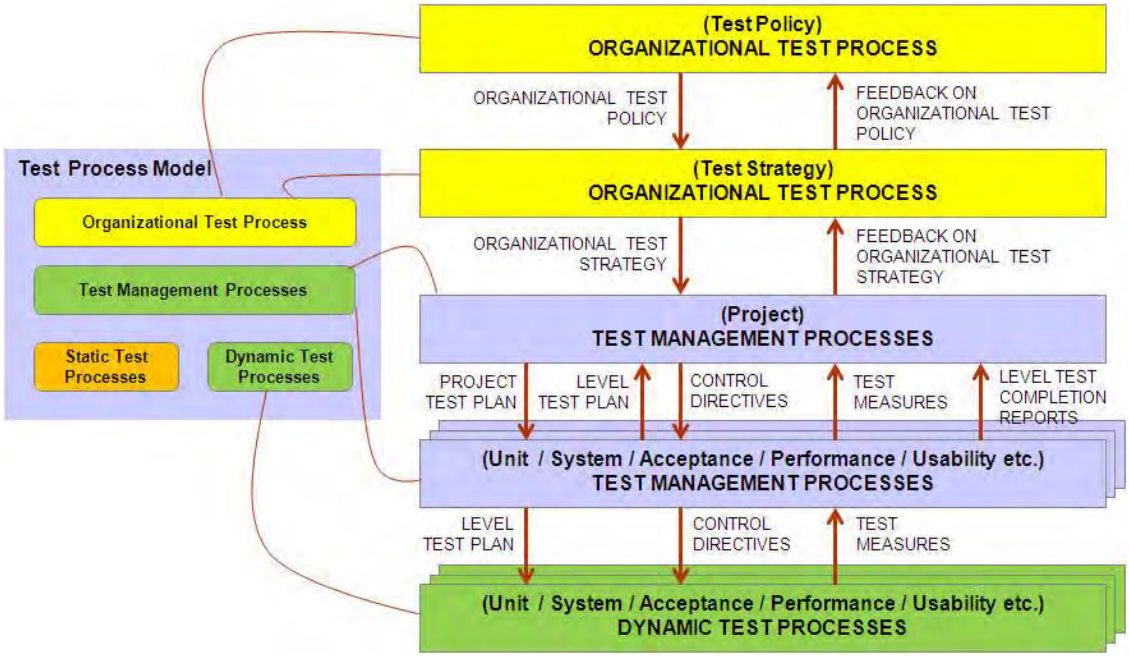


Figura 2. Procesos de ISO/IEC 29119 [16]

Referencias

[1] Aguirre A., Pardo C., Mejía M.,F., Pantoja W., L., Pino F., Reporte de Experiencias de la Aplicación de Competisoft en Cinco Mipymes Colombianas Revista EIA, ISSN 1794-1237 Número 13, p. 107-122. Escuela de Ingeniería de Antioquia, Medellín (Colombia) Julio 2010

[2] Balzer R. A 15 Year Perspective on Automatic Programming. IEEE Transactions on Software Engineering, vol.11, núm.11, pages 1257-1268, November 1985.

[3] Bertolino A., Software Testing Research: Achievements, Challenges,Dreams, Future of software Engineering2007.FOSE 07, ISBN: 0-7695-2829-5,pp85-103,223-25May 2007.R.
[4] Boehm B. W, Turner, R., A Spiral model of software development and enhancement, computer, vol. 21, No. 5, pages 61-72.1988
[5] Fernández, S., Piattini, V, M., Modelo para el gobierno de las TIC basado en las normas ISO. C.M. (c.). Cap. 10, pp. 297-318. Editorial AENOR, ISBN: 978-84-8143-790-4. 2012
[6] Fritzshe M., Keil,P., Agile Methods and CMMI: Compatibility or conflict,e Informatica Software Engineering Journal, Volume !, Issue 1, PP.9-26, 2007
[7] Gib Tom, Evolutionary Delivery versus the "Water Fall Model". ACM sigsoftware Engineering notes, vol. 10 No.3, page 50, july 1985.
[8] Jacboson I., Booch, G., Rumbaugh J. El Proceso Unificado de Desarrollo de Software, Addison Wesley 2000.
[9] IEEE Computer society, "Guide to the Software Engineering Body of Knowledge"- SWEBOK; BourqueP., Dupuis R., Moore J., W.,IEEE#Computer Society,2004.
[10] Larman, C., Agile& Iterative Development, a manager guide, Addison Wesley, 2004. [11] Mendoza A., Utility Computing Technologies, Standards and Strategies, Artech House, 2007.
[12] Potter, N., Sakry, M., Making Process Improvement Work. Addison-Wesley - Pearson Education, 2002.
[13] Pino, F., J., Vidal, J., Garcia, F., Piattini, M., Modelo para la Implementación de procesos en pequeñas organizaciones Software, en XII Jornadas de Ingenieria del software y Bases de datos(JISDBD 2007), Zaragoza, España Septiembre, 2007.
[14] Smite, D., Global software development project in one of the biggest companies in Latvia: is geographical distribution a problem?."Software Process Improvement and Practice",John Wiley and sons, Vol. 11, Issue 1, pp. 61-76.2009
[15] Turmino M., Bournissen J., Bracalenti c., Schlemper., Gestion y Desarrollo de Proyectos de software: Metodología Ágil basada en Telecomunicaciones- MATE Revista Latinoamericana de Ingenieria de Software, (6):253-258 ISSN 2314-2642. 2013
[16] Tuya, J., Estado actual del estándar ISO/IEC 29119 - Software Testing. IV Taller sobre Pruebas en Ingeniería del Software (PRIS), San Sebastián, 2009-09-08.
[17] Software Engineering Institute, CMMI for Development, Version 1.2, Pittsburg, PA15213-3890, august 2007.

CURRICULUM VITAE

1) Sergio Alan Martinez Romero

- Ingeniero Químico Industrial, Aspirante al Título de Maestro en Ciencias de la Computación, en el área de desarrollo de software en la Universidad Autónoma de Guadalajara.
- Docente en el Instituto Tecnológico de Puerto Vallarta, desde septiembre de 1999- a la fecha.
- En las carreras de Ingenieria electromecánica, Ingenieria en sistemas computacionales, Ingenieria en gestión empresarial en formato presencial y a Distancia.
- Asesor técnico Independiente en el área de procesos, mantenimiento y selección de equipo 1999-2004.
- Asesor Técnico en el área de procesos de Panamerican Enterprice S.A 1991-1996
- Ingeniero con experiencia en procesos, planeación y factibilidad de proyectos .Atisa-Atkins 1984-1990.

Artículos publicados:

- ADMINISTRACIÓN DE RIESGOS EN CMMI, ISO/IEC 12207, SPICE Y MOPROSOFT;

- ISBN: 978-1-4276-3507-5; Congreso de investigación y Posgrado de las Instituciones Tecnológicas del Estado de Chihuahua. CIPITECH-2008

2) Miguel Ángel Gallardo Lemus

- Ingeniero en sistemas computacionales, Aspirante al título de Maestro en Ciencias De La Computación, en Área de Desarrollo de Software, Becado por promedio por el Instituto Tecnológico Superior De Puerto Vallarta en la Universidad Autónoma De Guadalajara
- Docente en Instituto Tecnológico Superior De Puerto Vallarta, desde Agosto de 2004 – Actualidad
- Impartiendo las asignaturas en las carreras de, Ingeniería en Sistemas Computacionales, Ingeniería en Tecnologías de la Información y Telecomunicaciones e Ingeniería en Gestión Empresarial, en formato presencial y en formato a distancia utilizando la plataforma MOODLE.
- Diseñador de cursos en la modalidad de educación a distancia
- Administrador De Sistemas y Tecnologías de soporte para plataforma de Educación a Distancia(MOODLE) en el Instituto Tecnológico Superior De Puerto Vallarta 2012- a la fecha realizando :
    - Administración de sistemas linux
    - Administración de bases de datos mysql
    - Administración de servicios de hosting con apache
    - implementación de sistemas web con tecnología JOOMLA y WordPress
    - Administración de Moodle
- Desarrollador de software para el Instituto Tecnológico Superior De Puerto Vallarta de agosto 2008 - agosto 2010
- Desarrollador de software independiente (frelance) De Sistemas De Software, Desktop, Web y Móvil
- Artículos publicados: ADMINISTRACIÓN DE RIESGOS EN CMMI, ISO/IEC 12207, SPICE Y MOPROSOFT; ISBN: 978-1-4276-3507-5; Congreso de investigación y Posgrado de las Instituciones Tecnológicas del Estado de Chihuahua. CIPITECH-2008

Propuesta de Arquitectura de Software para la Integración de los Sistemas en el Instituto Tecnológico Superior de Puerto Vallarta

Área de conocimiento: Ingeniería de Software

Miguel Ángel Gallardo Lemus [1], Fabián García Carrillo [2]
[1] Universidad Autónoma de Guadalajara, Av. Patria 1201, Lomas del Valle C.P. 45129, Zapopan, Jalisco. México
miguel.gallardo.lemus@gmail.com
[2] Universidad Autónoma de Guadalajara, Av. Patria 1201, Lomas del Valle C.P. 45129, Zapopan, Jalisco. México
fabian.garcia@tecvallarta.edu.mx

Resumen. Todo desarrollo de software tiene sus cimientos sobre la arquitectura de software, sin embargo cuando en un mismo sitio se encuentran diferentes sistemas que fueron desarrollados de forma independientes uno del otro sin seguir ninguna arquitectura en común, pero que tienen que compartir información uno con otro es cuando comienzan los problemas. Para evitar este tipo de situaciones problemáticas se propone una arquitectura o diseño orientado a dominio para lograr la integración de los sistemas existentes y futuros en el ITSPV. La capa de dominio que se propone en esta arquitectura, contiene todas las clases que modelan dicho dominio, sus entidades, sus relaciones y las reglas de negocio que cumplan. De esta manera se busca la integración de los sistemas existentes y los que puedan ser desarrollados en el futuro según se tenga la necesidad.

Palabras claves: Arquitectura de software, patrón, estilos, capa de dominio, calidad.

1 Introducción

En el campo de la ingeniería de software, el concepto de arquitectura ha sido fundamentada como estrategia para enfrentar la complejidad del desarrollo de soluciones informáticas y establecer acuerdos de diseño de alto nivel para ellas (Bass, Clements, Kazman, 2003)[1].

En el Instituto Tecnológico Superior de Puerto Vallarta (ITSPV), se necesita que el software se apegue a los aspectos arquitecturales más actuales, contemplando las mejores prácticas sobre las arquitecturas de software y sobre calidad del software, es por eso que existe la necesidad de una guía que establezca el camino a seguir en la implementación del software para el Instituto.

La arquitectura de software es fundamental a la hora de garantizar que la solución cumpla con los criterios de calidad establecidos en los requisitos. Dichos criterios de calidad han sido caracterizados y estandarizados en propuestas como ISO 9126, la cual establece una taxonomía de características y sub características tanto internas como externas, que debe poseer un producto de software.

El objetivo de este trabajo es presentar una visión de lo que es la arquitectura de software ayer, hoy y mañana, dentro del instituto Tecnológico Superior de Puerto Vallarta, basándonos en una propuesta de arquitectura de software que le permita al ITSPV, construir software más robusto y complejo en cuanto al diseño arquitectónico y el diseño en general.

2 Estado del arte

Todo software conlleva de manera implícita una arquitectura, aun cuando no se haya diseñado o no se esté consciente de su existencia. Sin embargo es preferible comenzar la construcción de un software con una idea clara de los componentes del software, su organización y su relación entre ellos, para poder transmitirla correctamente y que el software cumpla los requerimientos funcionales, pero también los no funcionales. Es decir es preferible diseñar la arquitectura antes de construir el software.

Las arquitecturas de software se pueden clasificar de acuerdo al tipo de problema que resuelven, si es que el problema en sí, no es único ni particular, y si es así se consideran arquitecturas o modelos de referencia.

Los modelos de referencia, surgen a partir de la experiencia de resolver problemas e implementar soluciones, en un dominio específico, esas arquitecturas creadas para resolver dichos problemas, usadas y probadas en distintas ocasiones, son base de muchos ejemplos de desarrollo ya existentes, algunas tienen bastante tiempo siendo usadas, otras son recientemente propuestas, algunas de ellas son:

- Cliente servidor.
- N - Capas.
- Distribuido.
- Arquitectura Orientada a Objetos.
- Arquitectura Bus de Servicios.
- En pipeline.

De manera concreta, al diseñar una arquitectura de software debemos crear y representar componentes que interactúan entre ellos y tengan asignadas tareas específicas, además de organizarlos de forma tal que se logren los requerimientos establecidos. Podemos partir de soluciones ya probadas, con la intención de no comenzar de cero las propuestas y utilizar modelos que han funcionado.

Existen otros tipos de soluciones genéricas a problemas comunes, en el diseño y construcción de software, estas soluciones probadas se conocen como estilos arquitectónicos y patrones, los patrones pueden ser arquitectónicos y de diseño, los tres están relacionados entre sí y su aplicación corresponde al nivel de abstracción de cada uno, que van de lo general a lo particular, siendo los más generales los estilos y los más particulares los patrones de diseño.

Un estilo arquitectónico consiste de una colección de tipos de componentes con una descripción del patrón o interacción a través de ellos. (Traducido de Clements, P.; Bass, L.; Kazman, R. Software Architecture in Practice. Edited by SEI Series, first edition, SEI Series in Software Engineering: Addison Wesley, 2003.)[1].

Los patrones son una regla que expresa una relación entre un contexto, un problema y una solución.

Los patrones arquitectónicos Buschmann et al. (1996)[4] describen los principios fundamentales de la arquitectura de un sistema de software. Identifica los subsistemas, define sus responsabilidades y establece las reglas y guías para organizar las relaciones entre ellos

Los patrones de diseño proveen un esquema para refinar los subsistemas y las relaciones entre los componentes de un sistema. Describen una estructura recurrente de comunicación entre componentes para resolver un problema general de diseño dentro de un contexto particular (Buschman et al. 1996)[4]. Tienden a operar independientemente de un paradigma particular de programación o lenguaje de programación.

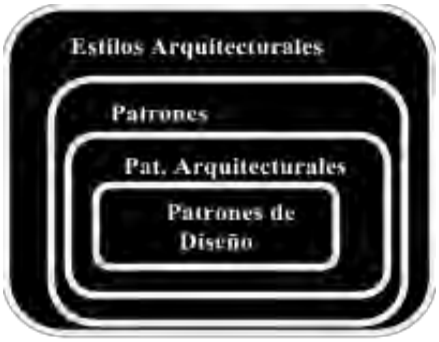


Fig. 1. Niveles de abstracción de los estilos arquitecturales y los patrones

Las bases teóricas utilizadas para este trabajo, indican que se debe tener conceptos sólidos como equipo de desarrollo de software, en donde el rol del arquitecto de software viene a ser un pilar dentro de caya desarrollo. El arquitecto es el que diseña el sistema a través de patrones, estilos y modelos de referencia, todo esto lo lleva a tener una visión más global del trabajo, como lo hace el arquitecto que diseña casas, el arquitecto de software tiene que expresar sus ideas a las demás personas del equipo, a través del diseño y contemplando todos los aspectos del dominio del problema, de tal manera que el diseño que este proponiendo, pueda cumplir con los estándares de calidad del software y con los requerimientos del problema.

Los atributos de calidad, son de vital importancia para el proceso en el cual, el arquitecto de software, diseña la arquitectura, estos permiten darle un objetivo a la arquitectura, para que cuando ya esté desarrollado el software, este cumpla con dichos atributos, como pueden ser: disponibilidad, seguridad y confiabilidad del software, entre otros. Todos estos atributos, pueden ser considerados al momento del diseño de la arquitectura, pero siempre hay que considerar, que entre los mismos atributos, se pueden contradecir en ocasiones y tenemos que identificar, las prioridades según los requerimientos y enfocarse más a uno u otro, según su importancia para la solución del problema. A continuación en la siguiente tabla (Tabla 1) se muestran los atributos externos e internos del software según I.S.O 9126.

CALIDAD DEL SOFTWARE INTERNA Y EXTERNA					
FUNCIONALIDAD	FIABILIDAD	USABILIDAD	EFICIENCIA	MANTENIBILIDAD	PORTABILIDAD
ADECUACIÓN	MADUREZ	FACIL COMPRENSIÓN	COMPORTAMIENTO FRENTE AL TIEMPO	FACILIDAD DE ANALISIS	ADAPTABILIDAD
EXACTITUD	TOLERANCIA A FALLAS	FACIL APRENDIZAJE	USO DE RECURSOS	CAPACIDAD PARA CAMBIOS	FACILIDAD DE INSTALACIÓN
INTEROPERATIVIDAD	CAPACIDAD DE RECUPERACION	OPERATIVIDAD	ADHERENCIA A NORMAS	FACILIDAD PARA PRUEBAS	COEXISTENCIA
SEGURIDAD	ADHERENCIA A NORMAS	SOFTWARE ATRACTIVO		ADHERENCIA A NORMAS	FACILIDAD DE REMPLAZO
ADHERENCIA A NORMAS		ADHERENCIA A NORMAS			ADHERENCIA A NORMAS

Tabla 1. Atributos externos e internos del software según I.S.O. 9126

3 Metodología usada

En esta sección se describen los pasos que se dieron para detectar las necesidades y la problemática con la que cuenta el instituto tecnológico superior de Puerto Vallarta, y poder realizar una propuesta de arquitectura, que solucione y satisfaga las necesidades de la institución.

3.1 Investigación documental

Se realizó una investigación documental, que consistió en buscar información referente a arquitecturas de software y sus componentes, principalmente artículos actuales y libros que marquen sus inicios, componentes e importancia para el desarrollo de software y poder así conocer la teoría que dirige esta área de conocimiento.

3.2 Recopilación de la información en el instituto tecnológico superior de Puerto Vallarta

Se recopiló la información necesaria para construir un software, mediante obtención de requerimientos dentro del ITSPV, a través de entrevistas, observación directa y la observación participante, lo cual permitió determinar las necesidades del instituto para la integración de sus sistemas.

3.3 Investigación de trabajos previos

Se realizó una búsqueda de información de trabajos previos, relacionados al tema. El estudio y análisis de estos trabajos, sirvieron de base para determinar las arquitecturas alternativas, que permitieron llevar a cabo esta investigación y/o sirvieron como puntos de referencia para obtener la base de nuestra propuesta.

3.4 Propuesta de la arquitectura

Los resultados de la investigación realizada, sobre el desarrollo de software en el ITSPV, arrojaron problemas de comunicación e integración de los sistemas que existen dentro del ITSPV, lo que provoca que los procesos administrativos dentro del Instituto, sean menos eficientes y/o eficaces.

La arquitectura de software que se propone para el desarrollo de software en el ITSPV, trata de abarcar todos los aspectos, que cumplen con los atributos fundamentales de una buena arquitectura de calidad y que se acopla a las necesidades de la institución.

Los requerimientos no funcionales, detectados como necesarios para el desarrollo de software de la institución son los siguientes: Modularidad, Escalabilidad, Adaptabilidad, Confiabilidad, Robustez, Seguridad.

La arquitectura que se elige para el ITSPV, utiliza una mezcla de elementos arquitecturales como son: patrones, estilos y modelos de referencia; así como desde una arquitectura orientada a objetos hasta una arquitectura N-niveles y se compone de la siguiente forma:

- Capa de Presentación: En esta capa se encuentran los aspectos lógicos del diseño, vistas, y los aspectos relacionados con la interfaz de los usuarios.
- Capa de Servicios (Web Service): aquí se encontrarán todos aquellos servicios que deban ser compartidos de forma distribuida al igual que los servicios web, incluye interfaces para los servicios para comunicarse con los consumidores. tales como la capa de presentación u otros servicios remotos
- Capa de Aplicación: En esta capa se encuentran los módulos que realizarán funcionalidades completas y las ofrecen como servicios a la capa de presentación que trabajan en la arquitectura, teniendo bien definido los tipos de aplicaciones que se podrán soportar: aplicaciones de escritorio como son las aplicaciones web, aplicaciones de servicios, aplicaciones RIA (Rich Internet Applications) y aplicaciones móviles, pero sin realizar acciones que pertenecen a otra capa.
- Capa de Dominio: Esta capa es el motor principal del software, contiene y es responsable de los conceptos de negocio, información sobre la situación de los procesos de negocio y las reglas del dominio.
- Capa de Persistencia: En esta capa se administran los accesos a datos y las diversas tecnologías relacionadas, encontrando los componentes de datos que proporcionan acceso a datos, que están hospedados en nuestro sistema.
- Capa Transversal: En esta capa se encontrarán todas aquellas funcionalidades que son utilizadas en común en las demás capas, funcionalidades tales como autentificación, autorización gestión de excepciones, validaciones, etc. Estas funcionalidades están disponibles para que puedan ser reutilizadas.

Estos elementos se encuentran distribuidos en las siguientes capas como lo muestra la siguiente figura (Figura2):

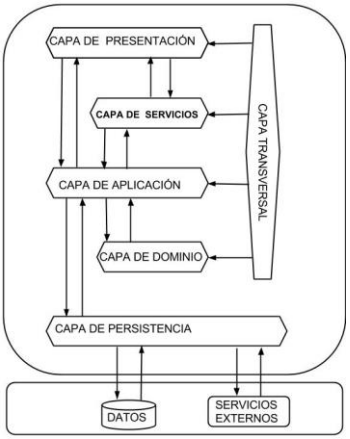


Fig. 2. Arquitectura de Software Propuesta para la Integración de los Sistemas en el Instituto Tecnológico Superior de Puerto Vallarta

Se considero que esta arquitectura, abarca todos los requerimientos no funcionales, para el desarrollo de software en el Instituto Tecnológico Superior de Puerto Vallarta, para el desarrollo actual y futuro, de sistemas que mantengan la integración y comunicación con otros sistemas, así como una aplicación de conceptos y soluciones genéricos como son los estilos arquitecturales, patrones y modelos de referencia

Aunado a esta organización de capas módulos y componentes también se establecen los elementos que deben documentarse de la arquitectura

1. Estructura de Proyecto por Módulos
2. Mapa de tecnologías
3. Estilos arquitecturales
4. Patrones de diseño
5. Framework a utilizar y la recomendación de uso.
6. Capas recomendadas.
7. Reglas de codificación.
8. Lineamientos de uso.
9. Seguridad y manejo de transacciones.

4 Conclusiones y trabajos futuros

Esta propuesta para el ITSPV es un modelo de solución, genérico, basada en la aplicación de conceptos y modelos, así como de la mejores prácticas existentes en el área de arquitecturas de software. Así pues se establece que en la implementación real deberán realizarse los ajustes necesarios, para adecuar el modelo de solución a las particularidades de cada software que se implemente basado en el.

De igual manera se debe contemplar que para asegurar la integración, los sistemas deberán implementar aspectos con tecnologías comunes, como los servicios web, para ofrecer y consumir servicios entre los diversos sistemas. También se podrá considerar

distribuir y/o centralizar componentes entre los sistemas, para lograr una homogeneidad más palpable, por ejemplo la autentificación es un componente común que podría centralizarse.

Agradecimientos.

Agradecemos de antemano a nuestros revisores de la UAG y al Instituto Tecnológico Superior de Puerto Vallarta por las facilidades otorgadas para llevar a cabo nuestra investigación.

Referencias

[1] Software Architecture in Practice. Front Cover · Len Bass, Paul Clements, Rick Kazman. Addison-Wesley Professional, 2003.
[2] Guía de Arquitectura N-Capas orientada al Dominio con .NET. 4.0. (Beta). César de la Torre Llorente. Unai Zorrilla Castro. Miguel Angel Ramos Barros.
[3] Fundamental Concepts for the Software Quality Engineer, Volumen 2 editado por Taz Daughtrey,Sue Carro.
[4] Pattern-oriented software architecture – A system of patterns. John Wiley & Sons. Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad y Michael Stal, 1996.
[5] Garlan, D., Shaw, M. An Introduction to Software Architecture. Advances in Software Engineering and Knowledge Engineering, vol. 1. Ed. V. Ambriola and G. Tortora, World Scientific Publishing Company, New Jersey. 1993.
[6] Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides. Design Patterns: Elements of reusable object-oriented software. Reading, Addison-Wesley, 1995.
[7] Recommended Practice for Software Architecture Descriptions of Software Intensive Systems. IEEE Presss. 2000.

On Project Management Process in Agile Systems Development Methodologies and the ISO/IEC 29110 Standard (Entry Profile)

Sergio Galván-Cruz[1], Manuel Mora[2], Rory O'Connor[3], Francisco Acosta-Escalante[4] and Francisco Alvarez[5]
[1] Universidad Autónoma de Aguascalientes- Av. Universidad No. 940, Ciudad Universitaria, C.P. 20131, Aguascalientes, Ags. México
checogc23@gmail.com
[2] Universidad Autónoma de Aguascalientes- Av. Universidad No. 940, Ciudad Universitaria, C.P. 20131, Aguascalientes, Ags. México
mmora@correo.uaa.mx
[3]Dublin City University, Glasnevin, Dublin 9, Ireland
roconnor@computing.dcu.ie
[4]Universidad Juárez Autónoma de Tabasco – Av. Universidad s/n, Zona de la Cultura, Col. Magisterial, C.P. 86040, Villa Hermosa, Tabasco, México.
francisco.acosta@ujat.mx
[5]Universidad Autónoma de Aguascalientes- Av. Universidad No. 940, Ciudad Universitaria, C.P. 20131, Aguascalientes, Ags. México
fjalvar@correo.uaa.mx

Abstract. The use of ISO/IEC systems and software engineering standards is recommended for software development organizations. In particular, in 2011, a new software process standard was released for Very Small Entities (VSEs): ISO/IEC 29110. Furthermore, the Agile-based System Development Methodologies (SCRUM, XP, Crystal, among others) have also gained interest by organizations. In this - in progress- research, we present an overview of the ISO/IEC 29110 standard, as well as of the Agile-based SDMs, and propose that such SDMs can be enhanced with recommendations from the ISO/IEC standard. In particular, we focus on the Project Management process – one of the two essential processes in the ISO/IEC 29110 standard – and its potential support through a Deployment Package (DP). A DP can be considered an electronic process guideline, and it is attempted to facilitate the implementation of the standard in a VSE.

Keywords: ISO/IEC 29110, Agile Methodologies, VSEs, Deployment Package, EPG, Project Management

1 Introduction

For many small and very small software development companies, implementing properly controls and methods for managing their software development is a big challenge. All software companies are not similar and they change according with many factors such

as: size, market sector, time in business, management style, location, and type of provided services and products. This fact, company diversity, remarks important points for those who develop software process and process improvement models or standards [7]. At present time, software quality is a key variable to competitive advantage, and the use of ISO/IEC systems and software engineering standards are becoming important for many software development organizations. However, most of these companies do not have the technical expertise or the required resources (e.g. sufficient number of employees, financial ones, and allowable time by customers) for implementing them. Additionally, it has been reported that small and very small software development enterprises reject its implementation (of international standards) by the negative perception on that they have been developed for large companies [7]. Consequently, small and very small software development companies lose all benefits that software process international standards or models can provide them, and lately their customers and this market also suffers of a high variability in the quality, costs and time core metrics of their software projects.

Nowadays, many countries distinguish the importance of the VSEs for their economies [6]. The lack of delivering a quality product on time and within budget threats –thus-, the competitiveness of these types of small or very small companies, and finally affects economic parameters. Several international standards or models like ISO/IEC 12207 and CMMI have been developed to collect the best software engineering and managerial practices, but they were not developed for small or very small businesses, so their implementation is inhibited in VSEs. Consequently, despite the potential initial interest in implementing well-defined processes (and supported by international standards or models) in VSEs, it has also identified a high resistance to adapt such standards or models to the needs of business and demonstrate its application in practice [7]. Thus, VSEs have negative perceptions about such software process models or standards by negative views of required additional cost, documentation and bureaucracy issues.

Fortunately, in 2011, a new software process standard was released for VSEs: the ISO/IEC 29110. This standard is formed by two types of processes: Project Management and Software Implementation. It should be noted that ISO/IEC 29110 standard defines the minimum activities and work products that require VSEs to perform [9].

In particular for Mexico country, the effects of globalization demand a better management of the business processes of an organization. Therefore, Information and Communications Technologies (ICT) services in Mexico have increased exponentially in recent years according to ProMéxico [13]. The ICT services market reached 793,000 million dollars worldwide in 2010, while in Mexico the same item reached an amount of 3,988 million dollars in the same period [13]. According to the same study, Mexico is the best destination in Latin America to establish IT companies. Mexico is ranked 6th behind only from India, China, Malaysia, Egypt and Indonesia countries [13]. Thus, the opportunities and needs are wide in Mexico to robust the software development industry.

2 State of the Art

2.1 ISO/IEC 29110 Life Cycle Software Development for VSE (Very Small Entities) Standard.

In January 2011 a new standard for Life Cycle Software Development for VSE (Very Small Entities) was released internationally: ISO/IEC 29110. This standard provides a lightweight process model developed for organizations classified as very small entities (VSEs employs from 1 to 25 people) [9]. ISO/IEC 29110 provides a standard according to

VSEs characteristics and needs. The ISO/IEC 29110 has two main categories of processes: Project Management and Software Implementation (see Fig. 1, adapted from [11]).

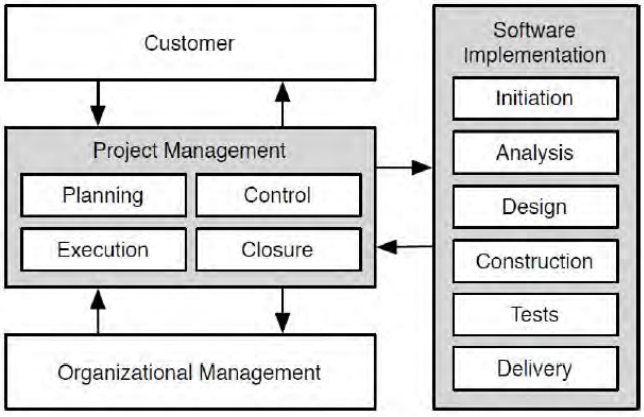


Fig. 1. Structure of the ISO/IEC 29110 Standard [11].

The main features of the two sides of previous model are as follows:

- Project Management (PM) aims to establish and carry out the tasks of the software implementation, which will fulfill the objectives of the project according to quality, time and expected costs. PM includes four activities: ***planning, control, execution and closure***.
- Software Implementation (SI) aims to systematically analyze, design, construction, integration and testing of software products processed according to specified requirements. SI includes six activities: ***initiation, analysis, design, construction, tests and delivery.***

The ISO/IEC 29110 standard provides several support documents of the software lifecycle, guidelines standards and technical reports for VSEs [11]. These documents are based on a subset of items which are known as profiles VSEs. The purpose of a profile is the definition of a subset of best practices from relevant international standards in the context of VSEs [11]. At present, there have been proposed four profiles: ***entry, basic, intermediate and advanced***, but only the first two have been released. In this doctoral research, we will focus in the first "***entry profile***" [5].

It can be identified slight differences between the two first profiles of ISO/IEC 29110 (entry and basic) in three issues: **benefits, entry conditions,** and **rigor of management project process**. According to ISO/IEC 29110-5-1-1 Entry Profile and ISO/IEC 29110-5-1-2 Basic Profile documents, fewer benefits can be reached with the Entry Profile than with the Basic Profile, and more entry conditions and project management rigor are asked in the Basic Profile than in the Entry Profile [4].

2.2 Agile-based Systems Development Approaches

Agile-based systems development approaches contrast strongly to traditional plan-based approach in software engineering practices [3]. These methods recently are emerged as a new and different way of developing software as compared with others "more

traditional" ones[2]. However, the success of these agile methods cannot be reached directly without a set of required factors such as: organizational development strategy, a well-defined process, and a competent development team (implied by an adequate training on the used agile method). [2]. In 2001 the "agile" objective was to determine the value and principles that should allow the team to develop software quickly and responding to changes that arise in project [1]. These new methods were considered as an alternative to traditional development software processes, characterized as rigid and bureaucratic [1].

Several agile methods have been proposed. These are: SCRUM, XP, and Crystal Methodology, among others [3]. According to [10], there has been a growing positive trend of utilization of agile methods by software development companies in the last 5 years. Also, more organizations are adapting their software development processes to agile conventions and adopting them in many workplaces. Some works teams are combining agile and non-agile techniques and practices to create a hybrid methodology. According with some companies who have successfully adopted agile methods, the benefits are well worth the effort.

2.3 Deployment Packages

"A ***deployment package is a set of artifacts developed to facilitate the implementation of a set of practices, of the selected framework, in a VSE***". However, a Deployment Package is not a process reference model [6]. A Deployment Package must contain the following elements: process description (activities, inputs, outputs and roles), guides, templates, checklists, examples, presentation materials, references and mapping to standards and models, and list of tools. Additionally this research [6] indicates that "***packages are designed such that a VSE can implement its content, without having to implement the complete framework at the same time***". It implies that a Deployment Package for the ISO/IEC 29110 does not need to include all of its elements, but rather it is a customized true and fair interpretation of the standard [6]. Fig. 2 displays the general content structure suggested for a DP for the ISO/IEC 29110.

1. Technical Description
   Purpose of this document
   Why this Topic is important
2. Definitions (Generic and Specific Definitions)
3. Relationships with ISO/IEC 29110
4. Detailed Description of Processes, Activities, Tasks, Steps, Roles and Products
   Role Description
   Product Description
   Artefact Description
5. Templates
6. Examples
7. Checklists
8. Tools
9. Reference to Other Standards and Models (ISO/IEC 12207, ISO 9001, CMMI for Development)
10. References
11. Deployment Package Evaluation Form

Fig. 2. Content official of a Deployment Package [6]

Thus, a Deployment Package can serve us as guidelines and support implementation documents to conduct pilot projects in VSEs [5].

2.4 VSEs (Very Small Entities).

A VSE is defined like an enterprise or project having upo to 25 people. The VSEs has great value due to the contribution of services and/or products which are used in large systems and hence the quality of software is required [9]. The Organization for Economic Cooperation and Development (OECD) produced a report in 2005 in which Small and Medium Enterprises (SMEs) are key part of business organizations of all countries of the world, being from the 95% and 99% of the total depending on the country [12].

Also, in July 2005, a survey was applied to 68 managers of companies in the software industry in Mexico. The results of this study revealed that 54.41% of the Mexican software industry are micro size (1-10 employees). For other side, the same study showed that 47% of that enterprises are emergent. In this case (our protocol and study) considers VSEs with a maximum of 5 participants by project [8]. Furthermore, the VSEs have differences compared with a large business [6]. In Fig.3, these characteristics are reported:

Characteristic	Small firm	Large firm
Planning orientation	Unstructured/operational	Structured/strategic
Flexibility	High	Structured/strategic
Risk orientation	High	Medium
Managerial process	Informal	Low
Learning and knowledge absorption capacity	Limited	High
Impact of negative market effects	More profound	More manageable
Competitive advantage	Human capital centered	Organizational capital centered

Fig. 3. Differences between Small and Large Companies [6].

3 Research Plan and Methodology

The research objective – of this in progress doctoral research - is to design and empirically validate an ISO/IEC 29110 Deployment Package (for Project Management Processes, Entry Profile) based on Agile-based system development approaches. Four research questions and null research hypotheses have been stated as follows:

1. ¿ What should be the detailed structure of a framework for assessing the compliance to the standard ISO/IEC 29110 (Project Management, Entry Profile) by agile software development methodologies that be accepted as theoretically valid by a panel of PhD experts ?
   a. H0. There is not a valid theoretically framework for this aim.

2. ¿What is the level of compliance to the ISO/IEC 29110 (Project Management, Entry Profile) for the best 3 agile software development methodologies proposed in the literature standard software?
   a. H0. The level of compliance to the standard is very low (1) or low (2) (in a 5-point scale from 1 (very low) to 5 (very high).

3. ¿What are the structural changes and additions required to the selected agile software development methodology, in order to fit the ISO/IEC 29110 standard (Project Management, Entry Profile)?
   a. H0. None structural changes or additions are required for this aim.

4. ¿What are the perceived values of usefulness, ease of use, compatibility, value, normative beliefs, and intention of use, on the elaborated ISO/IEC 29110 Deployment Package (Project Management, Entry Profile) on the selected agile software development methodology, by a pilot sample of international professionals and academics in Software Engineering?
   a. H0. The perceived values of usefulness, ease of use, compatibility, value, normative beliefs, and intention of use will be less than or equal to 3.0 on a Likert scale of 1 (low) to 5 (very high).

The research method can be defined as Conceptual Design with Proof of Concept [14, 15]. Its four phases are: (i) Formulation of Research Problem; (ii) Analysis of Related Work; (iii) Development of the Conceptual Design; and (iv) Conceptual Design Validation (empirical proof of concept). In phase (i), the background and history of the problem, the problematic situation, the type and purpose of research, the relevance, and the objectives, questions and hypotheses are reported. In phase (ii), analysis of related work, theories bases, related studies, contributions and limitations of related studies, selection/design of general conceptual framework, selection/design model of research are reported. In phase (iii), application or conceptual model or design (deployment package), application or creative-deductive relational conceptual design model are reported. In phase (iv), conceptual model validation implemented or design, validation of content by panel of experts, validation by logical argument, validation for proof of concept construction artifact are reported.

4 Conclusions and Directions for Future Research

This research is focused to study a recent and very important problem for the VSEs. It can be considered a relevant research problem according with the reviewed literature. This study will focus on the design and validation of a Project Management Deployment Package considering the ISO/IEC 29110 (entry profile) official recommendations for enhancing the Project Management process used in Agile Methodologies like: SCRUM, XP, and Crystal. The final product (the DP) will be empirically validated through a pilot sample of software engineers professionals from several countries (Mexico, Ireland, and USA). Hence, it is important to note that our specific research topic has been little investigated. However, it is essential for VSEs because it generally accepted that organizations that use better software development processes will be more competitive. We believe that an **ISO/IEC 29110 *Deployment Package*** can help to improve the Agile Software Development approaches –in particular the Project Management processes-.

References

[1] Canós, J., Letelier, P., & Penadés, M. (2004). Métodologías Ágiles en el Desarrollo de Software. *Conference on eXtreme Programming and Agile Processes in Sotware Engineering*.

[2] Chow, T., & Cao, D.-B. (2008). A survey study of critical success factors in agile software projects. *The Journal of Systems and Software*, 961-971.

[3] Dyba, T., & Dingsoyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology 50*, 833-859.

[4] ISO/IEC 2012. (2012). *ISO/IEC TR 29110-5-1-1*. ISO.

[5] Laporte, C., & O'Connor, R. (2010). *The Development and Experimentation of an International Standard for Very Small Entities Involved in Software Development.* Montreal, Canada: The Irish Software Engineering Research Centre.

[6] Laporte, C., Alexandre, S., & O'Connor, R. (2008). A Software Engineering Lifecycle Standard for very Small Enterprises. *Springer-Verlag Berlín Heidelberg*, 129-141.

[7] O'Connor, R., & Laporte, C. (2011). Using ISO/IEC 29110 to Harness Process Improvement in Very Small Entities. *18th European Software Process Improvement Conference*.

[8] SG. (26 de Septiembre de 2007). *sg.com.mx*. Obtenido de sg.com.mx: www.sg.com.mx

[9] Takeuchi, M., Kohtake, N., Shirasaka, S., Koishi, Y., & Shioya, K. (2013). Report on an assesment experience based on ISO/IEC 29110. *Software Research Associates*.
[10] West, D., & Grant, T. (2010). Agile Development: Mainstream Adoption Has Changed Agility. *Forrester*.
[11] Varkoi, T. (2010). *Process Assesment In Very Small Entities.* Finlandia: ISO.
[12] OCDE (2005). *Centre for Entrepreneurship, SMEs and Local Development.* OCDE.
[13]Secretaría de Economía. (2010). *ProMéxico*. Recuperado el 2013, de ProMéxico: www.promexico.gob.mx
[14] Glass, R., Vessey, I., & Ramesh, V. (2002). Research in software engineering: an analysis of the literature. *Elsevier Science*, 491-506.
[15] Mora, M., Gelman, O., Paradice, D., & Cervantes, F. (2008, May). The case for conceptual research in information systems. In *CONF-IRM 2008 Proceedings* (p. 52).

Diseño de una Aplicación Interactiva para el Aprendizaje de la Lecto-Escritura desde un Enfoque de Trabajo Colaborativo

Ingeniería de Software

Liliana Rodríguez-Vizzuett[1], Francisco J. Álvarez- Rodríguez[1], Jaime Muñoz-Artega[1], Josefina Guerrero-García[2], David Céspedes-Hernández[1]
{lilianarv90,jmauaa,fjalvar.uaa,dcespedesh@gmail.com}, jguerrero@cs.buap.mx,

[1] Universidad Autónoma de Aguascalientes, Av. Universidad 940, Aguascalientes, Aguascalientes, 20131. México

[2] Benemérita Universidad Autónoma de Puebla, Av. 14 Sur, esquina con San Claudio, Puebla, Puebla, 72592. México

Resumen. La lectura y la escritura son actividades de gran importancia ya que permiten la adquisición de otros conocimientos que serán utilizados a lo largo de la formación como persona. El uso de la tecnología en la educación ha ido modificando el sistema de enseñanza tradicional de manera considerable pues ayuda a los docentes brindándoles herramientas para que al utilizarlas se puedan reforzar los conocimientos que se dan en un salón de clases. El presente trabajo tiene como finalidad describir el proceso de diseño de una aplicación interactiva para el aprendizaje de la lecto-escritura desde un enfoque de trabajo colaborativo tomando en cuenta el documento oficial de competencias que la Secretaría de Educación Pública en México (SEP) ofrece a los docentes para la educación a nivel preescolar.

Keywords: Aprendizaje colaborativo, aplicaciones interactivas, TIC, competencia, lectoescritura.

1 Introducción

La educación preescolar debe ofrecer a los niños la oportunidad de desarrollar su creatividad, reforzar la confianza en sus capacidades, estimular su curiosidad y efectuar el trabajo en grupo para propósitos establecidos [1]. Un factor importante que se debe de considerar es la gran ayuda que las Tecnologías de la Información y la Comunicación (TIC) ofrecen pues proveen a los estudiantes gran cantidad de herramientas para que sean utilizadas dentro del aula y mejoren la forma en la que adquieren conocimientos[2].

Un tema importante dentro del desarrollo de los niños es la lectura y la escritura, pues si no cuentan con estas habilidades resultará difícil que logren un correcto desarrollo personal. La inclusión de la tecnología en la educación debe de considerarse sólo como una herramienta de apoyo, pues no viene a sustituir al maestro, sino pretende brindarle ayuda para que el estudiante tenga más elementos tanto visuales como auditivos para poder enriquecer el proceso de enseñanza-aprendizaje. Existen diversas aplicaciones interactivas que funcionan como apoyo para reforzar el aprendizaje de la lecto-escritura, sin embargo,

estas no reportan haberse desarrollado bajo alguna metodología de diseño que haga que no sólo entretengan a los niños sino que también les ayude para adquirir conocimientos [3]. Existen igualmente juegos que ayudan a reforzar el aprendizaje, sin embargo no todos atienden al método lúdico, el cuál no es sólo un espacio de juego en el que los niños resuelven sus necesidades recreativas, sino un elemento importante en el contexto escolar, en función de una pedagogía creativa, que ayuda a la formación integral del ser humano [4], es decir, estos juegos deben de verse como una oportunidad de enseñanza en la cual los niños desarrollen competencias y no sólo como un medio de entretenimiento entre actividades.

La SEP establece en documentos oficiales que se les brindan a las educadoras de nivel preescolar, las competencias que se deben satisfacer dentro de la educación preescolar, para que al concluir este periodo escolar los estudiantes hayan iniciado un proceso de contacto formal con el lenguaje escrito, por medio de la exploración de textos con diferentes características (libros, periódicos e instructivos, entre otros) [5]. Así mismo, para que se desarrolle el interés y gusto por la lectura, que hagan uso de diversos tipos de texto y sepan para qué sirven; se inicien en la práctica de la escritura al expresar gráficamente las ideas que quieren comunicar y reconozcan algunas propiedades del sistema de escritura [5].

Con la finalidad de desarrollar competencias de lectoescritura en los niños y considerando las ventajas que las TIC's ofrecen, se tiene como objetivo para este proyecto la creación e implementación de una aplicación interactiva que pueda utilizarse en una Laptop o bien en un dispositivo móvil para que de manera colaborativa y dentro de un ambiente de trabajo adecuado, los niños puedan desarrollar destrezas en el área de la lectura y la escritura tal como lo señala la SEP [5].

El artículo estará estructurado de la siguiente manera: en la segunda sección se analiza el estado del arte para estar familiarizados con el tema, la tercera sección abarca la metodología que se usó, la cuarta sección está relacionada con los resultados y finalmente se presentan las conclusiones y el trabajo futuro.

2 Estado del arte

En esta sección se encuentran definiciones que resultan de gran ayuda para el desarrollo del presente trabajo. La lúdica es un medio que brinda escenarios de interacción de manera comunicativa donde cada participante tiene un rol en el que debe de seguir reglas y normas que ha aceptado con responsabilidad [6].

Los juegos son recursos que se pueden utilizar en el proceso de enseñanza-aprendizaje para hacer un momento más agradable y participativo, sin embargo, al introducir el juego como un recurso, este debe tener objetivos pedagógicos establecidos para poder así cumplir con la función de enseñar. El uso de juegos dentro de un salón de clases permite a los niños trabajar diferentes habilidades que favorecen un mayor desarrollo cognitivo y una mayor interacción social, generando así un ambiente colaborativo donde exista el respeto, la creatividad, la comunicación y la tolerancia entre niños con diferentes formas de pensar [7].

Así como el juego fomenta un ambiente colaborativo, es importante tener en cuenta una definición sobre el aprendizaje colaborativo que se refiere a la actividad de grupos pequeños, que se puede desarrollar principalmente en un salón de clases, esto con la finalidad de intercambiar puntos de vista y aprender a tolerar diferentes estilos de aprendizaje. Los estudiantes aprenden más cuando utilizan el aprendizaje colaborativo pues tienen una mejor retención del contenido visto y desarrollan habilidades de razonamiento y de pensamiento crítico [8]. En las principales tiendas de aplicaciones para móviles, se pueden encontrar juegos que las educadoras recomiendan y utilizan como refuerzo de los

conceptos que enseñan. En el caso de la lecto-escritura, se cuenta con herramientas como "El Abecedario" [9], la cual permite a los niños que la utilizan, familiarizarse con el alfabeto proporcionándole recursos multimedia asociados a las diferentes letras. Cabe destacar que si bien esta aplicación cuenta con buenas evaluaciones y con contenidos adecuados para la SEP, no se reporta que sus desarrolladores hayan seguido un proceso de desarrollo formal ni se dispone de modelos o módulos que permitan la reutilización y extensión para la adquisición de otras competencias diversas como es el fin de este proyecto. La Fig 1 muestra la interfaz gráfica de "El Abecedario".

Fig 1. Interfaz gráfica de "El Abecedario" de [9].

El aprendizaje del lenguaje escrito consiste en adquirir un sistema determinado de símbolos y signos cuyo dominio marca un momento muy importante dentro del desarrollo cultural de un niño [10]. La lectura es una actividad intelectual en la que intervienen aspectos como la percepción visual, auditiva y mental para la comprensión de lo que se lee. Según D.B. Elkonin, "*la lectura se define como el proceso de reproducción de la forma sonora de las palabras, siguiendo sus modelos gráficos*" [11]. Para Vygotsky el lenguaje y la escritura son procesos de desarrollo para que el ser humano pueda dominar los medios externos dentro de su desarrollo cultural y del pensamiento [10].

3 Metodología Usada

Para el desarrollo de este proyecto se utilizó el marco de trabajo CAMELEON [12] que permite el diseño de aplicaciones altamente usables y que resulten atractivas a los usuarios potenciales que en este caso serían los niños en edad preescolar. La finalidad de la aplicación interactiva que se desarrollo es que mejore el proceso de adquisición de conocimientos y por tanto se utilizó el documento de la SEP [13] que especifica las competencias que se deben satisfacer junto con sus aprendizajes esperados: "*Construyen el significado de la escritura y su utilidad para comunicar. Comienzan el trazo de letras hasta lograr escribir su nombre*". La competencia de la SEP que se busca satisfacer con el desarrollo de la presente aplicación es: "*Compara las características gráficas de su nombre con los nombres de sus compañeros y otras palabras escritas. Intercambia ideas acerca de la escritura de una palabra. Reconoce las diferentes letras y distingue entre los sonidos que estas emiten"* [5].

Una vez que se tiene este objetivo, se procede a utilizar el marco de trabajo CAMELEON [12] que sugiere al diseñador la definición y ejecución de 4 etapas de desarrollo, las cuales son: 1. Creación de tareas y conceptos que describan acciones que tiene que realizar el usuario; 2. Diseño de interfaz de Usuario Abstracta (AUI), este tipo de interfaces son ajenas a la modalidad y a la plataforma, ya que sólo describen contenedores abstractos y componentes individuales de interacción; 3. Creación de interfaz de Usuario Concreta (CUI), describen la modalidad con la que el usuario interactuará el producto final, y por último; 4. Diseño de interfaz de Usuario Final (FUI), corresponde al diseño de la

interfaz que será implementada para una plataforma determinada [12]. Después de esto, se procede a realizar el diseño de la aplicación en términos del marco de trabajo, por tanto, utilizando la herramienta CTTE [14] se elabora el modelo de tareas que se muestra en las Figuras de la 2 a la 5.

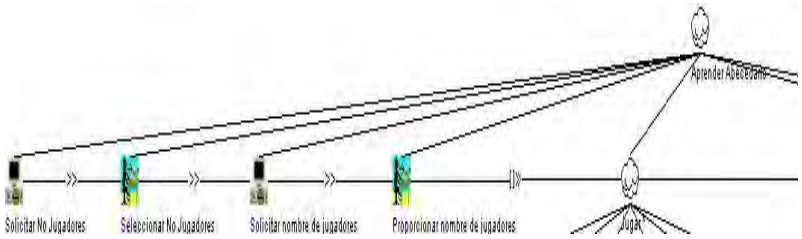


Fig 2. Modelo de tareas de aprender abecedario.

En la Fig 2, la tarea principal es la de *Aprender el abecedario* y dentro de esta tarea se tienen tareas interactivas y de sistema, la primera tarea de sistema es *Solicitar el número de jugadores*, después se ejecuta una tarea interactiva que es *Seleccionar el número de jugadores*. Una vez que se hace esto, la aplicación solicita a los jugadores el nombre de cada uno de ellos, esta actividad se debe de realizar con ayuda de un facilitador pues en algunas ocasiones los usuarios no saben aún leer ni identifican de manera correcta las letras. Ya que esta tarea se llevo a cabo, la siguiente a ejecutarse es la de *Jugar*. Después de esta tarea, sigue la de *Evaluar*, en donde el usuario tendrá que hacer una pequeña actividad para verificar si se aprendió o no lo jugado. Cuando esto finaliza, el sistema muestra la pantalla principal tal como se muestra en la Fig 3.

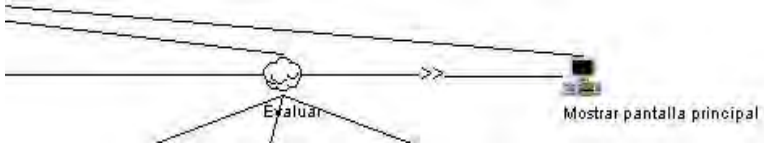


Fig 3. Modelo de tareas de aprender abecedario.

En la Fig 4 se muestra el desglose de la tarea *Jugar*, la cual tiene como sub tareas la de *Mostrar Turno* en donde se verá el nombre del jugador que debe realizar la actividad, posteriormente se mostrarán las 27 letras del abecedario (*Mostrar Letra*), en la tarea de *Arrastrar Letra*, los niños por turnos deberán ir colocando las letras en el lugar que corresponden, la aplicación las validará y les mostrará si lo hicieron bien o no, de no haberlo realizado de manera correcta se le mostrará un mensaje de error.

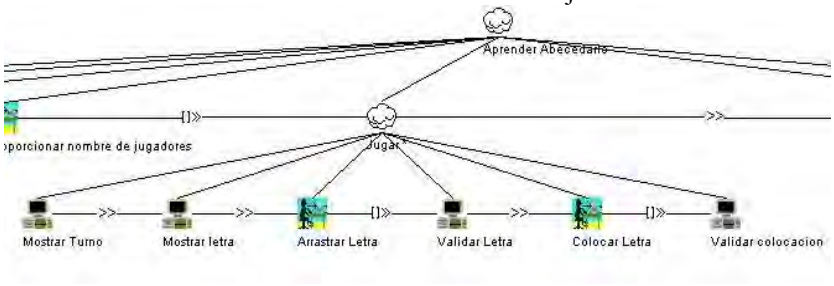


Fig 4. Modelo de tareas de aprender abecedario, tarea jugar.

Dentro de la Fig 5 se encuentra la tarea *Evaluar*, en ésta se presentarán preguntas para evaluar si los usuarios adquirieron el conocimiento esperado o no. Para esto, el sistema mostrará la pregunta junto con un audio asociado, se le mostrarán las respuestas, posteriormente el niño deberá seleccionar la respuesta, esa respuesta se validará y

posteriormente se reproducirá un sonido dependiendo de si es correcta o no. Finalizadas las preguntas, la actividad de evaluación acaba e inmediatamente la aplicación muestra la pantalla principal.

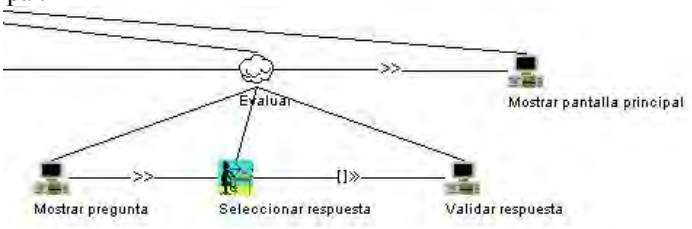


Fig 5. Modelo de tareas de aprender abecedario, tarea evaluar.

Una vez que los modelos de tareas han sido elaborados, siguiendo el marco de trabajo CAMELEON, se realizan las interfaces abstractas. En la Fig 6 se muestra una AUI conformada por un contenedor principal que corresponde a la ventana de jugar, este se compone de dos contenedores, en uno de ellos estarán las letras del abecedario y en la parte de arriba de este contenedor, se mostrará la letra que los niños tienen que ir arrastrando hacia el otro, una vez que todos los niños jueguen, deberán ir a la sección de evaluación dentro de la interfaz principal de abecedario.

Fig 6. Interfaz abstracta de 'Jugar'.

La siguiente etapa en el desarrollo es la elaboración de las interfaces concretas de la aplicación. En la Fig 7 se muestra la CUI correspondiente a la tarea de jugar en la aplicación de abecedario, ésta ya toma en cuenta la modalidad de interacción pero aún no toma en cuenta la plataforma de uso, de la misma forma en como se definió en los modelos previos, el usuario tiene elementos suficientes para, de manera colaborativa, colocar cada letra en su lugar hasta finalizar el abecedario.

Fig 7. Interfaz concreta de 'Jugar'.

Finalmente, se realizan las FUI que son las pantallas que se mostrarán ya en la aplicación. En la Fig 8 se muestra la FUI para la aplicación desarrollada, que como ya se

mencionó anteriormente los niños deberán jugar de manera colaborativa para ir identificando las letras del abecedario hasta que logren su comprensión total.

4 Resultados

Como resultados se obtuvieron documentos con los requerimientos de la aplicación, un documento de modelo de tareas y conceptos, especificando cada actividad que se realiza, cuatro interfaces abstractas con su descripción y funcionalidad, cuatro CUI con su descripción y por último seis FUI con su descripción, todo esto en un documento de diseño, se elaboró también un documento que contiene todas las tareas que la aplicación ejecuta, esto en forma de tabla describiendo la tarea, la naturaleza de la tarea, el número de tarea que es y su antecesor en caso de que existiera. Así mismo se obtuvo la codificación total de la aplicación junto con un documento de construcción de software.

Fig 8. Interfaz final de 'Jugar'.

Por último se realizaron pruebas en dos diferentes instituciones en la Ciudad de Aguascalientes, México, Centro Educativo Termápolis y el CENDI Girasol, la prueba que se realizó fue para comprobar usabilidad y funcionalidad de la aplicación, usando la técnica de observación [15], ya que por sus edades, los niños, no pueden responder cuestionarios. Los resultados obtenidos son cualitativos y se muestran en la Tabla 1.

Tabla 1. Resultados de la evaluación de usabilidad y funcionalidad.

Características	Grupo 1	Grupo 2	Grupo 3
Escuela	Centro Educativo Termápolis	CENDI Girasol	CENDI Girasol
Edad	5 años	4-5 años	5-6 años
Número de alumnos	9	7	7
Descripción del grupo	Niños de 5 años provenientes de diferentes escuelas que se conocen e integran a equipos de trabajo ese mismo día.	Niños de 4 a 5 años que provienen de la misma escuela, un grupo ya integrado iniciados en las competencias de lecto-escritura	Niños de 5 a 6 años que provienen de la misma escuela, un grupo ya integrado iniciados en las competencias de lecto-escritura
Observaciones de la interacción con las herramientas	En su aplicación los niños tuvieron la oportunidad de interactuar con la aplicación de abecedario, inicialmente los aplicadores tuvieron que intervenir un poco para poder organizar los tiempos de trabajo, posteriormente al recibir las instrucciones pudieron interactuar cómodamente.	Se establecieron módulos y tiempos definidos para la interacción con cada parte de la herramienta. Los niños requerían instrucciones para realizar la actividad, trabajaban de manera colaborativa y respetuosa.	Se establecieron módulos y tiempos definidos para la interacción con cada parte de la herramienta. Les fue muy sencillo interactuar con la herramienta y sus diferentes módulos, quizá el grado de dificultad para este grupo fue mínimo.

5 Conclusiones y trabajo futuro

Durante la realización de este proyecto se llevó a cabo el modelado, diseño, programación y evaluación de una aplicación interactiva con un componente colaborativo para niños de preescolar, la cual, busca la creación de ambientes de aprendizaje que permitan satisfacer las competencias de lenguaje y comunicación que se encuentran en [13], permitiéndole al docente conocer nuevas herramientas de apoyo para el aprendizaje, de manera que para los niños resulte más atractiva e interesante, brindando además al docente la obtención de resultados sobre el aprendizaje de los niños. Con los resultados obtenidos en la evaluación, se puede llegar a la conclusión de que la aplicación interactiva cumple con el propósito de entretener y de enseñar a los niños a trabajar de manera colaborativa para la adquisición de las competencias de lecto-escritura. En cuanto al diseño y la programación de la aplicación, se obtuvieron documentos que permitiran que el trabajo sea replicado para otras actividades interactivas, lo que es de gran ayuda para la creación de un repositorio de aplicaciones colaborativas para satisfacer competencias que deben cubrirse según lo establece la SEP. Queda como trabajo futuro la generación de más aplicaciones colaborativas para la lecto-escritura y realizar más etapas de evaluación para verificar la funcionalidad y la usabilidad de las aplicaciones implementando un estudio cuantitativo.

Agradecimientos. Se agradece el apoyo al programa de Becas Nacionales para el Posgrado de CONACYT y al proyecto Promep en red a cargo del CA de Objetos de Aprendizaje e Ingeniería de Software en la Universidad Autónoma de Aguascalientes (UAA). Igualmente se agradece al Centro Educativo Termápolis y al CENDI Girasol por las facilidades otrogadas para la realización de pruebas. Se reconoce además a Paul Alejandro Partida Mejía de la Universidad Autónoma de Nayarit por su participación en el desarrollo de la aplicación.

Referencias

[1] Chavira, E. B. La Educación Preescolar En México.
[2] Graells, P. M. (2000). Impacto de las TIC en educación: funciones y limitaciones. Departamento de Pedagogía Aplicada, Facultad de Educación, UAB.
[3] Proyecto H@z TIC. Guía práctica de aprendizaje digital de lectroescritura mediante tablet para alumnos con síndrome de Down, Ministerio de Educación, Cultura y Deportes, 2012.
[4] Martínez L., Lúdica como estrategia didáctica. Revista Escholarum UAG (2008).
[5] Graells, P. M. (2000). Impacto de las TIC en educación: funciones y limitaciones. Departamento de Pedagogía Aplicada, Facultad de Educación, UAB.
[6] Velásquez Navarro, J. Ambientes Lúdicos de Aprendizaje. Diseño y operación. Ed Trillas. 2008.
[7] Varetta, D. (2006). Lo lúdico en la enseñanza-aprendizaje del léxico: propuesta de juegos para las clases de ELE. redELE: Revista Electrónica de Didáctica ELE, (7), 3.
[8] Millis, Barbara J. Materials presented at The University of Tennessee at Chattanooga Instructional Excellence Retreat, 1996.
[9] Taos Games. (2013). El Abecedario, aplicación disponible en https://play.google.com/store/apps/details?id=air.com.taossoftware.abecedario&hl=es_419.
[10] Vygotsky, L. (1931). La prehistoria del desarrollo del lenguaje escrito. Obras Escogidas, 3.
[11] Elkonin, D. B. (1985). Psicología del juego. Visor.
[12] Calvary, G., Coutaz, J., et al., A Unifying Reference Framework for Multi-Target User Interfaces, Interacting with Computers, 15(3), June 2003, pp. 289–308
[13] Secretaría de Educación Pública. (2011). Programa de Estudios, Guía para la Educadora.
[13] Paternò, F., ConcurTaskTrees: An Engineered Notation for Task Models, in The Handbook of Task Analysis for Human-Computer Interaction. 2003. pp. 483-503.

[14] Nielsen, J. (1994). Usability engineering. Elsevier.

Aplicación con Realidad Aumentada y modelos 3D basada en Patrones de Diseño de Software

Iván Peredo Valderrama [1] Rubén Peredo Valderrama [2] Francisco Castillo Velásquez [1]
[1] Universidad Politécnica de Querétaro, Carretera Estatal 420 S/N el Rosario el Marqués, México, Querétaro, CP. 76240.
ivan.peredo@upq.edu.mx
francisco.castillo@upq.edu.mx
[2] Escuela Superior de Cómputo del Instituto Politécnico Nacional, Av. Juan de Dios Bátiz S/N esquina con Miguel Othón de Mendizábal, México, D.F., 07738. México
rperedo@ipn.mx

Resumen. La Realidad Aumentada (AR) desde hace más de 40 años ha sido tema de discusión, anteriormente era una estrategia tecnológica poco conocida y usada, debido a la complejidad de integración de varias tecnologías, lo cual se había hecho a través de dispositivos especializados. Es una tecnología prometedora misma que permite crear escenarios que son difícilmente visibles en el mundo real, facilitan la motivación de los aprendices y una mejor conceptualización para los estudiantes. La AR combina el mundo virtual con el real, de esta forma se enriquece las experiencias visuales y puede apoyar en mejorar múltiples técnicas de enseñanza/aprendizaje. La propuesta se enfocó en modelar en 3D el centro histórico de la ciudad de Querétaro basada en Patrones de Diseño de Software, e implementar la AR en una aplicación para dispositivos móviles basados en el sistema operativo Android, con el propósito de enriquecer experiencias de aprendizaje educativas.

Palabras clave: Realidad Virtual, Realidad Aumentada, modelado 3D.

1 Introducción

El rápido progreso que ha tenido la tecnología para el desarrollo de dispositivos inteligentes a lo largo de los últimos años, ha permitido a los desarrolladores de software e investigadores entrar a ramas o tópicos que no habían sido tomados en cuenta anteriormente, debido a su grado de complejidad y/o a la dificultad de contar con la tecnología adecuada en ese momento [1]. Los problemas de desarrollo se manifiestan cuando se utiliza una tecnología poco conocida, falta de documentación y en muchos casos asociada a elevados costos de licenciamiento. Pero si se tiene la fortuna de tener acceso libre a esta tecnología y se cuenta con una buena documentación, se logra así desarrollar nuevas y mejores aplicaciones, mejorando la interacción que tienen los usuarios con el entorno que los rodea. La AR es parte de un entorno mixto el cual presenta objetos del mundo real y del mundo virtual dentro de un solo entorno, la Fig. 1 muestra un entorno mixto.

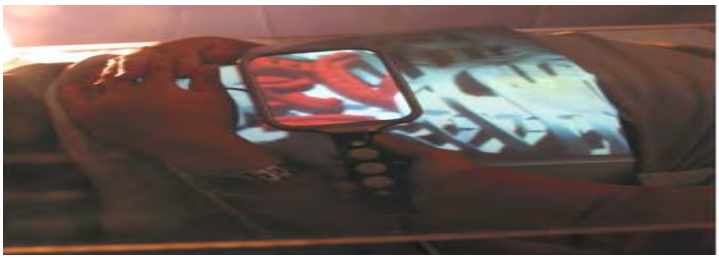


Fig. 7. Entorno mixto (L.D. Brown and H. Hua).

Para el diseño, desarrollo e implementación de una aplicación con AR; se requiere lo siguiente: Cámara Web, monitor de computadora, software y marcadores; dispositivos inteligentes (teléfonos móviles, smartphones, tablets, iPad, etc.), mismos que necesitan contar con aplicaciones previamente instaladas [2].

Esta propuesta presenta una aplicación que permite la manipulación de diferentes objetos 3D utilizando el Motor Unity y la plataforma Vuforia para la implementación de la AR.

2 Estado del Arte

Cuando se habla de AR, lo primero que se viene a la mente es un modelado en 3D que va a ser visualizado en un dispositivo, la Fig. 2 muestra un modelado y animación 3D de la catedral de Querétaro en **Google Sketchup** de la propuesta.

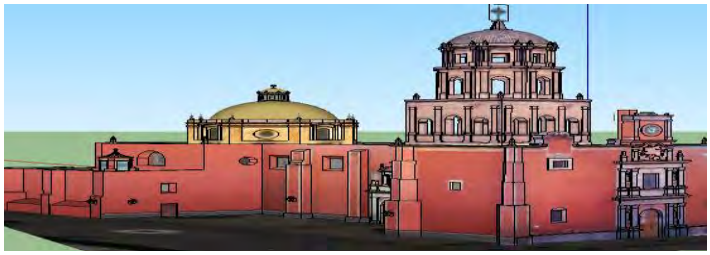


Fig. 8. Modelado y animación 3D de la catedral de Querétaro en Google Sketchup.

Para la construcción de los modelos en 3D, se utilizó el software de modelado **Google Sketchup**, debido a que es un software fácil de manipular, así como contiene herramientas capaces de realizar un modelado de calidad, permitiendo así el uso de texturas, personalización de las mismas, importación y exportación en diferentes formatos manteniendo compatibilidad. En el pasado era casi imposible realizar esto, ya que la tecnología no era lo suficientemente apropiada para diseñar una aplicación de esta magnitud. Actualmente existe diversos tipos de software de modelado en 3D, tales como: **Cinema 4D Studio**, **Maya AutoDesk**, **Blender**, etc. Para implementar la AR en nuestra propuesta e interactuar con estos modelados en 3D, empleamos el Motor Unity y la plataforma Vuforia, la Fig. 3 muestra la combinación de estas dos tecnologías. La idea de la propuesta es aplicar la AR en el sector educativo y turístico, con el objetivo de dar a conocer ciertas zonas de la ciudad de Querétaro con información relevante, brindando

recorridos turísticos virtuales más innovadores y amenos para aquellas personas que no han tenido oportunidad de visitarla.

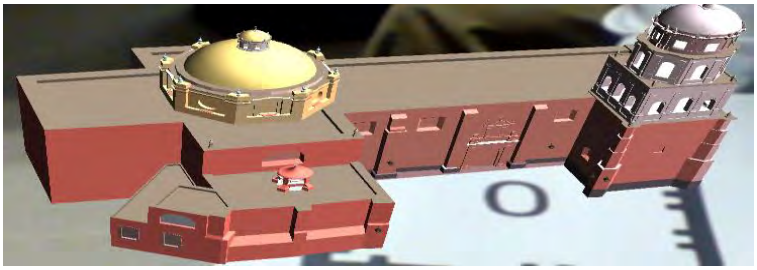


Fig. 3. Modelado 3D en Unity y la plataforma Vuforia de la catedral de Querétaro.

Poco a poco se puede observar que la AR ha ido alcanzado la madurez necesaria para poderla implementar en diversas áreas, y hay lugares como escuelas o industrias que requieren manipular objetos en 3D para enriquecer diversos aspectos de la información que reciben. Un problema importante en la implementación de la AR es el hecho de que requiere demasiadas líneas de código para su codificación, pero utilizando el motor Unity y la plataforma Vuforia, la codificación se reduce de manera significativa.

En la realización de la propuesta se indagaron diversos trabajos vinculados con aplicaciones que emplean la AR, con el propósito de excluir el uso del ratón y teclado para llevar a una forma de interacción más natural con el usuario y la computadora. Algunos de los trabajos revisados son los siguientes:

- **Google Sky Map** [3], aplicación gratuita ideal para el estudio de astronomía, enfocando la cámara del móvil en el cielo, el programa puede identificar estrellas, constelaciones, planetas y cuerpos celestes, ofreciéndonos en vivo los datos. Tiene opciones de búsqueda. Requiere Android 1.6 o superior.
- **Goggles** [4], servicio gratuito sólo para móviles con Android, que automatiza la búsqueda en Internet de objetos reales, usando su código de barras o su ubicación, también brinda reconocimiento de textos para digitalizar tarjetas de presentación.
- **TwittARound** [5], aplicación para el iPhone que permite observar los tweets que se están publicando en tiempo real cerca de la ubicación en la que se encuentra el dispositivo desde el cual se hace la consulta. Twitter quiere darle mucha fuerza a la posibilidad de geolocalización de los Tweets, y este tipo de programas van a resultar fundamentales para esa estrategia.
- **Yelp Monocle** [6]**,** red social que permite al usuario buscar información sobre locales comerciales cercanos a su ubicación. Ideal para viajes de negocios.

Como se aprecia en los diversos trabajos mostrados con anterioridad, cada uno de ellos utiliza la AR de una forma innovadora, así el usuario puede emplear movimientos que le parecen familiares para crear una interacción más natural, como si estuviera tocando objetos reales.

3 Metodología usada

Durante el desarrollo de un proyecto de software los cambios que se presentan a diario son constantes, surgiendo así la necesidad de una constante actualización del proyecto de software. Los diseños de software robustos permiten manipular de mejor manera el cambio en los proyectos de software y esto se logra utilizando patrones de diseño.

3.1 Patrones de Diseño de Software

Los Patrones de Diseño de Software (PDS) proporcionan una representación superior en el análisis y diseño de los proyectos de software, ofreciendo excelentes opciones de soluciones, con el propósito de tratar mejor el cambio a lo largo de la vida de un proyecto de software, maximizando la reutilización de las partes de los proyectos de software. Uno de los peores enemigos que pueden encontrarse los desarrolladores de software es el cambio, sin una arquitectura robusta en los proyectos de software, difícilmente los desarrolladores podrán manejar adecuadamente el proyecto de software a lo largo de su vida.

La aplicación propuesta emplea primordialmente dos PDS que son la base de la presente propuesta: Patrón de Composición, y patrón Modelo-Vista-Controlador (*Model-View-Controller*, MVC por sus siglas en inglés).

3.2 Patrón de Composición

Este permite construir estructuras complejas con base en estructuras más simples, se tienen dos tipos de componentes denominados: indivisibles y complejos, estos últimos son una conjunción de otros componentes. El patrón de Composición posibilito simplificar la API de la aplicación propuesta, permitiendo así manejar de forma indistinta componentes indivisibles y complejos. Permitiendo maximizar la reutilización de los componentes indivisibles y complejos en diferentes contextos de la aplicación.

3.3 Patrón Modelo-Vista-Controlador

Permite conjuntar diferentes patrones en una misma aplicación, facilitando embeber otros patrones en la aplicación, y trabajar de forma conjunta, con el propósito de desarrollar aplicaciones más complejas. El patrón está creado de tres partes fundamentales: Modelo, Vista y Controlador. El Modelo simboliza la base de datos de la aplicación, así como la lógica de negocios, la Vista representa la interfaz hacia el usuario y muestra el estado de la aplicación, por último el Controlador maneja la Vista de entrada del usuario con su correspondiente Vista de salida, y cambia el estado de la aplicación. La flexibilidad del patrón se debe a la dispersión de los tres componentes que lo conforman, sin traslapar sus responsabilidades, permitiendo a cada parte llevar a cabo sus respetivas tareas, colaborando de manera conjunta, los tres componentes del patrón.

Dentro de las diversas aplicaciones realizadas a lo largo de los años se ha optado por utilizar el patrón MVC en varios de los proyectos realizados [7-10], permitiendo obtener arquitecturas más robustas en la implementación de nuestras propuestas de proyectos de software, consiguiendo: Maximizar la reutilización de las partes que conforman el proyecto

de software, y además permite manejar mejor el cambio a lo largo de la vida del proyecto, proporcionando un código fácil de darle mantenimiento. Por los motivos expresados anteriormente en el punto 3.2 y 3.3, la nueva propuesta implementa los patrones fundamentales: MVC, y Composición. La Fig. 4 muestra la arquitectura interna de la propuesta con los dos principales patrones de diseño de software: MVC y composición.

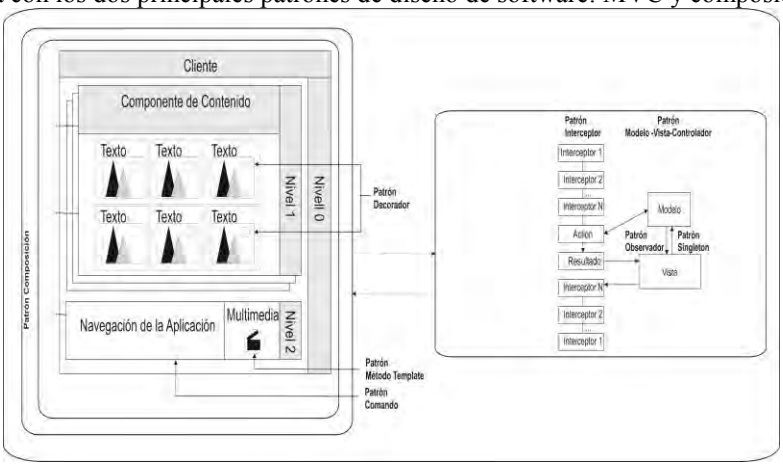


Fig. 4. Arquitectura interna de la propuesta basada en patrones.

3.4 Análisis y Diseño de la Aplicación Basada en Componentes

La propuesta implementa una aplicación con AR de la zona céntrica del estado de Querétaro, la cual permitirá enriquecer la experiencia a los usuarios sobre diversos escenarios que ofrece esta entidad.

Para la realización de la aplicación se utilizaron las siguientes herramientas de software: **Unity** [11], motor que permite crear videojuegos multiplataforma creado por **Unity Technologies. Vuforia** [12], producto de Qualcomm Technologies, Inc., en el desarrollo de la propuesta **Vuforia** es el plug-in que se agregó a **Unity**, y con el cual se implementaron las aplicaciones que permiten visualizar los objetos 3D con AR. **Monodevelop** [13], entorno de desarrollo integrado libre y gratuito, se utilizó para codificar scripts híbridos entre C# y JavaScript, para realizar las acciones necesarias de la propuesta. El **SDK de Android** [14], incluye un conjunto de herramientas de desarrollo para Android. **C#**, lenguaje de programación para **Unity** [15], orientado a objetos.

4 Resultados experimentales

La mayoría de los dispositivos inteligentes permiten desplegar aplicaciones con AR, en donde una cámara capta la información del mundo real y la transmite al software de la propuesta. El programa de software captura los datos reales y los transforma en AR. Los marcadores **s**on hojas de papel con símbolos que el software interpreta y de acuerdo a un marcador específico realiza una respuesta. La Fig. 6 muestra un ejemplo de AR dónde se puede apreciar la utilización de un marcador llamado framemarker, el cual es utilizado como eje para montar una imagen, y en este caso exponer imágenes de la zona céntrica de la entidad, además se puede observar que no se necesita ser un experto en algún área

técnica para utilizarla, ya que solo requiere utilizar los marcadores para interactuar con el software. La conjunción del software descrito anteriormente, así como el contar con ciertos conocimientos de programación y de diseño posibilitó hacer un software con AR basado en PDS.

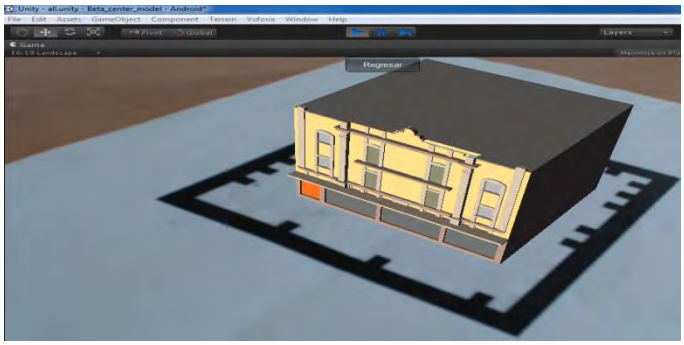


Fig. 5. Utilización de marcadores para interactuar con la aplicación de AR de nuestra propuesta.

El almacenamiento de la aplicación se lleva a cabo en la nube reduciendo así el tamaño considerablemente, lo cual permite agregar todas las fotografías necesarias para la aplicación. La Fig. 6 muestra una imagen de la propuesta descrita en el presente documento, aplicada al (Palacio Municipal) la cual denota gran interés histórico dentro del estado de Querétaro.

Fig. 6. Palacio Municipal de Querétaro.

La aplicación con AR funciona a través de diferentes dispositivos inteligentes, los cuales necesitan soportar AR, a su vez utilizan la combinación del Motor Unity y Vuforia para la creación de AR, capaces de cambiar parte de la realidad por modelos en tercera dimensión; el objetivo de la aplicación es recrear en tiempo de ejecución parte de las edificaciones de la entidad por medio de los teléfonos inteligentes. La aplicación con AR detecta algún edificio o algún paisaje, y en breve se procede a generar el bosquejo del terreno nuevo modelado en **Google Sketchup**.

Un punto esencial del desarrollo del proyecto fue conjuntar el mundo real y virtual del estado de Querétaro, con la finalidad de mejorar la interacción del usuario con el ambiente que lo rodea a través de AR, permitiendo así que el usuario pueda manipular parte del entorno por medio de la tecnología de AR, logrando enriquecer sus experiencias de aprendizaje.

5 Conclusiones y Trabajo Futuro

La simulación de la ciudad de Querétaro con AR, busca brindar experiencias más enriquecedoras a los turistas por medio de la AR, a través de dispositivos inteligentes que cuenten con soporte para AR. Este tipo de aplicaciones tiene la ventaja de que se puede seguir actualizando a lo largo del tiempo, debido a que se cuenta con un diseño robusto basado en patrones de diseño de software, además de seguir mejorando constantemente, y expandiéndose de forma indefinida, y no solo considerar a la ciudad de Querétaro, sino agregar a otras entidades o lugares turísticos de interés a nivel nacional. La propuesta se enfoca en el área turística, para enriquecer la estancia de los turistas de una forma agradable y enriquecedora.

Gracias a la tecnología, hoy en día es posible crear aplicaciones móviles con una facilidad increíble, ya que Android es bastante comercial, hay una información extensa del uso que se le puede dar a este sistema operativo para móviles. También se presta mucho para crear aplicaciones como la AR, pero debido a la poca información de esta tecnología es un poco difícil encontrar temas para su desarrollo, ya que hay pocos desarrolladores de este tipo de aplicaciones. La AR será lo nuevo en aplicaciones para los próximos años, y como toda aplicación para dispositivos inteligentes ya sea algún videojuego, aplicación interactiva, etc., solo nuestras capacidades, tanto mentales como el procesamiento de nuestros equipos, son el límite para crear nuevas aplicaciones innovadores con el uso de esta tecnología

Agradecimientos. Los autores de este artículo agradecen a la Universidad Politécnica de Querétaro, al Instituto Politécnico Nacional (IPN) y a la Escuela Superior de Cómputo (ESCOM) por su apoyo para este trabajo dentro de los proyectos SIP: 20140382 y 20140359, dentro del proyecto multidisciplinario 1665. Los autores desean reconocer a todos sus colegas y a los estudiantes que participaron en el diseño y desarrollo del software descritos en este artículo.

Referencias

[1] Alan B. Craig, Understanding Augmented Reality: Concepts and Applications, 1 edition, Morgan Kaufmann, 2013.
[2] Borko Furht, Handbook of Augmented Reality, Springer, 2011.
[3] Google Sky Map, “Google Mobile”, última fecha de modificación: 2014, disponible en el sitio Web con URL: http://www.google.com/mobile/skymap/.
[4] Goggles, “Google Goggles”, última fecha de modificación: 2014, disponible en el sitio Web con URL: https://support.google.com/websearch/answer/166331.
[5] TwittARound, “TwittARound”, última fecha de modificación: 2014, disponible en el sitio Web con URL: http://i.document.m05.de/twittaround/.
[6] Yelp Monocle, “Yelp Monocle”, última fecha de modificación: 2014, disponible en el sitio Web con URL: http://www.yelp.com/m%C3%A9xico-df.
[7] Rubén Peredo Valderrama, Alejandro Canales Cruz: Aplicaciones Web basadas en componentes de software para Educación Basada en Web. CNCIIC-ANIEI 2008.
[8] R. Peredo Valderrama, I. Peredo Valderrama, L. Balladares Ocaña, Sistema evaluador usando Web semántica para educación basada en Web, 5ta. Conferencia Iberoamericana en Sistemas, Cibernética e Informática (CISCI 2006), IIIS, 2006.
[9] R. Peredo Valderrama, I. Peredo Valderrama, L. Balladares Ocaña, Sistema generador de contenidos multimedia interactivos didácticos usando componentes de software para educación basada en Web, 6ta. Conferencia Iberoamericana en Sistemas, Cibernética e Informática (CISCI 2007), IIIS, 2007.

[10] Rubén Peredo, Alejandro Canales, Alain Menchaca, Iván Peredo, Intelligent Web-based education system for adaptive learning, Expert Systems with Applications, 38(12): 14690–14702, Pergamon Press, 2011.
[11] Unity, "Unity 3D", última fecha de modificación: 2013, disponible en el sitio Web con URL: http://unity3d.com/.
[12] Vuforia, "Vuforia Qualcomm Technologies", última fecha de modificación: 2013, disponible en el sitio Web con URL: http://www.qualcomm.com/solutions/augmented-reality.
[13] Monodevelop, "Monodevelop 4.0", última fecha de modificación: 2013, disponible en el sitio Web con URL: http://monodevelop.com/.
[14] SDK de Android, "SDK de Android Developers", última fecha de modificación: 2013, disponible en el sitio Web con URL: http://developer.android.com/sdk/index.html.
[15] Unity C#, "Unity C# ", última fecha de modificación: 2013, disponible en el sitio Web con URL:http://docs.unity3d.com/Documentation/Manual/VisualStudioIntegration.html.

Diseño y Desarrollo de un Videojuego Serio para Soporte a Terapias de Lenguaje Basado en Campos Semánticos

Ingeniería de Software

David Céspedes-Hernández[1], David Zavala-García[2], Francisco J. Álvarez Rodríguez[1], Juan Manuel González-Calleros[3], Liliana Rodríguez-Vizzuett[1], Roberto Guerrero-Bueno[4]
[1]Universidad Autónoma de Aguascalientes, Centro de Ciencias Básicas, Aguascalientes, México
[2]Universidad Autónoma Indígena de México, Unidad Mochicahui, Sinaloa, México
[3]Benemérita Universidad Autónoma de Puebla, Facultad de Ciencias de la Computación, Puebla, México
[4]Instituto Tecnológico de Acapulco, Departamento de Ingeniería en Sistemas, Guerrero, México
{dcespedesh, rey.12.91.30, fjalvar.uaa, lilianarv90, roberto.guebu90}@gmail.com, juan.gonzalez@cs.buap.mx

Resumen. Los videojuegos son utilizados como entretenimiento entre la población, sin embargo, existe una clasificación de éstos, denominados serios, los cuales son elaborados y utilizados para entrenar y educar. Al observar sesiones de terapia de lenguaje, se identifica el requerimiento de desarrollar un videojuego serio que sirva como apoyo y extensión al proceso de rehabilitación del habla. Para lograr este objetivo, se sigue un proceso de desarrollo que toma en cuenta las actividades realizadas en las sesiones conducidas por un especialista, para posteriormente, usando un marco de trabajo, elaborar modelos de interfaces de usuario que se deriven en la implementación de un videojuego serio. Los prototipos elaborados se evalúan siguiendo criterios de usabilidad.

Palabras clave: Videojuegos serios, Terapia de lenguaje, Rehabilitación.

1 Introducción

El uso de videojuegos se ha vuelto popular entre varios segmentos de la población, haciendo que la mayoría tenga acceso a ellos y los haya utilizado al menos de manera ocasional [1]. En la mayoría de los videojuegos, el jugador controla la acción de un personaje que se desarrolla en una realidad virtual recreada, basada en ciertas normas y en la cual hay un objetivo que debe alcanzarse [2]. Para la realización de este proyecto, es importante destacar que existe una categoría de videojuegos denominados serios, cuyo objetivo principal no es el de entretener sino el de ser una herramienta que permita educar, entrenar o informar al usuario [3]. Para comprender el dominio en que este proyecto se lleva a cabo es importante definir también el concepto de terapia de lenguaje, la cual, es un proceso que tiene como objetivo apoyar en dificultades al utilizar, articular y dar sentido al lenguaje [4]. La terapia de lenguaje forma parte de la rehabilitación, que es un proceso dinámico adaptativo que se lleva a cabo con el objetivo de cambiar condiciones de vida impuestas a un individuo [5].

Se plantea el objetivo general de llevar a cabo la elaboración de un videojuego serio para apoyar al especialista en terapia de lenguaje en la tarea de enriquecimiento del vocabulario de sus pacientes, utilizando imágenes y sonidos agrupados en campos semánticos, es decir,

en conjuntos de palabras que tienen alguna característica en común [6]. Los padecimientos cuya rehabilitación se desea apoyar son hipoacusia [7] y dislalia [8]. Por el tipo de herramienta de software que se desarrollará, se encuentra la necesidad de implementar interfaces de usuario altamente interactivas y usables [9]. Del objetivo general planteado y considerando además el trabajo llevado a cabo en [15], se desprenden los siguientes objetivos específicos: Consultar a un especialista en terapia de lenguaje para observar las sesiones de rehabilitación conducidas en pacientes con los padecimientos previamente mencionados. Documentar un estado del arte sobre aplicaciones existentes con objetivos similares, antecedentes y herramientas que se juzguen necesarias para la realización del proyecto. Con la asistencia del especialista, elaborar una lista de requerimientos para la aplicación a desarrollar. Diseñar modelos a partir de los requerimientos obtenidos para a partir de ellos implementar un videojuego que satisfaga las necesidades de los pacientes. Evaluar el prototipo implementado en términos de criterios ergonómicos.

El resto del artículo se organiza por secciones. La sección dos contiene el estado del arte, es decir, un reporte de los conceptos y antecedentes necesarios para la realización de este trabajo. La tercera sección contiene una descripción del proceso seguido para el desarrollo del prototipo de videojuego serio. Posteriormente, se presenta la sección de evaluación, para la que se considera la revisión del prototipo implementado en términos de una serie de criterios de usabilidad. En la sección cinco, se dan las conclusiones al proyecto en términos de los objetivos alcanzados y se menciona el trabajo futuro a realizar obtenido de la evaluación del prototipo implementado.

2 Estado del Arte

Como antecedente directo a este proyecto, se tiene la propuesta realizada en [15], en la que siguiendo un método formal, se diseña un videojuego serio para ejercitación de la discriminación de sonidos con el objetivo de favorecer y extender las sesiones que pacientes de terapia del lenguaje reciben. La evaluación del prototipo del videojuego elaborado, se lleva a cabo de acuerdo a la opinión de un experto en rehabilitación pero no considera una validación en cuanto a aspectos de usabilidad se refiere. Considerar usabilidad en el diseño de sistemas no sólo significa hacer sistemas amigables para el usuario sino de acuerdo con [16] consiste en buscar facilidad de aprendizaje, eficiencia, facilidad para recordar, carencia de errores y gusto subjetivo.

De acuerdo con sus características, se pueden utilizar diversos tipos de experimentos para evaluar la usabilidad del software como lo son: observación, cuestionarios, entrevistas y retroalimentación del usuario entre otros. Igualmente, el diseño de criterios ergonómicos es visto como un medio para la definición y operalización de dimensiones de usabilidad. Los factores que determinan la confiabilidad de dichos criterios son: 1) La calidad de las definiciones, 2) Lo explícito de la motivación, 3) La relevancia de los ejemplos provistos con cada criterio, y 4) Las distinciones dadas entre criterios que pueden ser conceptualmente cercanas unas a las otras. Otros requisitos con los que un método para evaluación por criterios ergonómicos debe cumplir son: basarse en el análisis de las interfaces y no en pruebas con el usuario debido a costos y tiempo; ser utilizable por personas con poca experiencia en factores humanos; ser explícito para permitir medición y suficientemente estandarizado para poder replicarse.

Así, el enfoque proporcionado por Bastien y Scapin [17] es confiable para realizar la evaluación del sistema que en este trabajo se desarrolla, ya que se realizó un experimento para medir la confiabilidad de los criterios que en él se definen, logrando crear un conjunto de criterios bien definidos y ejemplificados, de manera que los evaluadores puedan

identificarlos correctamente y asignarlos apropiadamente a problemas de diseño de interfaces. Los criterios ergonómicos pertenecientes al enfoque se describen a continuación. Compatibilidad: concordancia entre características del usuario, de las tareas y la organización de la salida, entrada y el dialogo de una aplicación. Coherencia: forma en que las decisiones del diseño de interfaces son mantenidas en contextos similares. Carga de trabajo: elementos de interfaz que juegan un rol en la reducción de la carga cognitiva o perceptual del usuario e incrementa la efectividad del dialogo. Adaptación: capacidad del sistema para comportarse contextualmente y de acuerdo a las necesidades y preferencias del usuario. Control de diálogo: procesamientos del sistema de acciones explicitas del usuario y el control que tienen los usuarios en el procesamiento de las mismas. Representatividad: califica la relación entre un término y/o un signo y su referencia, para determinar si facilitan la codificación y la retención. Guía: medios disponibles para aconsejar, orientar y guiar el usuario a través de su interacción con un sistema. Manejo de errores: evalúa la forma en la que el sistema se recupera de errores cometidos por el usuario.

En el presente proyecto, se hace uso de campos semánticos, los cuales, constituyen una herramienta de utilidad para la tarea de aumentar el vocabulario por medio de la relación de palabras de acuerdo a características en común, es importante señalar que en terapias de lenguaje observadas en el Centro Integral de Atención a la Salud de la Universidad Autónoma de Aguascalientes, el especialista conduce actividades apoyándose en ellos con el objetivo de que a la vez que los pacientes se rehabilitan de padecimientos del habla, se incremente la cantidad de palabras que manejan.

Para abordar el problema que supone el llevar a cabo el desarrollo de un videojuego que apoye o funja como herramienta auxiliar en las terapias de lenguaje, se identifica la necesidad de seguir una metodología de trabajo. Existen muchas metodologías para el desarrollo de videojuegos serios [10] [11] [12] pero en este caso, se tomará en cuenta la propuesta de [13] en donde se da como solución a este problema, la utilización del marco de trabajo CAMELEON [14] y se considerará también la adaptación de éste que se llevó a cabo en [15]. Así, se sigue un proceso que comienza con la observación y análisis de las sesiones de terapia de rehabilitación conducidas por un especialista, tomando nota de las actividades y ejercicios que se llevan a cabo, para después pasar a la elaboración de un modelo de tareas donde se definirán las actividades que se le permitirá al usuario realizar en la aplicación. Posteriormente, se elaboran Interfaces de Usuario Abstractas (AUI), las cuales son ajenas a un tipo de modalidad y a una plataforma. A partir de ellas, se elaboran Interfaces de Usuario Concretas (CUI), definiendo la modalidad de uso pero aun no la plataforma. Finalmente, se definen Interfaces de Usuario Finales (FUI), en las que se define la modalidad de interacción y la plataforma en la que el sistema funcionará.

3 Desarrollo

Para la realización de este proyecto se considera una actividad propuesta por un especialista en terapia de lenguaje, la cual se lleva a cabo en las sesiones de rehabilitación con pacientes en edad infantil diagnosticados con hipoacusia o dislalias con el objetivo de incrementar el vocabulario que manejan. Esta actividad, consiste en trabajar con campos semánticos, desarrollando la capacidad del paciente de relacionar objetos con sus nombres escritos y con la pronunciación de los mismos. Para cumplir con estos objetivos, se utilizan juegos de memoria formados por dos grupos de tarjetas, el primero de imágenes de objetos pertenecientes a un campo semántico específico y el segundo de textos correspondientes a los nombres de los mismos. Después de elegir el paquete de tarjetas que se utilizará, se le

solicita al niño que elija una de cada grupo, se le indica la pronunciación de las palabras correspondientes y se evalúa la forma en que los pares se forman. Una vez que se observó la actividad en sesiones de terapia, se lleva a cabo la identificación de los actores que en ella intervienen, siendo etiquetados como sistema (especialista) y usuario (paciente). Posteriormente, se deben definir las tareas que el usuario podrá realizar en el sistema, con respecto a ésto, se reconocieron las siguientes: jugar memoria, seleccionar paquete de tarjetas y seleccionar tarjeta. Igualmente se describen las tareas que el sistema llevará a cabo para interactuar con el usuario: mostrar paquetes de tarjetas disponibles, mostrar tarjetas para juego, mostrar imagen, reproducir sonido, evaluar la realización de pares y proporcionar retroalimentación. Adicionalmente, se agrega al modelo un menú que permitirá al usuario acceder a distintas áreas del juego. En la Fig, 1 se muestra el diseño del modelo de tareas elaborado usando notación CTTE [18].

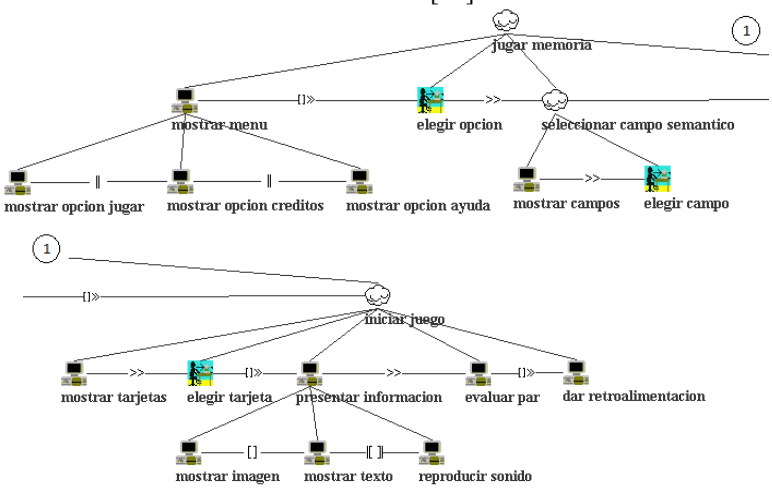


Fig 1. Modelo de tareas para la actividad descrita usando notación CTTE [18].

La tarea de nivel más alto, se denomina *jugar memoria*, es de naturaleza abstracta y representa a la actividad en su totalidad. Esta tarea se divide en otras cuatro, *mostrar menú*, *elegir opción*, *seleccionar campo semántico* e *iniciar juego*, las cuales a su vez están divididas en sub-tareas. *Mostrar menú* es una tarea automática que permite al usuario visualizar las diferentes secciones del sistema, se divide en *mostrar opción jugar*, *mostrar opción créditos* y *mostrar opción ayuda*. La tarea *elegir opción*, es de tipo interactivo y representa la capacidad del usuario para elegir la sección del sistema a la que desea dirigirse. *Seleccionar campo semántico* es una tarea de tipo abstracta que consiste en la presentación al usuario de los campos semánticos disponibles (*mostrar campos*) y de la elección del campo semántico a utilizar (*elegir campo*). Finalmente, la tarea *iniciar juego* de tipo abstracto, describe las actividades que el especialista lleva a cabo con el paciente en las sesiones de terapia convencionales pero en términos de sistema y usuario, es decir, el sistema muestra tarjetas, el usuario elige un par, se le muestra la información correspondiente (imagen, texto o sonido), se evalúa si formó correctamente un par y se le brinda la retroalimentación correspondiente.

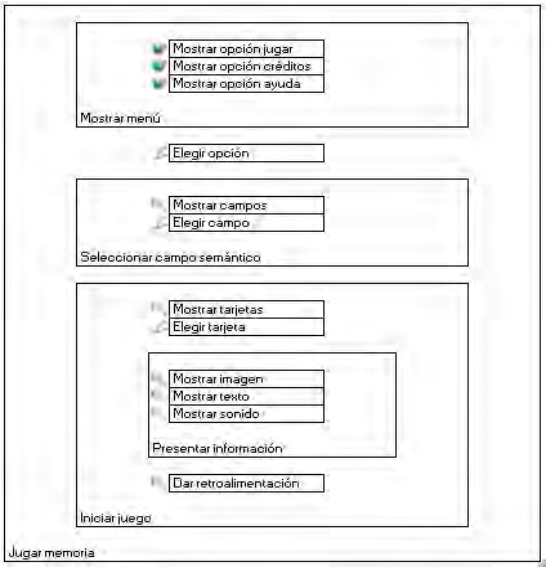


Fig. 2. Modelo de Interfaz de Usuario Abstracta para las actividades descritas.

Siguiendo los pasos del marco de trabajo elegido, a partir del modelo de tareas, se efectúa un proceso de concretización del que se obtiene la AUI mostrada en la Fig. 2. En este modelo, se puede observar un contenedor general correspondiente a la tarea de mayor jerarquía, a su vez las tareas que agrupan a otras se pueden observar en el modelo como contenedores, las tareas restantes se representan como íconos que describen la interacción que el usuario tendrá con los componentes del sistema, en términos de entradas de información, consultas de información y navegación, correspondientes a las tareas de más bajo nivel del modelo presentado en la Fig. 1. Después de haber definido la AUI, la siguiente actividad a llevar a cabo como parte del proceso de desarrollo consiste en la realización de la interfaz concreta considerando para este caso en particular, una modalidad de interacción gráfica. Los contenedores en la AUI se traducen en ventanas y agrupaciones de componentes, por su parte los elementos de interacción modelados, se convierten en botones, etiquetas y objetos interactivos que se presentarán al usuario en la interfaz final. La Fig. 3 muestra los modelos de CUI realizados. Con lo realizado hasta ahora, de acuerdo al marco de trabajo CAMELEON, lo que resta es realizar el diseño de IU Finales.

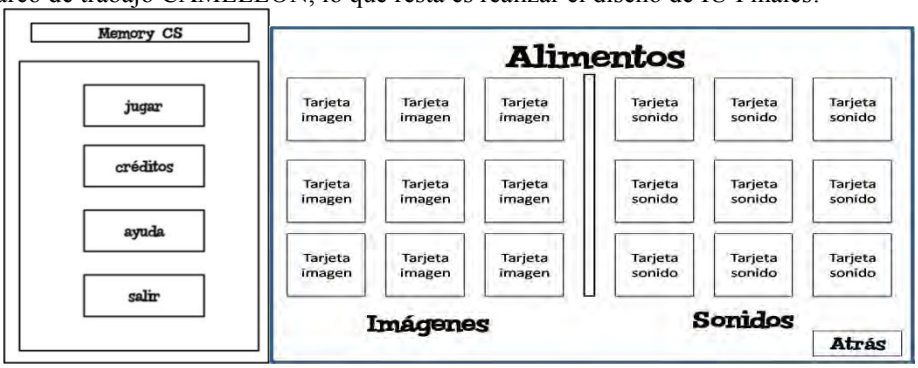


Fig. 3. Modelo de Interfaces Concretas de Usuario para el sistema diseñado.

Esta etapa involucra elegir la plataforma en la que el sistema funcionará. Se decide orientar el desarrollo del videojuego serio a tecnologías Web, para que pueda ser usado tanto en computadoras portátiles o de escritorio como en dispositivos móviles. Los prototipos de FUI generados se presentan en la Fig. 4.

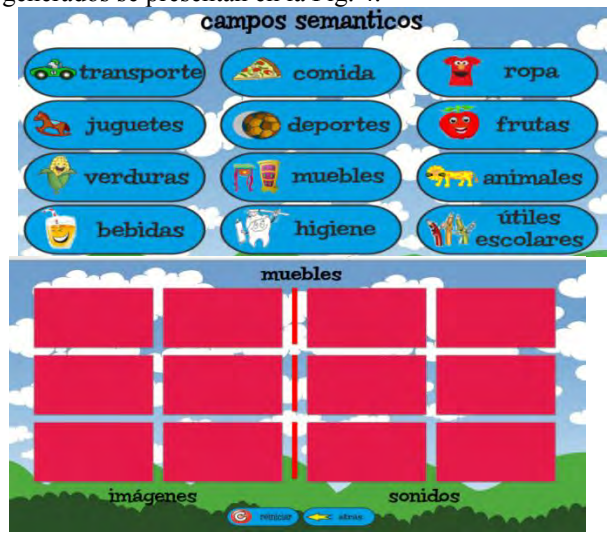


Fig. 4. Prototipos de Interfaz Final de Usuario para el videojuego serio desarrollado

Posteriormente al diseño de las FUI, se añade funcionalidad al sistema de manera que permita que las actividades identificadas como requerimientos puedan ser llevadas a cabo. La siguiente sección muestra los resultados de la evaluación realizada al sistema en terminos de usabilidad.

4 Evaluación

Para la validación del sistema, un experto en evaluación de usabilidad utilizó los criterios ergonómicos propuestos en [17] y descritos en el estado del arte. Los resultados de esta revisión para cada uno esos criterios se muestran en la Tabla 1.

Criterio	Evaluación
Compatibilidad	Se usan imágenes representativas de los objetos que se desea representar, los sonidos utilizados en el sistema son adecuados. Por otra parte, la aplicación no se comporta siempre igual.
Coherencia	No hay coherencia en el tamaño de fuente de los botones, hay problemas para distinguirlos.
Carga de trabajo	No hay problemas de brevedad y concisión, la carga cognitiva y perceptual es limitada tanto para entradas como para salidas, el número de acciones requeridas para lograr un objetivo es reducido, la densidad de la información es adecuada.
Adaptación	El sistema no lo permite pero tampoco lo requiere.
Control de diálogo	La propuesta muestra control de diálogo de tipo explícito.
Representatividad	Las imágenes son representativas, el sistema no requiere representatividad en el dialogo, ni en la presentación.

Guía	No es atendida por el sistema en cuanto a prompting, agrupamiento, retroalimentación o legibilidad.
Manejo de errores	El sistema es simple y el usuario no se equivoca en su uso.

Tabla 4. Resultado de la evaluación siguiendo los criterios ergonómicos de [17].

En la siguiente sección, se describen las conclusiones al proyecto en términos de los objetivos alcanzados y la evaluación llevada a cabo, así como el trabajo futuro a llevar a cabo.

5 Conclusiones y Trabajo Futuro

En el presente proyecto, se logró implementar un prototipo de un videojuego serio para soportar a la terapia de lenguaje, puntualmente para enriquecer el vocabulario de los pacientes a través de campos semánticos. Para el desarrollo, se consideraron los requerimientos expresados por el especialista y los obtenidos durante la observación de tareas llevadas a cabo en el consultorio. Posteriormente, se usó un marco de trabajo para la elaboración de modelos de interfaces de usuario altamente interactivas y usables y se implementó la parte funcional del juego. Los prototipos fueron evaluados en términos de usablidad, considerando una serie de criterios. De esta evaluación se conocieron aspectos con los que la aplicación en esta primera versión ya cuenta y se hallaron otros que deberán ser considerados en una siguiente iteración. Es importante aclarar que esta evaluación solo tiene que ver con la interfaz de usuario propuesta y es necesario hacer una con respecto a la jugabilidad del sistema y a su efectividad en torno a la rehabilitación. Como trabajo futuro, se deben atender los criterios en los que la evaluación no resultó positiva, además, se utilizará la aplicación elaborada para extender y apoyar las sesiones de terapiaconvencionales, tomando en cuenta que en ningún momento se busca remplazarlas. Una vez que la aplicación sea utilizada por usuarios, se podrán realizar otras evaluaciones a la misma, a través de la observación de la interacción con el usuario, así como entrevistas y cuestionarios a familiares [16].

Referencias

[1] de Aguilera, M. (2004). La institucionalización de una industria cultura. Los desafíos de la industria de los videojuegos. Telos: Cuadernos de comunicación e innovación, (59), 38-44.

[2] Gros, B. (2008). Videojuegos y aprendizaje. Aula de innovación educativa.

[3] Yildirim, S., Serious Game Design for Military Training, Proceedings of the Games: Design & Research Conference. Volda University College. Noruega, 2010, Michael y Chen (2006)

[4] Uc, R. D. R. C. (2008). Uso de Reconocimiento de Voz en un Juego Electrónico para la Rehabilitación de Niños con el Problema de Lenguaje Dislalia.

[5] Cleuren, L., "Speech Technology in Speech Therapy?", State of the Art and Onset to the development of a Therapeutic Tool to Treat Reading Difficulties in the First Grade of Elementary School. SLT Internship en ESAT-PSI Speech Group. 2003.

[6] Venegas, C., Laserna, P. C., & Urrea, C. (2008). The Right Setting. The Right Tools. The Right Teacher.

[7] Fitzpatrick, E., Graham, I. D., Durieux-Smith, A., Angus, D., & Coyle, D. (2007). Parents' perspectives on the impact of the early diagnosis of childhood hearing loss: Perspectiva de los padres sobre el impacto del diagnóstico temprano en la hipoacusia de la infancia. International journal of audiology,46(2), 97-106.

[8] Pascual García, P. (1988). La dislalia. Naturaleza, Diagnóstico y Rehabilitación. Madrid: CEPE.
[9] Shneiderman, B. (2005). Diseño De Interfaces De Usuario 4/E.
[10] Gómez, J. M. M., Marín, M. E. H., & Díaz, E. A. (2014). Enfoque Metodológico para el Diseño de Interfaces Durante el Ciclo de Vida de Desarrollo de Software. Revista GTI, 12(34).
[11] Valencia, R. E. C., Morales, A. F., & Morales, S. F. (2014). Diseño de un sistema para generar tablas semánticas. Vínculos, 10(2), 93-104.
[12] Laviola, J., Bringing VR and Spatial 3D Interaction to the Masses through Video Games, IEEE Computer Graphics and Applications.2008.
[13] Gonzalez-Calleros, J.M., A Model-Driven Approach for Developing Three-Dimensional User Interfaces of Information Systems in a Principle-based Way, PhD Thesis, Université catholique de Louvain Press. 2010.
[14] Calvary, G., Coutaz, J., Bouillon, L., Florins, M., et al., The CAMELEON Reference Framework, Deliverable 1.1, Version V1.1, CAMELEON Project Campbell, E., Maintaining accessible Websites with Microsoft Word and XML, Proc. of XML Europe 2003, XML Workshop Ltd., Londres. 2003.
[15] Céspedes-Hernández, D., Álvarez F., Muñoz-Arteaga, J., et al, Diseño de un Videojuego para Terapia de Lenguaje en Pacientes con Hipoacusia Bilateral Profunda con Implante Coclear, Technologies and Learning: Innovations and Experience. Tecnologías y Aprendizaje: Innovaciones y experiencias, CIATA, 2014.
[16] Nielsen, J. (1994). Usability engineering. Elsevier.
[17] Scapin, D. L., & Bastien, J. C. (1997). Ergonomic criteria for evaluating the ergonomic quality of interactive systems. Behaviour & information technology, 16(4-5), 220-231.
[18] Paternò, F., ConcurTaskTrees: An Engineered Notation for Task Models, en The Handbook of Task Analysis for Human-Computer Interaction. 2003. pp. 483-503.

Medición de Presencia Social en Sistemas Groupware Conscientes del Contexto

Adrián García-Arcos, Luis G. Montané-Jiménez, Carmen Mezura-Godoy
Facultad de Estadística e Informática, Universidad Veracruzana, Xalapa, México 91020.
zs12015363@estudiantes.uv.mx,
{lmontane, cmezura}@uv.mx

Abstract. Los Sistemas Groupware Conscientes del Contexto apoyan a los grupos de personas a alcanzar satisfactoriamente los objetivos en común, a través de la comunicación, cooperación y coordinación. Estos sistemas tradicionalmente incorporan mecanismos de razonamiento con variables centradas en el trabajo individual y no en el trabajo grupal, debido a que los estudios se han realizado para determinar el contexto de un usuario sin considerar el aspecto social de las tareas en las que participan. Estos mecanismos buscan adaptar de forma pertinente los servicios o la información dependiendo de la situación. Por lo tanto, en este trabajo se propone medir una variable compuesta de índole social llamada Presencia Social, que refleje la importancia de cada usuario en una actividad grupal y que pueda ser utilizada y aprovechada por estos mecanismos para mejorar la colaboración de los usuarios. Por lo tanto, este documento presenta un análisis de los trabajos de presencia social y el diseño de un método para medir la presencia social en un sistema groupware del tipo videojuego First-Person-Shooter.

Keywords: Trabajo Colaborativo Asistido Por Computadora, Groupware, Consciencia Contextual.

1 Introducción

Actualmente la tecnología permite a grupos de usuarios disponer de información y participar en la construcción de actividades colaborativas. Un área encargada del estudio de la colaboración es el Trabajo Cooperativo Asistido por Computadora (CSCW, por sus siglas en inglés Computer-Supported Cooperative Work), el cual se apoya en los sistemas groupware para proveer herramientas que permitan la coordinación, cooperación y comunicación de los usuarios para alcanzar satisfactoriamente sus objetivos en común. Estos sistemas incorporan mecanismos para la adquisición, gestión y uso de información contextual para desencadenar acciones inteligentes que apoyen de mejor forma la realización de las actividades sociales de los usuarios.

Los sistemas groupware están presentes en distintos dominios, como negocio, educación y entretenimiento, en este último un área de interés son los videojuegos colaborativos. Un ejemplo de estos sistemas es AssaultCube, un videojuegos de Disparos en Primera Persona (First Person Shooter - FPS) que incluye escenarios multijugador donde pueden llevarse a cabo actividades colaborativas, por ejemplo, la modalidad de "Captura de bandera", que implica tener que cumplir objetivos por los miembros de un grupo. En modalidades como estas, se genera información en el ambiente que puede ser utilizada por diversos mecanismos de razonamiento.

Tradicionalmente estos mecanismos utilizan información como ubicación, tareas, objetivos para configurar comportamientos inteligentes con respecto a las actividades realizadas, buscando automatizar los espacios de trabajo. Sin embargo, hasta el momento no se ha encontrado un mecanismo que permita medir una variable de un mayor nivel de abstracción que refleje el desempeño de los miembros de un equipo cuando trabajan en sus actividades - está medición de desempeño se conoce como Presencial Social. Por ello, se espera que el diseño e implementación de un mecanismo de medición con estas características permita observar nuevos comportamientos grupales que ayuden a mejorar la experiencia de los usuarios cuando colaboran.

Para la determinación del nivel de presencia social de los integrantes que participan en el desarrollo de una actividad, se propone inicialmente la recuperación de un conjunto de datos generados en los espacios de trabajo grupales, particularmente en el videojuego AssaultCube. En este documento se presenta una revisión de trabajos que abordan el tema de presencia social y se presenta un método respecto a la medición de la variable presencia social. El documento está organizado de la siguiente forma. En la sección 2 se introduce a los Sistemas Groupware Conscientes del Contexto y se presenta un análisis de trabajos relacionados con presencia social. En la sección 3 se propone el diseño de un mecanismo de razonamiento basado en el desempeño de los usuarios. En la sección 4 se explica el prototipo y se presentan variables que pueden ser utilizadas para la medición de presencia social. Finalmente la sección 5 presenta las conclusiones y los trabajos futuros explicando la información que se pretende evaluar para medir la presencia social.

2.1 Sistemas Groupware y Consciencia Contextual

Los sistemas Groupware tienen como finalidad proporcionar herramientas a equipos de trabajo para facilitar sus tareas en común. Clarence Ellis define el sistema groupware como "sistemas basados en computadoras que apoyan a grupos de personas que participan en una tarea en común (u objetivo) y que proporcionan una interfaz para un entorno compartido" [1]. Existen sistemas groupware que toman en cuenta la información de contexto que se genera en el ambiente para mejorar las actividades colaborativas, estos sistemas son llamados sistemas groupware conscientes del contexto.

Particularmente los videojuegos se han estudiado ampliamente con el fin de que el uso de la información contextual propicie mejores escenarios para la colaboración de los jugadores. Los sistemas groupware conscientes del contexto utilizan mecanismos de razonamiento porque permite proporcionar recursos y/o servicios de una forma inteligente a un usuario de acuerdo a su contexto, por ejemplo dependiendo la hora y la ubicación en donde se encuentre un usuario, su celular se configura a un modo en específico(por ejemplo: en silencio).

La incorporación de un mecanismo que mida la presencia social de un usuario utilizando información sensible a las actividades individuales y colaborativas del equipo podría servir para la construcción de mecanismos de razonamiento que suministren recursos para mejorar las actividades de comunicación, cooperación y coordinación dentro de un sistema groupware consciente de contexto.

2.2 Presencia Social

En las comunidades en línea, la capacidad social y tecnológica de los medios de colaboración juega un papel clave en la determinación del comportamiento individual [2]. Aquí es donde Presencia Social (PS) emerge como un elemento de diseño para el estudio de la colaboración en los sistemas groupware. En [4], definen PS como el grado de relevancia

de la otra persona en la ejecución de una interacción social, y la importancia de las relaciones interpersonales. Esta definición aborda elementos importantes que pueden ser considerados en los videojuegos. Una clasificación de presencia social se presenta en [5], que incluye tres categorías de respuestas del usuario: afectivas, interactivas y cohesivas, representados con 12 indicadores obtenidos a partir de una comunidad en línea donde los usuarios publican mensajes (véase la Tabla 1).

Categoría	Indicadores
Afectiva	Expresión de emociones, uso del humor.
Interactiva	Citando otros mensajes, Al referirse explícitamente a los mensajes de otros, hacer preguntas, expresar agradecimiento, Complementando, Expresando acuerdo.
Cohesiva	Direcciones o referencias al grupo utilizando pronombres inclusivos, saludos.

Tabla 1. Categorías de presencia social.

Varios trabajos [5], [4], [2], [6] han explorado la presencia social desde una perspectiva educativa, pero pocas obras han estudiado la presencia social desde una perspectiva de juegos. En [8] se presenta una crítica de cómo los estudiantes perciben los medios sociales para apoyar sus actividades de estudio, mientras que en [9] se menciona que el progreso tecnológico ha llevado a la creación de nuevos modelos para los diferentes tipos de comunicación que utilizan métodos síncronos y asíncronos de la enseñanza en línea. En escenarios de juego, hay estudios [10] que exploran esta cuestión de experimentar con los juegos móviles de colaboración, pero pocos trabajos abordan este problema desde los diferentes tipos de interacciones.

3 Propuesta para la Medición de Presencia Social

Después de haber realizado el análisis de los trabajos que abordan la presencia social, se propone una arquitectura conceptual y un método para la obtención de variables que puedan cuantificar la variable de presencia social.

3.1 Arquitectura

A continuación se presenta la adecuación de una arquitectura presentada en [10] para la incorporación de un mecanismo de medición de presencia social que sea utilizado en los Groupware. Estos elementos se derivan del análisis de actividades colaborativas en un videojuego colaborativo. La arquitectura propuesta en [10] propone cuatro capas para la representación de un sistema groupware consciente de contexto. La primera capa de "sistema groupware" es dónde existe la comunicación directa entre el sistema y los usuarios. La segunda capa de "uso de contexto" es para inferir y razonar sobre el contexto de los usuarios proveniente de la capa de "administrador de contexto". La tercera capa "administrador de contexto" se encarga de almacenar, actualizar y recuperar la información de la capa de "adquisición de datos". La capa de "adquisición de datos" es donde se adquiere la información desde proveedores internos (p.ej. sistema groupware) o externos

(p.ej. servicios web, sensores de luz, etc.). En la capa de "adquisición de información" los autores presentan un módulo donde se adquiere el valor de presencia social, sin embargo, la especificación del mecanismo de presencia social únicamente se presenta a un nivel conceptual. Por lo tanto, en este trabajo se pretende diseñar e implementar este componente de forma más detallada, incluyendo un mecanismo que permita obtener el valor de presencia social mediante la medición de variables en términos de cohesión (por ejemplo capturar bandera, mantener bandera) e interactividad (por ejemplo disparos efectivos, ver mapa).

3.2 Diseño del mecanismo de medición

En la Fig.1 se presenta un componente de medición de presencia social basado en el módulo de Adquisición de datos Contextuales de la arquitectura descrita anteriormente.

Se propone implementar un componente que mida la presencia social utilizando información contextual (variables para la comunicación, cooperación y coordinación).

El componente se encuentra estructurado en 4 capas que se describen a continuación. En la capa 1 de ordenamiento, los datos se obtienen de los proveedores contextuales. Se clasifican en datos cualitativos y cuantitativos. La capa 2 de procesamiento estadístico se encargará de aplicar métodos estadísticos para encontrar los datos con mayor relación. En la capa 3 de Agrupación de datos se clasificaran los datos con mayor relación en grupos de interactividad y cohesión. En la capa 4 de adquisición se obtendrá el valor de Presencia Social.

Fig. 1. Componente de medición de Presencia Social en términos de interactividad y cohesión.

En la Fig.2 se muestra el método que se utiliza para llevar a cabo la medición y tratamiento de los datos, posteriormente se describe el flujo de procesos mediante el cual se desea dar un valor significativo a la variable de presencia social. Posteriormente se pretende construir mecanismos de razonamiento que utilicen esta variable para mejorar la colaboración de los usuarios cuando intentan cumplir objetivos en común.

Por ejemplo, en el caso de un video-juego First Person Shooter, actualmente se pueden proporcionar recursos como municiones, mapas, resultados parciales. Sin embargo, con la creación de mecanismos de razonamiento que incluyan la presencia social podrían proveer recursos más significativos que sean adecuados al momento actual del grupo y de los

jugadores, ayudando a que desempeñan de manera más eficientemente las tareas colaborativas.

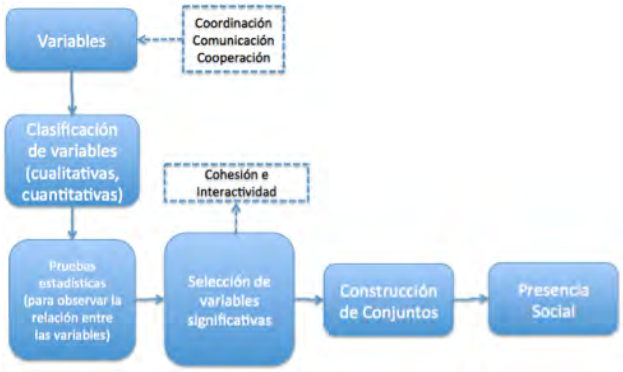


Fig. 2. Método de construcción de mecanismo de presencia social.

El método mostrado en la Fig. 2 sigue estos pasos: en la etapa uno se eligen las variables de acuerdo a la información que se requiera medir basadas en interacciones en el videojuego. En la etapa dos se clasifican las variables de acuerdo a su tipo y escala. En la etapa tres se realizan pruebas estadísticas para saber la relevancia de cada variable.

En la etapa cuatro se seleccionan las variables significativas de acuerdo a la relación con las demás variables en términos de cohesión e interactividad. Posteriormente en la etapa cinco se componen las variables de cohesión e interactividad construyendo conjuntos para cada una de ellas con las variables seleccionadas para en la etapa seis dar el valor de presencia social.

4 Prototipo

Para el caso de estudio se va a utilizar un videojuego de disparos en primera persona nombrado AssaultCube. Este videojuego es de código libre cuya licencia funciona bajo zliblike open source license y está basado en el motor de videojuegos Cube. Contiene 12 modalidades de juegos multijugador como: Partidas a muerte, sobreviviente, capturar bandera, mantener la bandera, entre otros (véase Figura 3). Éste videojuego ayudará a estudiar el ambiente colaborativo, es decir, la forma en que los jugadores se comunican, coordinan y cooperan para alcanzar metas en común.

Fig. 3. Escenario de videojuego AssaultCube.

En una primera etapa se han llevado a cabo experimentos exploratorios para obtener datos de variables contextuales generados en partidas del videojuego AssaultCube de forma local (en el mismo lugar).

En la realización de los experimentos han participado 16 personas las cuales se han dividido en partidas de seis personas divididas en dos equipos con el mismo número de integrantes, los integrantes del equipo son personas que oscilan entre los 19 y 29 años, de los cuales un 75% han sido hombres y un 25% mujeres.

El modo del videojuego AssaultCube que se eligió para llevar a cabo el estudio de las actividades colaborativas es el de "Captura de bandera", el cual consiste en que los equipos mantengan el mayor tiempo posible la bandera para ganar. Los puntos se adquieren cada vez que algún miembro del equipo conserve la bandera por un lapso de 15 segundos.

Mediante el análisis de los datos que se han obtenido durante los experimentos se ha observado que los jugadores tienen comportamientos similares mientras realizan actividades colaborativas. Se observó que por la forma en que se comunicaban y la interacción que desempeñaban se generaron roles que se adoptaron en los equipos. Por ejemplo, cuando el equipo tenía la bandera los roles observados fueron guardián, vigilante y abanderado, mientras que cuando no tenían la bandera los roles fueron explorador, protector y recuperador. Las interacciones del videojuego utilizadas durante las partidas se muestran en la Tabla 2.

Interacción	Descripción
Disparar	Activación del arma equipada actualmente.
Enviar mensaje	El jugador envía un mensaje a los miembros de su equipo con el comando de mensaje del juego.
Enviar mensaje de voz predefinida	El jugador puede escoger una de varias grabaciones de voz para avisar sobre varias situaciones a los demás jugadores.
Cambiar rol	El jugador camia de rol según la actividad que esté haciendo.
Auto eliminarse	El jugador se autodestruye con un arma explosiva.
Ser eliminado	La salud del jugador llega a 0 a consecuencia de fuego enemigo.
Eliminar oponente	La salud de un jugador del equipo contrario llega a 0 a consecuencia de disparos del jugador.
Capturar bandera	Un jugador tiene posesión de la bandera del equipo contrario.

Tabla 2. Interacciones observadas en AssaultCube.

Las interacciones observadas durante los experimentos se utilizarán como medidas estadísticas acumulativas para llevar a cabo los pasos que se proponen en el método, cuantificando el valor de la variable presencia social.

5 Conclusiones y trabajos futuros

Este artículo presenta un método para la medición de una variable que cuantifique la relevancia de los usuarios cuando participan en una actividad colaborativa, esta variable es

conocida como presencia social. Para ello fue necesario realizar un análisis de un videojuego de disparos en primera persona (FPS), el cual es un tipo de sistema Groupware. La finalidad de este análisis y estudio fue explorar el comportamiento e interacciones de los jugadores que participan en el proceso colaborativo. Con este estudio se identificaron elementos (roles, objetivos) que sirven como base para la construcción de un mecanismo de medición de presencia social utilizando información que se genera en el videojuego durante las partidas colaborativas.

Finalmente, el trabajo futuro consiste en la modificación del videojuego AssaultCube para incluir mecanismos de razonamiento que utilicen la presencia social del jugador para mejorar la experiencia del usuario. Por lo tanto, se espera hacer que el juego proporcione de una forma dinámica y eficiente medios que ayuden a los usuarios a tener un mejor desempeño en actividades de grupo. Para validar el método propuesto se establecerá un diseño experimental con las variables significativas en términos de cohesión e interactividad.

Reconocimiento. Los autores desean reconocer a los revisores anónimos de este artículo por sus útiles comentarios y sugerencias. El primer autor agradece a CONACYT su apoyo para la realización de sus estudios de posgrado (No. de becario: 366082).

Referencias

[1] Ellis, Clarence A., Gibbs, Simon J., Rein, Gail. Groupware: some issues and experiences. ACM,1991.

[2] K. N. Shen and M. Khalifa, "Design for social presence in online com- munities: a multi-dimensional approach," AIS Transactions on Human- Computer Interaction (1), pp. 33–54, 2008.

[3] J. Fulk, J. Schmitz, and C. W. Steinfield, "A Social Influence Model of Technology Use," in Organizations and communication technology. Sage, 1990, pp. 71–84.

[4] F. Biocca, C. Harms, and J. K. Burgoon, "Toward a more robust theory and measure of social presence: review and suggested criteria," Presence: Teleoperators and Virtual Environments, vol. 12, no. 5, pp. 456–480, Oct. 2003.

[5] L. Rourke, T. Anderson, D. R. Garrison, and W. Archer, "Assessing social presence in asynchronous text-based computer conferencing," Distance Education, vol. 14, no. 2, pp. 50–71, 1999.

[6] S. T. Bulu, "Place presence, social presence, co-presence, and satisfac-tion in virtual worlds," Computers and Education, vol. 58, no. 1, pp. 154–161, jan 2012.

[7] Y. A. W. de Kort, W. A. IJsselsteijn, and K. Poels, "Digital games as social presence technology: Development of the social presence in gaming questionnaire (spgq)," in Proc. of The 10th International Workshop on Presence (PRESENCE'07), Barcelona, Spain, oct 2007, pp. 195–203.

[8] S. Hrastinski and N. M. Aghaee, "How are campus students using social media to support their studies? an explorative interview study," Education and Information Technologies, vol. 17, no. 4, pp. 451–464, Dec. 2012.

[9] C. Roseth, M. Akcaoglu, and A. Zellner, "Blending synchronous face-to-face and computer-supported cooperative learning in a hybrid doctoral seminar," TechTrends, vol. 57, no. 3, pp. 54–59, May 2013

[10] Montané- Jiménez, Luis G. And Benítez-Guerrero, Edgard and Mezura-Godoy, Carmen. A Context-Aware Architecture for Improving Collaboration of Users in Groupware Systems. 9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom'13). 2013ent Classroom.